Learning to Optimize Meets Neural-ODE: Real-Time, Stability-Constrained AC OPF

Team Aletheia : Reenav Ray, Sachin Saini, Rahul, Nischay Sanjay Jiwankar, Aryan Mehra

Motivation

The AC OPF problem is fundamental to power grid operations, determining the optimal generator dispatch to meet real-time demand subject to physical constraints. Traditional numerical techniques solve this in 5-60 minutes. However, the rise of variable renewable energy demands real-time grid adjustments to maintain stability. While recent "Learning to Optimize" (LtO) approaches use neural networks to approximate AC OPF solutions in milliseconds, they only focus on steady-state physics. This paper argues that ignoring the dynamic behavior of synchronous generators (e.g. rotor angle and frequency) results in optimal solutions that are physically unstable, and lead to failure.

Neural Ordinary Differential Equations(NODEs)

A **NODE** models system dynamics by replacing the true differential equation with a neural network. For a dynamic system:

$$\frac{dx(t)}{dt} = p(x,t) \tag{1}$$

NODE approximates it as:

$$\frac{dx(t)}{dt} = N_\phi(x,t) \tag{2}$$

where N_ϕ is a neural network that learns the vector field. The state trajectory is computed using an ODE solver applied to N_ϕ.

In this paper, NODE predicts generator dynamics as:

$$\hat{\delta}_g(t), \hat{\omega}_g(t) = N_\phi\left(\hat{\delta}_g(0), \hat{\omega}_g(0), |\hat{V}_g|, \hat{\theta}_g\right) \tag{3}$$

This enables approximately $20\times$ faster dynamic prediction, stability constraint checking and end-to-end differentiable training for real-time stability-constrained OPF.

Methodology: DynOPF-Net

The authors propose **DynOPF-Net**, a differentiable technique that solves for cost optimality and dynamic stability simultaneously. This model is the first to integrate LtO architectures with **NODEs** for power systems. The architecture consists of two coupled components:

1. **The LtO Model (D_ψ):** A neural network that predicts the static optimal dispatch variables (voltage magnitude $|V|$, phase angles θ, and power injections) based on load demand (S^d).
2. **The Dynamic Predictors ($\mathcal{N}_\phi$):** A set of Neural ODEs, one for each generator, that approximate the "Swing Equations". These networks take the static dispatch from the LtO model and simulate the trajectory of the generator's rotor angle $\delta(t)$ and frequency $\omega(t)$.

Lagrangian Dual Learning

This framework uses a **Lagrangian Dual Learning** strategy to enforce constraints. The loss function includes standard terms for prediction error and penalty terms for physical violations. The model treats dynamic stability as a constraint: if the NODE simulation predicts that a generator's rotor angle will exceed a critical threshold (δ_{max}), a penalty is applied:

$$\mathcal{L}_c \propto \sum_{g\in\mathcal{G}} \max\left(0, \delta_g(t) - \delta_{\max}\right) \tag{4}$$

where $\mathcal{G}$ denotes the set of synchronous generators.

As the NODE solver is differentiable, gradients can backpropagate from the future simulated state violation back to the initial static dispatch. This allows the network to learn how to adjust current power settings to prevent future instability.

Conclusion

The authors prove that steady-state models alone are inadequate for real time power dispatch. DynOPF-Net demonstrates that machine learning can effectively handle the complex relationship between economic dispatch and grid dynamics.

arXiv:2410.19157v1 [eess.SY] 24 Oct 2024

LEARNING TO OPTIMIZE MEETS NEURAL-ODE: REAL-TIME, STABILITY-CONSTRAINED AC OPF

Vincenzo Di Vito*
University of Virginia
eda8pc@virginia.edu

Mostafa Mohammadian*
University of Colorado Boulder
mostafa.mohammadian@colorado.edu

Kyri Baker
University of Colorado Boulder
kyri@colorado.edu

Ferdinando Fioretto
University of Virginia
fioretto@virginia.edu

ABSTRACT

Recent developments in applying machine learning to address Alternating Current Optimal Power Flow (AC OPF) problems have demonstrated significant potential in providing close to optimal solutions for generator dispatch in near real-time. While these *learning to optimize* methods have demonstrated remarkable performance on steady-state operations, practical applications often demand compliance with dynamic constraints when used for fast-timescale optimization. This paper addresses this gap and develops a *real-time stability-constrained OPF* model (DynOPF-Net) that simultaneously addresses both *optimality* and *dynamical stability* within learning-assisted grid operations. The model is a unique integration of learning to optimize that learns a mapping from load conditions to OPF solutions, capturing the OPF's physical and engineering constraints, with Neural Ordinary Differential Equations, capturing generator dynamics, enabling the inclusion of a subset of stability constraints. Numerical results on the WSCC 9-bus and IEEE 57-bus benchmark systems demonstrate that DynOPF-Net can produce highly accurate AC-OPF solutions while also ensuring system stability, contrasting the unstable results obtained by state-of-the-art LtO methods.

1 Introduction

The AC Optimal Power Flow (AC-OPF) problem plays a fundamental role in power systems as it determines the optimal generator dispatch to meet demands while adhering to physical and engineering constraints. Traditionally, OPF problem instances are solved on timescales spanning 5 minutes to 1 hour, and often utilize linear approximations which do not satisfy steady-state AC power flow [1]. Additionally, the stochastic nature of renewable energy sources has increased uncertainty, necessitating more accurate power flow representations, and more frequent adjustments by system operators to preserve system stability [2]. Currently, these operations typically rely on heuristics for maintaining stable frequencies and voltages of the electrical generators, which lead to inefficiencies and higher losses. Rapid fluctuations of intermittent renewable generation, for example, can lead to curtailments as well as increasing emissions [3]. Similarly, governor droop control, which uses proportional control rules to modify the power generator based on frequency imbalances, introduces additional inefficiencies, losses, and emissions [4].

There are two key challenges within this setting. The *first* is the complexity of solving the full AC-OPF problem, which limits the frequency of these operations, and the *second* is the dynamic nature of the system, which impacts global system stability [5]. Indeed, relying solely on steady-state models for power dispatching operations is insufficient to guarantee compliance with system dynamic requirements. Nonetheless, adhering to these dynamics is not only computationally demanding on its own, but fully incorporating them within the AC-OPF model would introduce complex interdependencies among the problem variables, significantly increasing the computational complexity of an already challenging problem.

*Equal contribution.

In an effort to address the first challenge, there has been recent interest in applying machine learning (ML) based models to learn solutions mappings from load conditions to AC-OPF solutions [6]. These approaches have demonstrated enormous potential for replacing traditional numerical solvers to approximate complex supply/demand balance problems in near real-time. Central to these advancements is the concept of *Learning to Optimize* (LtO), [7] where neural network-based models serve as proxies for traditional numerical solvers to produce near-optimal solutions for parametric constrained optimization problems. However, while LtO approaches have shown promising results in solving the AC-OPF problem with high fidelity in real-time [8, 9], they address the *steady-state* problem, and may not be suitable to capture the dynamics of the system. In Section 6 the paper will show that, on the benchmark systems analyzed, many of the solutions generated by state-of-the-art LtO methods for AC-OPF violate the dynamic requirements of a synchronous electrical generator, which is unacceptable in practice.

To overcome these challenges, this paper proposes *Dynamic OPF-Net* (DynOPF-Net), a learning-based model that combines Learning to Optimize with ML-based dynamic predictors to solve real-time AC OPF problems while also addressing the dynamic stability of the generators. A key component of DynOPF-Net is the integration of neural ODEs (NODEs) [10] within the Learning to Optimize framework. NODEs approximate the continuous dynamics of systems through neural networks, allowing efficient modeling of system behaviors that evolve over time. The paper shows how this integration is able to produce near-optimal and stable solutions for a variety of operational conditions.

Contributions. This paper offers the following key contributions: **(1)** We introduce Dynamic OPF-Net (DynOPF-Net), a novel framework that seamlessly integrates machine learning with optimization techniques to incorporate generator dynamics directly into the AC-OPF model. **(2)** We provide empirical evidences which highlight the critical need to model the system dynamics within the OPF framework. Specifically, we demonstrate that existing LtO methods, which overlook these dynamics, systematically fail to satisfy stability requirements. In contrast, DynOPF-Net consistently generates solutions that adhere to these dynamic constraints, helping address stability. **(3)** We show that DynOPF-Net not only achieves a decision quality comparable to that of state-of-the-art LtO methods that address only steady-state constraints but also uniquely ensures compliance with dynamic requirements. This is achieved without sacrificing computational efficiency, offering the first solution, to our knowledge, for real-time power system operations under dynamic conditions.

2 Related Work

Learning methods for OPF In recent years, numerous ML-based approaches have been proposed to replace traditional OPF numerical solvers. Among the first attempts, [11] uses a statistical learning-based approach to predict DC-OPF solutions, while [12] proposes a DNN informed by a sensitivity to learn OPF solutions, which requires computing the sensitivities of OPF solvers with respect to load variables. Despite the promising results, these approaches did not focus on constraint satisfaction and thus may produce many infeasible solutions.

These deficiencies lead to developing methods that integrate the OPF problem structure within deep learning-based models, giving rise to *physics informed* or *learning to optimize* methods for OPF[1]. In particular, [13] uses a DNN to identify the active constraint sets to simplify and enhance learning DC OPF solution mappings. A Lagrangian-Dual deep learning-based approach was introduced in [8], which aims to learn near-optimal solutions while also encouraging satisfaction of the OPF constraints. Their approach modifies the neural network training by parameterizing the loss function with constraint penalty terms proportional to the degree of the predicted constraint violations, mimicking a dual ascent method [14]. Similarly, the method proposed in this paper uses dual penalty terms to drive the learning model towards solutions which are feasible and stable. There has been since then a number of approaches developing on and improving these techniques, including recurrent architectures [15] and decomposition methods [16]. While these methods produce state-of-the-art results for AC OPF, they do not guarantee strict compliance with feasibility requirements. Hence, recent work in ML-driven OPF solvers focuses on ensuring constraint satisfaction. Specifically, [17] predicts partial OPF solutions and subsequently resolve the remaining variables by addressing balance flow equations. In [9], this approach is built upon by employing implicit layers, enabling the correction of inequalities and the completion of equality constraints within the training process.

While these works have shown that machine learning methods can produce near-optimal OPF solutions, they focus on the *steady-state* dispatch problem, ignoring power system dynamics. This fundamentally limits applicable timescales (and speed improvements) from these methods. To the best of our knowledge, this paper is the first to introduce a machine learning-based method for solving AC OPF while simultaneously integrating synchronous generator stability.

[1] Here, we are going to use the latter term, as physics informed neural networks refers to another concept used to approximate PDEs with neural networks, as we will discuss later in this paper.

Learning methods for system dynamics The system dynamics typically take the form of a set of Ordinary (ODEs) or Partial Differential Equations (PDEs), which describe the laws governing the state variables of the system. When the number of ODE variables is large, the computational complexity renders precise numerical solutions impractical for real-time applications [18], where instead highly accurate approximations of the system of ODEs (PDEs) are preferred.

In this context, several works proposed physics-informed neural networks (PINNs) [19] to embed governing equations within the training of a ML model. PINNs have been shown highly effective in approximating various systems of complex PDEs at extremely fast inference times. In the power system literature, PINNs have mainly been used in single-machine infinite bus systems to estimate power system state variables and unknown parameters [20]. While PINNs offer significant modeling advantages by integrating physical behaviors within the model, they face training stability challenges when applied to complex systems, such as power grids [21]. In addition, as PINNs are designed to approximate the solution of *specified* system of differential equations, they suffer from limited generalization capability [22]. This severely limits their applicability in the context studied in this paper where the system dynamics can vary, and thus neural surrogates are required to capture *family* of dynamics.

To address this challenge, researchers have recently developed neural differential equations [10]. Among those, neural ODEs learn the system dynamics by approximating the vector field with a neural network. This capability makes NODEs well-suited as dynamic predictors for parametric systems of ODEs. In the context of power system dynamics, this allows for stability analysis under various operational conditions. In the power system literature, NODEs have been adopted for inferring critical state information of the power system dynamics [23]. To the best of the authors' knowledge, ours is the first attempt to integrate NODE models as dynamic predictors within the AC OPF problem.

3 Problem Formulation

This section introduces the stability-constrained AC OPF problem, starting from two key components: the (steady state) AC OPF problem and the synchronous generator dynamics. The paper adopts the following notation: lowercase symbols are used for scalars, bold symbols represent vectors, and uppercase symbols represent complex variables, denoted either in rectangular or polar form. Sets are denoted with standard calligraphic symbols (e.g., $\mathcal{X} = \{x_1, \ldots, x_n\}$), and special calligraphic symbols are reserved for models, such as a deep neural network parameterized by vector $\boldsymbol{\phi}$, denoted as $\mathscr{X}_{\boldsymbol{\phi}}$.

AC Optimal Power Flow. The AC OPF problem determines the most cost-effective generator dispatch that satisfies demand within a power network subject to various physical constraints. Typically, the setting focuses on a snapshot of the OPF problem in time. In a power network, represented as a graph $(\mathcal{N}, \mathcal{L})$, the node set $\mathcal{N}$ consists of n buses, and the edge set $\mathcal{L}$ comprises l lines. Additionally, $\mathcal{G}$ is used to represent the set of synchronous generators in the system. The AC power flow equations use complex numbers for current I, voltage V, admittance Y, and power S, interconnected through various constraints. The power generation and demand at a bus $i \in \mathcal{N}$ are represented by $S_i^r = p_i^r + jq_i^r$ and $S_i^d = p_i^d + jq_i^d$, respectively, while the power flow across line ij is denoted by S_{ij}, and θ_i describes the phase angles at bus $i \in \mathcal{N}$. The Kirchhoff's Current Law (KCL) is represented by $I_i^r - I_i^d = \sum_{(i,j)\in\mathcal{L}} I_{ij}$, Ohm's Law by $I_{ij} = Y_{ij}(V_i - V_j)$, and AC power flow denoted $S_{ij} = V_i I_{ij}^*$. These principles form the AC Power Flow equations, described by equation 1f and equation 1g in Model 1. The goal is to minimize a function equation 1a representing dispatch costs for each generator. Constraints equation 1b-equation 1c represents voltage operational limits to bound voltage magnitudes and phase angle differences, while equation 1d-equation 1e set boundaries for generator output and line flow. Constraint equation 1h sets the reference phase angle. Finally, constraints equation 1f and equation 1g enforce KCL and Ohm's Law, respectively.

Note that formulation alone does *not* capture synchronous generator dynamics, although stability-aware linearized OPF formulations have been developed [24].

Synchronous Generator Model. To capture the dynamic behavior of synchronous generators with high fidelity, this study considers the *classical* "machine model" [25], a simplified version of the "two-axis model". The dynamics of a synchronous generator $g \in \mathcal{G}$ are defined as:

$$\frac{d}{dt}\begin{bmatrix} e_q'^g(t) \\ e_d'^g(t) \\ \delta^g(t) \\ \omega^g(t) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \omega_s(\omega^g(t) - \omega_s) \\ \frac{1}{m^g}(p_m^g - e_d'^g(t) i_d^g(t) - e_q'^g(t) i_q^r(t) - d^g(\omega^g(t) - \omega_s)) \end{bmatrix}, \tag{2}$$

where stator currents $i_d^g(t)$ and $i_q^g(t)$ in the reference frame of machine $g \in \mathcal{G}$, are computed as:

$$\begin{bmatrix} i_d^g(t) \\ i_q^r(t) \end{bmatrix} = \begin{bmatrix} 0 & -x_d'^g \\ x_d'^g & 0 \end{bmatrix}^{-1} \begin{bmatrix} e_d'^g(t) - |V_g| \sin(\delta^g(t) - \theta_g) \\ e_q'^g(t) - |V_g| \cos(\delta^g(t) - \theta_g) \end{bmatrix}. \tag{3}$$

Model 1 The AC Optimal Power Flow Problem (AC-OPF)

$$\textbf{variables:} \quad S_i^r, V_i \;\; \forall i \in \mathcal{N}, \quad S_{ij} \;\; \forall (i,j) \in \mathcal{L}$$

$$\textbf{minimize:} \quad \sum_{i \in \mathcal{G}} c_{2i}(\mathrm{Re}(S_i^r))^2 + c_{1i}\,\mathrm{Re}(S_i^r) + c_{0i} \tag{1a}$$

$$\textbf{subject to:} \quad v_i^l \le |V_i| \le v_i^u \;\; \forall i \in N \tag{1b}$$

$$-\theta_{ij}^{\Delta} \le \angle(V_i V_j^*) \le \theta_{ij}^{\Delta} \;\; \forall (i,j) \in \mathcal{L} \tag{1c}$$

$$S_i^{rl} \le S_i^r \le S_i^{ru} \;\; \forall i \in \mathcal{N} \tag{1d}$$

$$|S_{ij}| \le s_{ij}^u \;\; \forall (i,j) \in \mathcal{L} \tag{1e}$$

$$S_i^r - S_i^d = \textstyle\sum_{(i,j)\in L} S_{ij} \;\; \forall i \in \mathcal{N} \tag{1f}$$

$$S_{ij} = Y_{ij}^* |V_i|^2 - Y_{ij}^* V_i V_j^* \;\; \forall (i,j) \in \mathcal{L} \tag{1g}$$

$$\theta_{\text{ref}} = 0 \tag{1h}$$

For a generator $g \in \mathcal{G}$, the state vector $\begin{bmatrix} e_q'^g & e_d'^g & \delta^g & \omega^g \end{bmatrix}^\top$ describes its voltage components, $e_q'^g$ and $e_d'^g$, associated with the 'd' and 'q' axis (dq-axes of a reference frame for g), its rotor angle δ^g, and its angular frequency ω^g. The synchronous angular frequency is denoted as ω_s in Equation equation 2. The mechanical power p_m^g from the shaft turbine serves as the control input. Constitutive machine parameters are given by $\begin{bmatrix} x_d'^g & m^g & d^g \end{bmatrix}^T$, where $x_d'^g$ denotes the the transient reactance, m^g the machine's inertia constant and d^g the damping coefficient. Importantly, stator currents are connected to the terminal voltage magnitude $|V_g|$ and phase angle θ_g via equation 3. In this model, voltage components $e_d'^g(t)$ and $e_q'^g(t)$ are held constant, meaning the update function for these states is zero. The interested reader is referred to [25] for a more in-depth discussion of the classical machine model. Given the assumptions on the voltage $e_d'^g$ and $e_q'^g$, the dynamic model of synchronous generators results in:

$$\frac{d}{dt}\begin{bmatrix} \delta^g(t) \\ \omega^g(t) \end{bmatrix} = \begin{bmatrix} \omega_s(\omega^g(t) - \omega_s) \\ \frac{1}{m^g}\left(p_m^g - d^g(\omega^g(t) - \omega_s) - \frac{e_q'^g(0)|V_g|}{x_d'^g}\sin(\delta^g(t) - \theta_g)\right) \end{bmatrix}. \tag{4}$$

Initial Values of Rotor Angles and EMF Magnitudes. For each generator $g \in \mathcal{G}$, the initial values of rotor angle $\delta^g(0)$ and electromotive force (EMF) $e_q'^g(0)$ are derived from the active and reactive power equations, assuming the dynamical system equation 4 initially being in a steady state condition, $\frac{d}{dt}\left[\delta^g(t) \quad \omega^g(t)\right]_{t=0}^T = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$, from which:

$$\frac{e_q'^g(0)|V_g|\sin(\delta^g(0) - \theta_g)}{x_d'^g} - p_g^r = 0, \tag{5}$$

$$\frac{e_q'^g(0)|V_g|\cos(\delta^g(0) - \theta_g) - |V_g|^2}{x_d'^g} - q_g^r = 0. \tag{6}$$

Following the same assumption, it follows that

$$\omega^g(0) = \omega_s. \tag{7}$$

Stability Limit. To guarantee stability of a synchronous generator $g \in \mathcal{G}$, the rotor angle $\delta^g(t)$ is required to remain below an instability threshold $\delta^{\max}$, as defined by SIngle Machine Equivalent (SIME):

$$\delta^g(t) \le \delta^{\max} \quad \forall t \ge 0. \tag{8}$$

Unstable conditions arise when violating equation 8, which is the principal binding constraint that necessitates re-dispatching.

Stability-Constrained OPF. The *stability-constrained* OPF problem is thus formulated in Model 2. This problem determines the optimal power dispatch subject to physical, engineering, and dynamic stability constraints equation 9b-equation 9h. Given a load demand $\boldsymbol{S}^d$, the objective is to find the OPF variables $\boldsymbol{S}^r, \boldsymbol{V}$ that minimize the cost function equation 9a while satisfying the constraints equation 9b-equation 9h. The inclusion of the dynamic behavior of synchronous generators complicates the problem due to the coupling between the OPF variables S_g^r, V_g and the generator state variables $\delta^g(t), \omega^g(t)$, as shown in equation 9d-equation 9f. Additionally, the non-linear nature of the generator model and the time dependencies of the state variables further increase the complexity. As a result, traditional numerical optimization algorithms are unable to handle, in a computationally viable way, the stability-constrained AC-OPF. To address this challenge, we develop a learning-based dual framework integrating NODEs with LtO models.

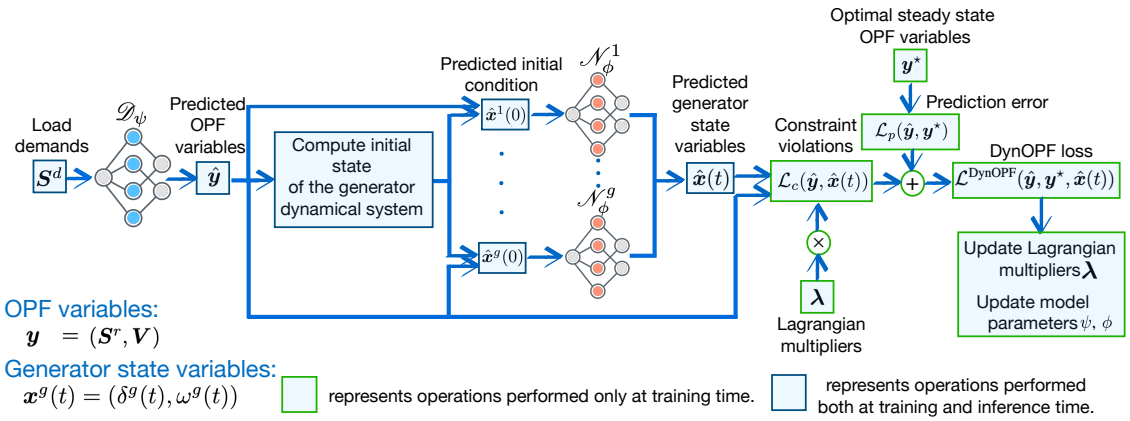


Figure 1: DynOPF-Net uses a dual network architecture consisting of a LtO model $\mathscr{D}_\psi$ which takes as input a load demand $\boldsymbol{S}^d$ and output an estimate the OPF variables $\hat{\boldsymbol{y}}$, based on which neural-DE models $\mathscr{N}_\theta^g$ output the generators state-variables $\hat{\boldsymbol{x}}^g(t)$. These estimates are used to compute the constraint violations $\mathcal{L}_c(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}(t))$ and the prediction error $\mathcal{L}_p(\hat{\boldsymbol{y}}, \boldsymbol{y}^\star)$ terms, which sum constitutes the total loss function $\mathcal{L}^{\text{DynOPF}}(\hat{\boldsymbol{y}}, \boldsymbol{y}^\star, \hat{\boldsymbol{x}}(t))$.

Model 2 The Stability Constrained AC-OPF Problem

$$\textbf{variables:} \quad S_i^r, V_i \ \ \forall i \in \mathcal{N}, \quad \delta^g(t), \omega^g(t) \ \ \forall g \in \mathcal{G}, \\ S_{ij} \ \ \forall (i,j) \in \mathcal{L}$$

$$\textbf{minimize:} \quad \sum_{i \in \mathcal{G}} c_{2i} (\text{Re}(S_i^r))^2 + c_{1i} \, \text{Re}(S_i^r) + c_{0i} \tag{9a}$$

$$\textbf{subject to:} \quad (1b) - (1h) \tag{9b}$$

$$\frac{d\delta^g(t)}{dt} = \omega_s(\omega^g(t) - \omega_s) \ \ \forall g \in \mathcal{G} \tag{9c}$$

$$\frac{d\omega^g(t)}{dt} = \frac{1}{m^g}\left(p_m^g - d^g(\omega^g(t) - \omega_s)\right) \\ - \frac{e_q'^g(0)|V_g|}{x_d'^g m^g} \sin(\delta^g(t) - \theta_g) \ \ \forall g \in \mathcal{G} \tag{9d}$$

$$\frac{e_q'^g(0)|V_g| \sin(\delta^g(0) - \theta_g)}{x_d'^g} - p_g^r = 0 \ \ \forall g \in \mathcal{G} \tag{9e}$$

$$\frac{e_q'^g(0)|V_g| \cos(\delta^g(0) - \theta_g) - |V_g|^2}{x_d'^g} - q_g^r = 0 \ \ \forall g \in \mathcal{G} \tag{9f}$$

$$\omega^g(0) = \omega_s \ \ \forall g \in \mathcal{G} \tag{9g}$$

$$\delta^g(t) \leq \delta^{\max} \ \ \forall g \in \mathcal{G}. \tag{9h}$$

4 Dynamic OPF-Net

Given a load profile $\boldsymbol{S}^d = (\boldsymbol{p}^d, \boldsymbol{q}^d)$, the goal is to *predict* the generators set-points that minimize the objective equation 9a while simultaneously satisfying the operational and dynamic constraints equation 9b-equation 9h of Problem 2. A significant challenge in this learning task is ensuring the satisfaction of both steady-state equation 9b and dynamic constraints equation 9h. To address this challenge, we propose *Dynamic OPF-Net* (DynOPF-Net), a dual neural network architecture consisting of a learning-to-optimize model (cyan network in Fig. 1) and a collection of NODE models, one for each generator, as illustrated in the orange networks of Fig. 1. The LtO model, denoted as $\mathscr{D}_\psi$, where ψ represents its parameters, predicts the OPF variables $\boldsymbol{S}^r = (\boldsymbol{p}^r, \boldsymbol{q}^r)$ and $\boldsymbol{V}$. Each NODE model $\mathscr{N}_\phi^g$, parametrized by ϕ, captures the system dynamics of the corresponding generator $g \in \mathcal{G}$. Both the steady-state and dynamic constraints are integrated into the learning task using a Lagrangian Dual Learning framework [8] as reviewed in Section 4.2.

4.1 Approximating Synchronous Generators Dynamics

This section first focuses on approximating the dynamics of synchronous generators (see Equation equation 4) using NODE models [10]. This proxy for a numerical differential solver allows us to infer generator state variables $\delta^g(t), \omega^g(t)$ for each generator g at fast computational times.

Consider a generic ODE,

$$\begin{cases} \frac{d\boldsymbol{x}(t)}{dt} = \boldsymbol{p}(\boldsymbol{x}, t) \\ \boldsymbol{x}(0) = \boldsymbol{x}_0, \end{cases} \tag{10}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ describes the n-dimensional vector of state variables, $\boldsymbol{x}_0 \in \mathbb{R}^n$ the initial conditions, $t \geq 0$ the time, and $\boldsymbol{p}\colon \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}^n$ a vector field. A neural ODE defines the continuous dynamics of system equation 10 by replacing the vector field $\boldsymbol{p}(\boldsymbol{x}, t)$ with a neural network $\mathscr{N}_\phi(\boldsymbol{x}, t)$, such that $\boldsymbol{p}(\boldsymbol{x}, t) \approx \mathscr{N}_\phi(\boldsymbol{x}, t)$.

From a practical standpoint, the forward pass of a NODE model is computed using a numerical algorithm similar to how a traditional ODE solver computes the numerical solution (by iteratively applying an update step starting from the initial condition $\boldsymbol{x}(0)$). However, instead of evaluating the true vector field $\boldsymbol{p}$, it uses the neural network $\mathscr{N}_\phi$. This approximation simplifies the computation of the update step in a numerical ODE solver, as it does not require evaluating the vector field $\boldsymbol{p}$ but only its approximation $\mathscr{N}_\phi$, thus enabling fast computation of the forward pass [10]. This results in a significant speed advantage over traditional ODE solvers that use the true $\boldsymbol{p}$.

Training a NODE $\mathscr{N}_\phi$, involves a dataset $\mathcal{D}$ of pairs $(\boldsymbol{x}_0, \boldsymbol{x}(t))$ to optimize the following empirical risk problem:

$$\underset{\phi}{\text{Minimize}} \;\; \mathbb{E}_{(\boldsymbol{x}_0, \boldsymbol{x}(t)) \sim \mathcal{D}} ||\hat{\boldsymbol{x}}(t) - \boldsymbol{x}(t)||^2 \tag{11a}$$

$$s.t. \;\; \hat{\boldsymbol{x}}(t) = \textit{ODEsolver}(\mathscr{N}_\phi, \boldsymbol{x}_0, \Delta_t) \tag{11b}$$

$$\boldsymbol{x}(t) = \textit{ODEsolver}(\boldsymbol{p}, \boldsymbol{x}_0, \Delta_t). \tag{11c}$$

Therein, *ODEsolver* denotes a numerical algorithm such as Euler's implicit and Δ_t is the integration time step. In this paper, NODE models are used to learn the solutions of a family of generator dynamics given by Equation equation 4. They can be trained as described in equation 11, with the difference that the numerical solver solutions $\boldsymbol{x}(t)$ are obtained from different instances of the generator model by varying the OPF variables which influence the initial conditions $\boldsymbol{x}_0$ and governing equations equation 4 (denoted with $\boldsymbol{p}$ in equation 11c). Due to the interaction between the generator state variables $\delta^g(t), \omega^g(t)$ and the OPF variables $|V_g|, \theta_g$, it is useful to express the NODE estimates as a function of the initial conditions $\boldsymbol{x}^g(0) = (\delta^g(0), \omega^g(0))$ and the estimated OPF voltages variables $\hat{V}_g, \hat{\theta}_g$ of Model 2:

$$\hat{\delta}^g(t), \hat{\omega}^g(t) = \mathscr{N}_\phi^g(\hat{\delta}^g(0), \hat{\omega}^g(0), |\hat{V}_g|, \hat{\theta}_g). \tag{12}$$

Here $|\hat{V}_g|$ and $\hat{\theta}_g$ are the (approximate) voltage magnitude and angles associated with bus of generator g as predicted by a learning-to-optimize model $\mathscr{D}_\psi(\boldsymbol{S}^d)$, as discussed in the following section. Note that, as these predicted values depend on the particular stability-constrained OPF problem instance and are not known a priori, the NODE model has to be able to produce accurate state variable estimates for a *range* of predicted $|\hat{V}_g|$, $\hat{\theta}_g$ values, which (feasible) bounds are given by Constraints equation 1b and equation 1c. Given these predicted values, the estimated initial state variables $\hat{\delta}^g(0)$ and $\hat{\omega}^g(0)$ are instead defined by Equations equation 5–equation 7. Note that the estimated OPF quantities $|\hat{V}_g|$ and $\hat{\theta}_g$ not only influence the initial state variables $\delta^g(0), \omega^g(0)$, but also impact the governing equation of the generator model, formalized in Equation equation 4.

Equation equation 12 implies that we consider an *augmented* generator model, where two additional state variables, $|V_g|(t)$ and $\theta_g(t)$, with no dynamics (i.e., $\frac{d|V_g|(t)}{dt} = 0$, $\frac{d\theta_g(t)}{dt} = 0$), and initial condition $|V_g|(0) = |\hat{V}_g|, \theta_g(0) = \hat{\theta}_g$ are incorporated alongside the actual state variables $\delta^g(t)$ and $\omega^g(t)$. This approach allows us to explicitly inform the NODE of the role played by V_g on the dynamics of each generator.

This setup establishes the foundation for training $\mathscr{N}_\phi^g$ to learn a family of generator dynamics, defined by different *extended* initial condition vector $[\hat{\delta}^g(0)\ \hat{\omega}^g(0)\ |\hat{V}_g|\ \hat{\theta}_g]^T$ and different instances of its governing equations.

4.2 Learning Stability-constrained ACOPF Solutions

We are now ready to discuss the interaction between the Neural ODEs $\mathscr{N}_\phi^g$ capturing the generators dynamics and the learning to optimize model $\mathscr{D}_\phi$ that predicts the OPF dispatch values. The synergistic integration of these components allows us to incorporate both static and dynamic constraints into the learning process, ensuring that the predicted solutions satisfy the stability-constrained AC OPF requirements.

Training this learning task involves a dataset $\{(\boldsymbol{S}^{d,i}, \boldsymbol{y}^{\star,i})\}_{i=1}^{n_{\text{obs}}}$, where each of the n_{obs} data points describes the observations of load demands ($\boldsymbol{S}^d$) and the corresponding optimal values of the OPF variables $\boldsymbol{y}^\star = (\boldsymbol{S}^{\star,r}, \boldsymbol{V}^\star)$, under the assumption that all synchronous generators are at a *steady-state*.

As shown in Figure 1, given a load demand $\boldsymbol{S}^d$, the DynOPF-Net' LtO model $\mathscr{D}_\psi$ produces an estimate of the OPF variables $\hat{\boldsymbol{y}} = \mathscr{D}_\psi(\boldsymbol{S}^d)$. To ease notation, in the following, we denote with $\hat{\boldsymbol{x}}^g(t) = \mathscr{N}_\phi^g(\hat{\boldsymbol{x}}^g(0), \hat{V}_g)$ the NODE estimate of the generator g's state variables, highlighting they are a function of the estimated initial conditions $\hat{\boldsymbol{x}}^g(0)$ and predicted voltage $\hat{V}_g$) (see Equation equation 12). Additionally, $\boldsymbol{x}(t)$, denotes all the generators' state variables, while $\hat{\boldsymbol{x}}(t)$ denotes the state variable estimated by NODE models $\mathscr{N}_\phi^g$ associated with each generator g in the network.

The dynamics learned by each NODE are integrated with the LtO predictions exploiting a Lagrangian dual framework [8]. For a given constrained optimization problem, in Lagrangian duality, a subset of the problem's constraints is relaxed into the objective function, each associated with a multiplicative weight called a *Lagrange multiplier*. This modified objective function is known as the *Lagrangian dual* function. To find the best Lagrange multipliers, the framework solves a max-min problem that seeks the optimal multipliers while minimizing the associated Lagrangian dual function. In the proposed deep learning framework, we use a similar technique to integrate the "stability constraints" into the learning process. Within this framework, the LtO model may also employ a Lagrangian approach to incorporate the OPF constraints equation 1b–equation 1h directly into the optimization process, depending on the specific learning scheme adopted (the various LtO schemes adopted are elaborated in the Experimental setting, Section 5).

The augmented Lagrangian loss function incorporates both the prediction error $\mathcal{L}_p$ and penalty terms for constraint violations $\mathcal{L}_c$:

$$\mathcal{L}^{\text{DynOPF}}(\hat{\boldsymbol{y}}, \boldsymbol{y}^\star, \hat{\boldsymbol{x}}(t)) = \mathcal{L}_p(\hat{\boldsymbol{y}}, \boldsymbol{y}^\star) + \mathcal{L}_c(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}(t)), \tag{13}$$

where

$$\mathcal{L}_p(\hat{\boldsymbol{y}}, \boldsymbol{y}^\star) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}^\star\|^2, \tag{14}$$

represents the prediction error (MSE), with respect to the *steady-state* optimal OPF variable $\boldsymbol{y}^\star$ and

$$\mathcal{L}_c(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}(t)) = \sum_{j=1}^{n_{\text{eq}}} \lambda_{h_j} \nu\left(h_j(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}(t))\right) + \sum_{l=1}^{n_{\text{ineq}}} \lambda_{u_l} \nu\left(u_l(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}(t))\right) \tag{15}$$

is a weighted sum of constraint violations incurred by the DynOPF-Net predictions $\hat{\boldsymbol{y}}$, $\hat{\boldsymbol{x}}(t)$. Here ν is a differentiable function which computes the amount of violation of a given constraint (e.g. for a linear inequality constraint $ay \leq b$, the corresponding violation returned by ν is given by $\max(0, ay - b)$), while for a linear equality constraint $ay = b$, the violation returned by ν is $|ay - b|$). Functions $h_j,\ j = 1, \ldots, n_{eq}$ denote the static equality constraints in equation 9b and dynamic constraints equation 9d-equation 9g of Model 2, where constraints equation 9d, equation 9e and equation 9g are written in an implicit form. Functions $u_l,\ l = 1, \ldots, n_{ineq}$ denote the static inequality constraints in equation 9b and the dynamic constraints equation 9h, where the stability constraints are also written in an implicit form. By expressing the generator dynamics and the stability constraints in the same implicit form as the static equality and inequality constraints equation 9b, the system dynamics can be incorporated into the static constraints of model 2, and integrated seamlessly into the optimization process. This unified framework simplifies handling both the static and dynamic requirements of the problem. Importantly, the stability constraints equation 8 are estimated as

$$\hat{\delta}^g(t) - \delta^{\max} \leq 0 \quad \forall g \in \mathcal{G}, \tag{16}$$

which enables, together via equation 13 and equation 15, end-to-end training of the DynOPF-Net model because of the *differentiable* nature of the generator dynamic predictor $\mathscr{N}_\phi^g$. At iteration $k+1$, finding the optimal DynOPF-Net parameters requires solving

$$(\psi, \phi)^{k+1} = \arg\min_{\psi,\phi} \mathbb{E}_{(\boldsymbol{S}^d, \boldsymbol{y}^\star) \sim \mathcal{D}} \left[\mathcal{L}^{\text{DynOPF}} \left(\mathscr{D}_{\psi^k}^{\boldsymbol{\lambda}^k}(\boldsymbol{S}^d), \boldsymbol{y}^\star, \quad \mathscr{N}_{\phi^k}^{\boldsymbol{\lambda}^k}(\hat{\boldsymbol{x}}(0), \hat{V}_g) \right) \right],$$

where $\mathscr{D}_\psi^{\lambda^k}$, $\mathscr{N}_\phi^{\lambda^k}$ denotes the DynOPF-Net proxy optimization model $\mathscr{D}_{\psi^k}$ and the array of NODE models $\mathscr{N}_\phi^g, \forall g \in \mathcal{G}$ at iteration k, with $\boldsymbol{\lambda}^k = [\lambda_{h_1}^k, \ldots, \lambda_{h_{n_{eq}}}^k, \lambda_{u_1}^k, \ldots, \lambda_{u_{n_{ineq}}}^k]^T$. This step is approximated using a stochastic gradient descent method

$$\psi^{k+1} = \psi^k - \eta \nabla_\psi \mathcal{L}^{\text{DynOPF}} \left(\mathscr{D}_{\psi^k}^{\boldsymbol{\lambda}^k}(\boldsymbol{S}^d), \boldsymbol{y}^\star, \hat{\boldsymbol{x}}(t) \right)$$
$$\phi^{k+1} = \phi^k - \eta \nabla_\phi \mathcal{L}^{\text{DynOPF}} \left(\hat{\boldsymbol{y}}, \boldsymbol{y}^\star, \mathscr{N}_{\phi^k}^{\boldsymbol{\lambda}^k}(\hat{\boldsymbol{x}}(0), \hat{V}_g) \right),$$

where η denotes the learning rate and $\nabla_{\psi(\phi)} \mathcal{L}^{\text{DynOPF}}$ represents the gradient of the loss function $\mathcal{L}^{\text{DynOPF}}$ with respect to the parameters $\psi(\phi)$ at the current iteration. Note this step does not retrain $\mathscr{D}_\psi$, $\mathscr{N}_\phi$ from scratch, but uses a hot start for the weights ψ, ϕ. Finally, the Lagrange multipliers are updated as

$$\lambda_{h_j}^{k+1} = \lambda_{h_j}^{k} + \rho\, \nu\left(h_j\left(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}(t)\right)\right) \quad j = 1, \ldots, n_{eq}$$
$$\lambda_{u_l}^{k+1} = \lambda_{u_l}^{k} + \rho\, \nu\left(u_l\left(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}(t)\right)\right) \quad l = 1, \ldots, n_{ineq}\ ,$$

where ρ denotes the Lagrangian step size. The DynOPF-Net training scheme is presented in Algorithm 1. It takes as input the training dataset $\mathcal{D}$, learning rate $\eta > 0$, and step size $\rho > 0$. The Lagrange multipliers $\boldsymbol{\lambda}$ are initialized in line 2. As shown in Figure 1, for epoch k and each sample i, given $\boldsymbol{S}^{d,i}$ (line 4), the DynOPF-Net's proxy optimization model $\mathscr{D}_{\psi^k}$ computes an estimate of the OPF variables $\hat{\boldsymbol{y}}^i$. Given these estimates, for each generator, $g \in \mathcal{G}$, the initial values of the state variables and EMF are computed according to Eq. equation 5, equation 6, equation 7 (line 7). For each generator $g \in \mathcal{G}$, the estimated initial values $\hat{\boldsymbol{x}}^{g,i}(0)$ are input to NODE $\mathscr{N}_{\phi^k}^{g}$, which computes an estimate of the generator state variables $\hat{\boldsymbol{x}}^{g,i}(t)$ (line 8), based on which the violation of each stability constraint (line 9) is computed. The OPF variables' prediction error and the total constraint violations are then used to compute the loss function $\mathcal{L}^{\text{DynOPF}}$ (line 10) equation 13 as specified by equation 14-equation 16, using predicted values $\hat{\boldsymbol{y}}$, $\hat{\boldsymbol{x}}(t)$ and multipliers $\boldsymbol{\lambda}^k$ at the current epoch k. The weights ψ, ϕ of the DynOPF-Net model are then updated using stochastic gradient descent (lines 11). Finally, at the end of the epoch, the multipliers are updated based on the respective constraint violations (line 12).

Algorithm 1 DynOPF-Net for stability-constrained OPF

1: **Input:** Dataset $\mathcal{D} = \{(\boldsymbol{S}^{d,i}, \boldsymbol{y}^{\star,i})\}_{i=1}^{n_{\text{obs}}}$; optimizer method, learning rate η and Lagrangian step size ρ.
2: Initialize Lagrange multipliers $\boldsymbol{\lambda}_h^0 = 0$, $\boldsymbol{\lambda}_u^0 = 0$.
3: **For** each epoch $k = 0, 1, 2, \ldots$
4: **For** each $(\boldsymbol{S}^{d,i}, \boldsymbol{y}^{\star,i}) \in \mathcal{D}$
5: $\hat{\boldsymbol{y}}^i \leftarrow \mathscr{D}_{\psi^k}(\boldsymbol{S}^{d,i})$.
6: **For** each generator $g \in \mathcal{G}$
7: Compute $\hat{\boldsymbol{x}}^{g,i}(0)$ using equation 5-equation 7.
8: $\hat{\boldsymbol{x}}^{g,i}(t) \leftarrow \mathscr{N}_{\phi^k}^{g}(\hat{\boldsymbol{x}}^{g,i}(0), \hat{V}_g^i)$.
9: Compute $\max(0, \hat{\delta}^{g,i}(t) - \delta^{\max})$.
10: $\mathcal{L}^{\text{DynOPF}}(\hat{\boldsymbol{y}}^i, \boldsymbol{y}^{\star,i}, \hat{\boldsymbol{x}}^i(t)) \leftarrow \mathcal{L}_p(\hat{\boldsymbol{y}}^i, \boldsymbol{y}^{\star,i}) + \mathcal{L}_c(\hat{\boldsymbol{y}}^i, \hat{\boldsymbol{x}}^i(t))$.
11: Update DynOPF-Net model $\mathscr{D}_{\psi^k}$, $\mathscr{N}_{\phi^k}$ parameters

$$\psi^{k+1} \leftarrow \psi^k - \eta \nabla_\psi \mathcal{L}^{\text{DynOPF}}(\mathscr{D}_{\psi^k}^{\boldsymbol{\lambda}^k}(\boldsymbol{S}^{d,i}), \boldsymbol{y}^{\star,i}, \hat{\boldsymbol{x}}^i(t))$$
$$\phi^{k+1} \leftarrow \phi^k - \eta \nabla_\phi \mathcal{L}^{\text{DynOPF}}(\hat{\boldsymbol{y}}^i, \boldsymbol{y}^{\star,i}, \mathscr{N}_{\phi^k}^{\boldsymbol{\lambda}^k}(\hat{\boldsymbol{x}}^{g,i}(0), \hat{V}_g^i)).$$

12: Update Lagrange multipliers

$$\lambda_{h_j}^{k+1} \leftarrow \lambda_{h_j}^{k} + \rho\, \nu\left(h_j\left(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}(t)\right)\right) \quad j = 1, \ldots, n_{eq},$$
$$\lambda_{u_l}^{k+1} \leftarrow \lambda_{u_l}^{k} + \rho\, \nu\left(u_l\left(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}(t)\right)\right) \quad l = 1, \ldots, n_{ineq}.$$

5 Experimental Setting

The effectiveness of DynOPF is tested on two power networks of different sizes and complexity: the WSCC 9-bus system, and the IEEE 57-bus system [26]. Our approach is benchmarked against 3 widely adopted LtO methods approximating the AC-OPF problem 1 solutions $\boldsymbol{y} = (\boldsymbol{S}^r, \boldsymbol{V})$ given $\boldsymbol{S}^{\boldsymbol{d}}$. With reference to equation 13, equation 14, equation 15, they use the following loss functions for model training.

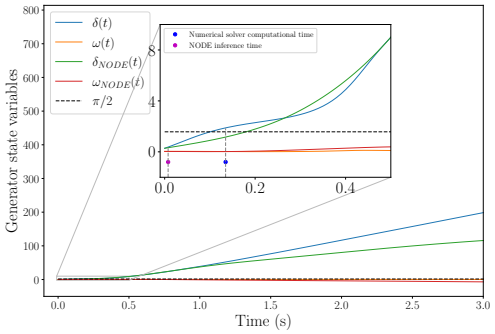
- Zamzam et. al [17] uses a loss function:

$$\begin{cases} \mathcal{L}_p(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}^\star\|^2 \\ \mathcal{L}_c(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}(t)) = 0, \end{cases}$$

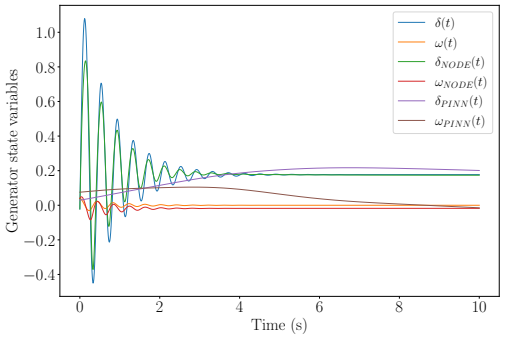
  which minimizes the mean squared error (**MSE**) between the predicted solution and its corresponding label.

- **Lagrangian-Dual** [8] uses a Lagrangian loss function:

$$\begin{cases} \mathcal{L}_p(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}^\star\|^2 \\ \mathcal{L}_c(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}(t)) = \displaystyle\sum_{j=1}^{n'_{eq}} \lambda_{h_j} \nu\left(h_j(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}(t))\right) + \sum_{l=1}^{n'_{ineq}} \lambda_{u_l} \nu\left(u_l(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}(t))\right), \end{cases}$$



(a) Numerical and NODE estimated solutions of system equation 4 in unstable conditions. The NODE model detects instability conditions much faster than a numerical solver and before the physical system transitions into an unstable state.



(b) Numerical, PINN, and NODE estimated solutions of system (4) in stable conditions.

Figure 2: Comparison of solutions in unstable and stable conditions using a numerical ODE solver, NODE, and PINN model.

where n'_{eq}, n'_{ineq} denotes the number of static equality and inequality constraints of model 2, specified by functions $h_j,\ j = 1, \ldots, n'_{eq}$ and $u_l,\ l = 1, \ldots, n'_{ineq}$ equation 9b.

- **DC3** [9] uses the loss function:

$$\begin{cases} \mathcal{L}_p(\boldsymbol{y}, \hat{\boldsymbol{y}}) = f(\hat{\boldsymbol{y}}) \\ \mathcal{L}_c(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}(t)) = \lambda_h \sum_{j=1}^{n'_{eq}} \|h_j(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}(t))\|^2 + \lambda_u \sum_{l=1}^{n'_{ineq}} \max(0, u_l(\hat{\boldsymbol{y}}, \hat{\boldsymbol{x}}(t))^2), \end{cases}$$

where f is objective function equation 9a. This method relies on a completion-correction technique to enforce constraints equation 9b satisfaction of Model 2, in self-supervised training.

The rest of this section describes the training setting.

Practical considerations

since the the generator model equation 4 is known, to obtain accurate estimates of the state variables each neural ODE model can be hot-started and then integrated within the DynOPF-Net model as detailed in Section 4.2.

Dataset creation and training setting Based on the motivations above, each NODE model $\mathscr{N}_\phi^g$ is trained in a supervised fashion as described in Section 4.1. For each generator $g \in \mathcal{G}$, the dataset $\mathcal{D}^g$ used for training $\mathscr{N}_\phi^g$ consists of distinct time series, each representing the solution of equation 4 given a different instance of the generator model. Some operational set points yield stable conditions, while others are unstable. The NODE models are trained on a dataset $\mathcal{D}^g = \{\big((\boldsymbol{x}^{g,i}(0), V_g^i), (\boldsymbol{x}^{g,i}(t), V_g^i)\big)\}_{i=1}^{10,000}$, where $(\boldsymbol{x}^g(0), V_g)$ represents the input to the dynamic predictor $\mathscr{N}_\phi^g$ as discussed in 4.1, while $\boldsymbol{x}^g(t) = (\delta^g(t), \omega^g(t), V_g)$ is the corresponding target and solution of the augmented generator model (12). For each $\boldsymbol{x}^g(0), V_g$, each OPF variable S_g^r, V_g is sampled from a uniform distribution in which lower and upper bounds are given by the corresponding operational limits equation 1b, equation 1c and equation 1d, equation 1e. Note that this sampling scheme spans the full range of *feasible* OPF variables; since the NODEs' inputs are provided by $\mathscr{D}_{\psi^k}$; this approach is robust with respect to its small prediction error, as verified in practice. Given the values of $|V_g|, \theta_g$, the initial state variables values $\delta^g(0)$ and $\omega^g(0)$ are obtained from equation 5-equation 7. The parameters of generator model equation 4, such as the damping coefficient d^g and inertia constant m^g, are adopted from [27]. Given initial condition $\boldsymbol{x}^g(0)$, the corresponding solution of equation 4 $\boldsymbol{x}^g(t)$ is generated by adopting a numerical ODE algorithm named $Dopri5$ [28], an adaptive Runge-Kutta method of order 5. The simulation time is set to 3 seconds (s) and the initial Δ_t is set at 0.001 (s). Each dataset $\mathcal{D}^g$ is divided into training, validation, and test sets with a $80/10/10$ split.

On both the IEEE 57 and WSCC 9-bus systems, DynOPF-Net and each baseline LtO model is trained on a dataset $\mathcal{D} = \{\big(\boldsymbol{S}^{d,i}, \boldsymbol{y}^{\star,i}\big)\}_{i=1}^{10,000}$. Each input $\boldsymbol{S}^d$ represents a load vector, while the corresponding target is the associated

optimal generator set-points $\boldsymbol{y}^\star = (\boldsymbol{S^r}, \boldsymbol{V})$ of Model 2, with the assumption that the generators are in steady-state. Similarly to [8], the load demands vector S_i^d is generated by applying to each nominal load $S_i^d,\ i \in \mathcal{N}$, a uniform random perturbation which results in load demands set to $\pm 20\%\ S_i^d$. The dataset only reports feasible AC-OPF instances. Thus, each training data point represents a valid snapshot of the AC-OPF problem, consisting of a load profile along with the corresponding optimal generator set points. The AC-OPF models are implemented with MATPOWER and solved with IPOPT, a numerical implementation of the interior point method [29]. For each test case, the dataset $\mathcal{D}$ is divided into training, validation, and test sets on a $80/10/10$ split.

Table 1: Average and standard deviation of computational times for numerical solvers vs. neural ODE inference times

Method	Numerical Solver	NODE
Dopri5 (default)	0.135 ± 0.015 (sec)	$\mathbf{0.008 \pm 0}$ (sec)
Bosh3	0.446 ± 0.039 (sec)	$\mathbf{0.017 \pm 0}$ (sec)

Table 2: Average and standard deviation of MSE, constraint violations, and steady-state gap for the WSCC 9-bus system across different approaches based on 40 trials.

Method	Steady-State Metrics					Dynamic Metrics		
	MSE ($\boldsymbol{p^r}$) $\times 10^{-3}$	MSE ($\boldsymbol{q^r}$) $\times 10^{-3}$	MSE ($\lvert\boldsymbol{V}\rvert$) $\times 10^{-4}$	MSE ($\boldsymbol{\theta}$) $\times 10^{-4}$	Optimality Gap % $\times 10^{-1}$	Flow Vio. equation 1f $\times 10^{-3}$	Boundary Vio. equation 1b-equation 1e $\times 10^{-4}$	**Stability Vio.** equation 9h $\times \mathbf{10^1}$
MSE	1.90 ± 0.272	1.65 ± 1.018	0.32 ± 0.153	0.43 ± 0.149	1.29 ± 0.008	10.45 ± 2.183	9.72 ± 4.930	$\mathbf{2.26 \pm 0.189}$
DC3	1.86 ± 0.217	1.56 ± 0.326	0.26 ± 0.195	0.48 ± 0.343	1.17 ± 0.006	0.00	0.00	$\mathbf{2.45 \pm 0.205}$
LD	1.77 ± 0.163	1.72 ± 0.248	0.16 ± 0.099	0.55 ± 0.051	1.12 ± 0.008	7.19 ± 0.425	0.00	$\mathbf{2.13 \pm 0.175}$
DynOPF-Net	2.41 ± 0.253	3.18 ± 1.273	2.55 ± 0.354	3.82 ± 0.924	1.32 ± 0.032	8.32 ± 0.596	0.41 ± 0.243	**0.00**

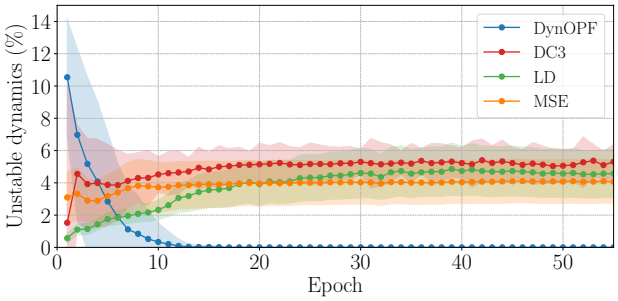
Table 3: Average and standard deviation of MSE, constraint violations, and steady-state gap for the IEEE 57-bus system across different approaches based on 40 trials.

Method	Steady-State Metrics					Dynamic Metrics		
	MSE ($\boldsymbol{p^r}$) $\times 10^{-3}$	MSE ($\boldsymbol{q^r}$) $\times 10^{-3}$	MSE ($\lvert\boldsymbol{V}\rvert$) $\times 10^{-3}$	MSE ($\boldsymbol{\theta}$) $\times 10^{-3}$	Optimality Gap % $\times 10^{-1}$	Flow Vio. equation 1f $\times 10^{-3}$	Boundary Vio. equation 1b-equation 1e $\times 10^{-4}$	**Stability Vio.** equation 9h $\times \mathbf{10^1}$
MSE	3.48 ± 0.321	3.86 ± 1.512	0.77 ± 0.148	1.42 ± 0.237	1.66 ± 0.243	12.65 ± 2.281	6.44 ± 1.434	$\mathbf{2.33 \pm 0.206}$
DC3	3.31 ± 0.579	6.74 ± 0.580	0.51 ± 0.078	0.64 ± 0.081	1.64 ± 0.049	0.00	0.00	$\mathbf{2.86 \pm 0.232}$
LD	3.97 ± 0.279	3.52 ± 2.427	0.34 ± 0.012	0.95 ± 0.054	1.68 ± 0.125	6.23 ± 0.125	0.00	$\mathbf{2.31 \pm 0.219}$
DynOPF-Net	5.05 ± 0.175	7.42 ± 1.482	2.99 ± 0.214	4.43 ± 0.673	2.26 ± 0.180	9.15 ± 0.442	0.25 ± 0.172	**0.00**

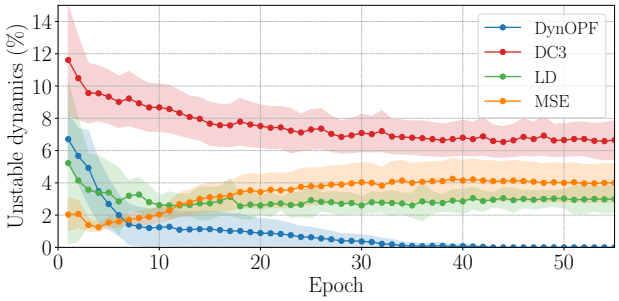
6 Experimental Results

This section presents the results of each benchmark system of DynOPF-Net and each baseline LtO method. The experiments focus on two main aspects: **(1)** assessing the effectiveness of NODEs and PINNs in capturing generator dynamics and comparing their precision and computational efficiency to a numerical ODE solver; **(2)** comparing DynOPF-Net with LtO methods that capture only the steady-state AC-OPF problem, focusing on the stability constraint violations. Unless otherwise stated, results are reported as an average of 40 independent runs on a subset of dataset $\mathcal{D}$ where DynOPF-Net and each LtO model is *not* trained on, which we refer to as the test set. Specifically, we report:

- The inference time (measured in seconds) of NODEs which we compare to the computational time of a traditional numerical ODE solver and the precision of state variables' estimate (measured as the percentage relative ℓ^2 error) of NODEs and PINNs, a different learning based approach discussed in Section 6.1.
- The average constraint violations (measured as $\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} |h_j(\hat{\boldsymbol{y}}^i, \hat{\boldsymbol{x}}^i(t))|$ for the j-th equality and $\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} \max(0, u_l(\hat{\boldsymbol{y}}^i, \hat{\boldsymbol{x}}^i(t)))$ for the l-th inequality, where n_{test} is the test set size), incurred by DynOPF-Net and each baseline method approximation of $\boldsymbol{y}$.
- The average percentage *steady-state* gap incurred by DynOPF-Net and the baselines LtO predictions $\hat{\boldsymbol{y}}$, and measured as $\frac{|f(\boldsymbol{y}^\star(\boldsymbol{S}^d)) - f(\hat{\boldsymbol{y}}(\boldsymbol{S}^d))|}{|f(\boldsymbol{y}^\star(\boldsymbol{S}^d))|} * 100$, where f is objective equation 9a and prediction error (MSE) $\|\boldsymbol{y}^\star - \hat{\boldsymbol{y}}\|^2$ of the OPF variables with respect to optimal (steady-state) generators set-points values. The steady-state assumption is crucial for evaluating how closely each solution approximates the AC-OPF optimal results, though it *does not reflect the stability-constrained AC-OPF problem results* which is the main focus of this paper. Given the non-linearity of both the dynamics and optimization in the stability-constrained AC-OPF, computing exact optimal decisions $\boldsymbol{y}^\star$ with traditional methods is unfeasible. Consequently, while our method may achieve slightly higher steady-state gaps or prediction errors, *these should not be interpreted in the context of the stability-constrained AC-OPF problem* and the key desiderata and goal is that of producing solutions with low stability violations.



(a) WSCC 9-bus system - Percentage of unstable solutions at training time for different methods.



(b) IEEE 57-bus system - Percentage of unstable solutions at training time for different methods.

- The inference time (measured in seconds) of DynOPF-Net and each LtO model to generate $\hat{\boldsymbol{y}}$.

6.1 Dynamic Forecasting

Runtime comparison between NODEs and a traditional ODE solver Here the goal is to evaluate the NODEs' inference time to produce the generator state variables' estimates and to compare it with the computational time of a traditional ODE solver. Given the synchronous generator model described by (4), a numerical ODE solver could be adopted to determine the evolution in time of the state variables $\delta^g(t)$ and $\omega^g(t)$. However, in case of unstable conditions, the system dynamics can be as rapid as, or even surpass, the time required for computing the ODE solution with a numerical solver. This situation is depicted in Figure 2a where unstable conditions arise before a numerical solution to the system of differential equations equation 4 is computed. Conversely, the neural ODE model $\mathscr{N}_\phi^g$ is capable of detecting unstable conditions before the system transitions into an unstable state, while also providing a good approximation of the solution. As anticipated in Section 4.1, this speed advantage arises from the vector field approximation of (4), enabling quicker computation of update steps in a numerical ODE solver. Table 1 reports the average and standard deviation of computational time, for numerical solvers, and inference time, for neural ODEs, given 2 different numerical algorithms. *Neural ODE models are, on average, about* 20 *times faster than a numerical solver which uses the governing equations of equation 4. This aspect makes neural ODE models natural candidates as dynamic predictors for the generator model in real-time applications.*

Comparison between NODEs and PINNs. Here the goal is to assess the precision of the NODEs' estimate of the generator state variables and to compare them with PINNs [20]. PINNs are ML-based models that incorporates known physical laws into the learning process. Instead of relying solely on data, PINNs use physics-based constraints to guide the training, ensuring that the model's predictions are consistent with the underlying scientific principles. Figure 2b shows the NODE and PINNs' state variables estimates in case of stable conditions. While a NODE model produces highly accurate state variables' predictions, a PINN model trained on the same dataset $\mathcal{D}^g$ but affected by a generalization bias, is incapable of capturing the generator dynamics across different instances of equation 4 and produces a poor state variables' estimate. Specifically, the percentage ℓ^2 error between the numerical ODE solver solutions $\delta(t), \omega(t)$ and the NODE predictions $\delta_{\text{NODE}}(t), \omega_{\text{NODE}}(t)$ is 5.17%, while for the PINN predictions $\delta_{\text{PINN}}(t), \omega_{\text{PINN}}(t)$ is significantly higher at 69.45%.

6.2 Steady-state Prediction Errors and Constraint Violations

Next, we investigate constraint violations, with a focus on the stability constraint violations equation 9h and how they relate with *steady-state* prediction errors. Tables 2 and 3 report the average and standard deviation of predicted set-point errors (MSE), constraint violations and steady-state gap (which will be discussed in the next subsection) on the test set, for each method and benchmark system. First note that, for each test case, the tables report comparable prediction errors and static constraint violations equation 9b, across different methods. DC3 achieves no static (flow and boundary) constraint violations, being designed to ensure feasibility during training. However, all baseline methods systematically fail to satisfy stability requirements equation 9h. By integrating the generator dynamics within training, DynOPF-Net learns to meet the stability requirements in the first stage of training, as seen in Figures 3a and 3b, achieving compliance with the dynamic requirements around epochs 20 and 50, respectively. Test results in Tables 2 and 3 show the stability constraint violations align with training.

Figures 3a and 3b show the percentage of solutions violating the stability constraints in the first 50 epochs of training. Figure 3a, pertaining to the WSCC 9-bus system, shows that DynOPF-Net learns rapidly to meet the dynamic constraints, which violations approach zero level after epoch 10 of training. In contrast, all the baseline methods produce between 4% and 6% of unstable solutions throughout training. Figure 3b shows a similar scenario for the IEEE 57-bus system. DynOPF-Net learns to address the dynamic requirements and achieves no stability constraint violations at training time after epoch 40. Conversely, the baseline methods produce between 4% and 7% of unstable solutions during training. The convergence rate of DynOPF-Net is slightly slower than on the WSCC-9 bus system, likely due to the complexity of the test case. *These results provide strong empirical evidence of the importance to integrate dynamic requirements into the AC-OPF problem.* Our approach learns to modify potentially unstable set points, at a cost of only slightly higher steady-state MSE than the baseline approaches. Note in particular the MSE of $|\boldsymbol{V}|$ and $\boldsymbol{\theta}$; these variables directly affect the generator dynamics in equation 4, and thus their modification is necessary to ensure stability constraints satisfaction. In contrast, all the baseline methods achieve slightly smaller steady-state prediction error than DynOPF-Net, as they solve the steady-state AC-OPF problem 1 ignoring the generator dynamics, and produce significant violations of the stability constraints. *These results indicate that the OPF voltage setpoints have high impact on the generator dynamics.*

6.3 Steady-state Gaps

This section discusses the suboptimality of the estimated solution $\hat{\boldsymbol{y}}$ with respect to the ground truth $\boldsymbol{y}^\star$, given loads $\boldsymbol{S}^d$ and measured in terms of objective value equation 9a, of DynOPN-Net and each baseline method. Note that the steady-state gap is measured with respect to the optimal solution $\boldsymbol{y}^\star$ of the stability-constrained AC-OPF Model 2, with the assumptions that the generators are in steady-state conditions, and thus does not measure the stability-constrained AC-OPF optimality gap, which is unattainable in the setting considered. Nonetheless, it provides valuable insights on the decision quality of each LtO method and DynOPF-Net for solving Problem 2. Tables 2 and 3 report the steady-state gaps on the WSCC-9 and IEEE-57 bus systems, respectively. The tables report that all methods achieve comparable gaps in each test case. This is intuitive, since objective equation 9a depends solely on the power generated $\boldsymbol{p}^r$, and all methods produce similar $\boldsymbol{p}^r$'s prediction error, as displayed in Tables 2 and 3. DynOPF-Net produces average percentage steady-state gaps of $1.32 \times 10^{-1}\%$ and $2.26 \times 10^{-1}\%$ for the WSCC 9 and the IEEE 57-bus system while preserving system stability, that are comparable with the best gaps - LD with $1.12 \times 10^{-1}\%$ and DC3 with $1.64 \times 10^{-1}\%$ - which often violates stability constraints. The higher objective costs observed with DynOPF-Net is intuitively attributed to a restricted feasible space due to the integration of generator stability constraints equation 9h within the AC OPF model 1.

Table 4: Average and standard deviation of inference times for different OPF learning approaches

Method	WSCC 9-bus	IEEE 57-bus
DynOPF-Net	0.001 ± 0.00 (sec)	0.009 ± 0.00 (sec)
DC3	0.025 ± 0.00 (sec)	0.089 ± 0.00 (sec)
LD	0.000 ± 0.00 (sec)	0.001 ± 0.00 (sec)
MSE	0.000 ± 0.00 (sec)	0.001 ± 0.00 (sec)

6.4 Inference Time

Finally, we evaluate the average inference time of DynOPF-Net and each baseline LtO model. Table 4 shows the inference time of each LtO method on each test case. On average, DynOPF produces near-optimal and stable solutions in 1 (ms) and 9 (ms) for the WSCC-9 bus and IEEE 57-bus, respectively, which is slightly slower than the MSE and LD approaches, and about $15\times$ faster than the DC3 method. These results suggest that there is also a tradeoff between compliance with dynamic requirements and inference time. DC3 achieves the highest inference time, due to its correction and completion procedure, which requires solving a nonlinear system of equations and the Jacobian matrix computation. While DynOPF-Net is already applicable for real-time applications, its efficiency could be improved by computing the state variables in parallel, since each dynamic predictor is independent. *This aspect makes DynOPF-Net's inference time independent from the size of the power network, suggesting potential for large-scale systems.*

7 Conclusion

This paper was motivated by the need of developing a fast surrogate AC-OPF solver which takes into account the dynamic nature of a power network. While an increasing effort has been devoted to developing surrogate optimization models for AC-OPF, the paper shows their inability to deal with the system dynamics. We proposed DynOPF-Net, a novel integration of Learning to Optimize neural ODEs that successfully integrates synchronous generator dynamics

within model training. To integrate the generator dynamics within the optimization, we employed a dual network architecture, consisting of a LtO model for estimating the OPF variables and neural ODE models to infer the dynamic behavior of each generator. Importantly, the proposed model is fully differentiable and allows for end-to-end training. DynOPF-Net is compared with state-of-the-art LtO methods that have been proposed to solve the steady-state AC-OPF problem. Empirical results report that all the baseline Learning to Optimize methods systematically fail to ensure system stability. The findings pertaining to feasibility, optimality, and inference time consistently demonstrate that DynOPF-Net is capable of producing stable solutions, while achieving low steady-state gaps and minimal prediction errors, for real-time applications. Future work will focus on testing DynOPF-Net on larger power systems and will investigate the correlation between the prediction error of the OPF variables and violations of the stability constraints. Another line of work will focus on developing methods to ensure compliance with the system dynamics together with static AC-OPF constraint satisfaction.

Acknowledgments

This research was partially supported by NSF awards EPCN-2242931, CAREER-2143706, and NSF awards 2041835, 2242930. Its view and conclusions are those of the authors only.

References

[1] Kyri Baker. Solutions of DC OPF are Never AC Feasible. In *Proc. of the Twelfth ACM Intl. Conference on Future Energy Systems*, 2021.

[2] Li Yao, Xiuli Wang, Yujun Li, Chao Duan, and Xiong Wu. Distributionally Robust Chance-Constrained AC-OPF for Integrating Wind Energy Through Multi-Terminal VSC-HVDC. *IEEE Transactions on Sustainable Energy*, 11(3):1414–1426, 2020.

[3] N. Mlilo, J. Brown, and T. Ahfock. Impact of intermittent renewable energy generation penetration on the power system networks – a review. *Technol Econ Smart Grids Sustain Energy*, 6:25, 2021.

[4] Changhong Zhao, Ufuk Topcu, Na Li, and Steven Low. Design and stability of load-side primary frequency control in power systems. *IEEE Transactions on Automatic Control*, 59(5):1177–1189, 2014.

[5] Nikos Hatziargyriou et al. Definition and classification of power system stability – revisited & extended. *IEEE Transactions on Power Systems*, 36(4):3271–3281, 2021.

[6] Min Zhou, Minghua Chen, and Steven H. Low. DeepOPF-FT: One Deep Neural Network for Multiple AC-OPF Problems With Flexible Topology. *IEEE Transactions on Power Systems*, 38(1):964–967, 2023.

[7] James Kotary, Ferdinando Fioretto, Pascal Van Hentenryck, and Bryan Wilder. End-to-end constrained optimization learning: A survey. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4475–4482, 2021.

[8] Ferdinando Fioretto, Terrence W.K. Mak, and Pascal Van Hentenryck. Predicting AC Optimal Power Flows: Combining Deep Learning and Lagrangian Dual Methods. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(01):630–637, Apr. 2020.

[9] Priya L Donti, David Rolnick, and J Zico Kolter. Dc3: A learning method for optimization with hard constraints. In *ICLR*, 2020.

[10] Patrick Kidger. On neural differential equations, 2022.

[11] Yeesian Ng, Sidhant Misra, Line Roald, and Scott Backhaus. Statistical learning for DC optimal power flow. In *Power Systems Computation Conference*, 2018.

[12] Manish K. Singh, Vassilis Kekatos, and Georgios B. Giannakis. Learning to solve the ac-opf using sensitivity-informed deep neural networks. *IEEE Transactions on Power Systems*, 37(4):2833–2846, 2022.

[13] Deepjyoti Deka and Sidhant Misra. Learning for dc-opf: Classifying active sets using neural nets. *2019 IEEE Milan PowerTech*, pages 1–6, 2019.

[14] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[15] Mostafa Mohammadian, Kyri Baker, My H. Dinh, and Ferdinando Fioretto. Learning solutions for intertemporal power systems optimization with recurrent neural networks. In *2022 17th International Conference on Probabilistic Methods Applied to Power Systems (PMAPS)*, pages 1–6, 2022.

[16] Minas Chatzos, Terrence W. K. Mak, and Pascal Van Hentenryck. Spatial Network Decomposition for Fast and Scalable AC-OPF Learning, 2021.

[17] Ahmed Zamzam and Kyri Baker. Learning optimal solutions for extremely fast AC optimal power flow. In *IEEE SmartGridComm*, Dec. 2020.

[18] Qinggang Su, Habib Ullah Khan, Imran Khan, Bong Jun Choi, Falin Wu, and Ayman A. Aly. An optimized algorithm for optimal power flow based on deep learning. *Energy Reports*, 7:2113–2124, 2021.

[19] 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*, 378:686–707, Feb 2019.

[20] George S. Misyris, Andreas Venzke, and Spyros Chatzivasileiadis. Physics-informed neural networks for power systems. *2020 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5, 2019.

[21] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 26548–26560. Curran Associates, Inc., 2021.

[22] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24:1–97, 2023.

[23] Tannan Xiao, Ying Chen, Shaowei Huang, Tirui He, and Huizhe Guan. Feasibility study of neural ode and dae modules for power system dynamic component modeling. *IEEE Transactions on Power Systems*, 38(3):2666–2678, 2023.

[24] Mohammadhafez Bazrafshan, Nikolaos Gatsis, Ahmad F. Taha, and Josh A. Taylor. Augmenting the optimal power flow for stability. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 4104–4109, 2016.

[25] Peter W. Sauer and M. A. Pai. *Power System Dynamics and Stability*. Prentice Hall, Upper Saddle River, N.J., 1998.

[26] Sogol Babaeinejadsarookolaee et al. The Power Grid Library for Benchmarking AC Optimal Power Flow Algorithms, 2021.

[27] Peijie Li, Junjian Qi, Jianhui Wang, Hua Wei, Xiaoqing Bai, and Feng Qiu. An SQP Method Combined with Gradient Sampling for Small-Signal Stability Constrained OPF. *IEEE Transactions on Power Systems*, 32, 07 2016.

[28] Liu Liu, Felix Felgner, and Georg Frey. Comparison of 4 numerical solvers for stiff and hybrid systems simulation. In *2010 IEEE 15th Conference on Emerging Technologies & Factory Automation*, 2010.

[29] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.