Learning to Pursue AC Optimal Power Flow Solutions with Feasibility Guarantees

Damola Ajeyemi, Yiting Chen, Antonin Colot, Jorge Cortés, and Emiliano Dall'Anese

***Abstract*—This paper focuses on an AC optimal power flow (OPF) problem for distribution feeders equipped with controllable distributed energy resources (DERs). We consider a solution method that is based on a continuous approximation of the projected gradient flow – referred to as the safe gradient flow – that incorporates voltage and current information obtained either through real-time measurements or power flow computations. These two setups enable both online and offline implementations. The safe gradient flow involves the solution of convex quadratic programs (QPs). To enhance computational efficiency, we propose a novel framework that employs a neural network approximation of the optimal solution map of the QP. The resulting method has two key features: (a) it ensures that the DERs' setpoints are practically feasible, even for an online implementation or when an offline algorithm has an early termination; (b) it ensures convergence to a neighborhood of a strict local optimizer of the AC OPF. The proposed method is tested on a 93-node distribution system with realistic loads and renewable generation. The test shows that our method successfully regulates voltages within limits during periods with high renewable generation.**

I. Introduction

This work considers power distribution systems with controllable distributed energy resources (DERs), and aims to advance real-time control strategies and computational methodologies in this domain. The focus is on the AC optimal power flow (OPF) problem [1] and, in particular, on its real-time implementation. These include recent frameworks that leverage feedback-based implementations [2]–[5] or low-latency batch solutions [6], [7]. These real-time implementations seek to generate setpoints at a time scale that is consistent with the variability of uncontrollable loads and power available from renewable sources [8].

Prior work. Feedback-based online algorithms have been explored in the context of AC OPF for distribution systems [2]–[5]. Shifting from a feedback optimization paradigm to feedforward optimization, a substantial body of work has explored the use of neural networks and deep learning techniques to approximate solutions to the AC OPF problem; see, for example, [7], [9]–[17], the generative model in [18], and the foundation models in [19]. While these methods primarily target AC OPF tasks in transmission networks, some of them can be adapted to distribution grids as well. This body of literature has adopted various approaches: some aim to directly predict an optimal solution to the AC OPF problem [7], while others focus on predicting a Karush–Kuhn–Tucker (KKT) point [14].

In general, these methods lack formal guarantees in terms of generating optimal solutions of the AC OPF, not to mention feasible points (as we will show in our numerical results). Once a candidate solution is generated by the neural network, recovering a valid operating point that satisfies all AC OPF constraints can be computationally demanding – offsetting the speed advantages offered by the neural network approximation; heuristics may be used, but still lack formal guarantees. Post-processing for neural networks approximating solutions to problems with linear constraints are available in the literature [20], but they are not applicable to the AC OPF. The AC OPF is nonconvex and may admit multiple globally and locally optimal solutions; this means that the function that maps loads in the network and parameters of the problem into optimal solutions is a set-valued mapping. Such a set-valued mapping cannot be approximated with the single-valued mapping of a neural network; see the discussion in, e.g. [21] and [12], [22]. One workaround suggested in [21] is when the number of solutions (or KKT points) of the AC OPF is finite and they can all be identified; in this case, the neural network can be trained to output the vector enumerating the optimal solutions (or KKT points). Enumerating the solutions (or KKT points) of the AC OPF is computationally infeasible [22].

An alternative strategy involves replacing algorithmic updates in traditional optimization methods – such as Newton-type or gradient-based methods – with neural networks [16], [17]. These methods offer computational advantages, but existing works do not offer convergence and feasibility guarantees.

Contributions. In this paper, we consider a solution method for the AC OPF that is based on a continuous approximation of the projected gradient flow – hereafter referred to as the safe gradient flow [23], [24] – incorporating voltage and current information obtained either through real-time measurements (as in feedback optimization [2]–[5]) or power flow computations. To favor computational efficiency and speed, we propose a novel framework that employs a neural network approximation of the safe gradient flow. In particular, the neural network predicts the unique optimal solution of a quadratic program (QP) defining the map of the safe gradient flow. The learning task is well-posed, in the sense that the optimal solution map of the QP is a single-valued function and it is continuous. The

This work was supported in part by the NSF awards 2444163 and 1947050.

Damola Ajeyemi is with the Division of Systems Engineering, Boston University, Boston, MA 02215, USA (email: dajeyemi@bu.edu).

Yiting Chen is with the Department of Electrical and Computer Engineering, Boston University, Boston, MA 02215, USA (email: yich4684@bu.edu).

Antonin Colot is with the University of Liège, B-4000 Liège, Belgium (email: antonin.colot@eliagrid-int.com).

Jorge Cortés is with the Department of Mechanical and Aerospace Engineering, University of California San Diego, CA 92093 San Diego, USA (email: cortes@ucsd.edu).

Emilliano Dall'Anese is with the Department of Electrical and Computer Engineering and the Division of Systems Engineering, Boston University, Boston, MA 02215, USA (email: edallane@bu.edu).

learned safe gradient flow is then used in conjunction with voltage and current information to identify AC OPF solutions. We summarize our contributions as follows:

(c1) We propose an iterative method where the neural network approximation of the safe gradient flow is used with either real-time measurements or power flow computations.

(c2) We show that our method leads to solutions that are practically feasible. The term practical feasibility refers to the fact that we provide guarantees on the maximum constraint violation (which is found to be negligible through our numerical experiments); the analytical estimate of the violation allows for a careful tightening of the constraints in the AC OPF so that the neural network can be trained not to violate the actual constraints. The practical feasibility is at *any time*, in the sense that the algorithm produces feasible points even when terminated before convergence or implemented online.

(c3) We show that the proposed learning-based method converges exponentially fast within a neighborhood of KKT points of the AC OPF that are strict local optimizers.

(c4) We perform numerical experiments on a 93-bus distribution system [25] and with realistic load and solar production profiles from the Open Power System Data. We show that our approach ensures voltage regulation and satisfaction of the DERs' constraints. Our method shows far superior performance in terms of voltage regulation compared to approaches that attempt to approximate the solutions of the OPF directly.

The remainder of the paper is organized as follows. Section II will formulate the AC OPF and will explain our proposed mathematical model. Section III will provide details on the neural network-based safe gradient flow, while Section IV will illustrate simulation results. Section V will present our theoretical results, and Section VI will conclude the paper.

II. Problem Formulation and Proposed Model

A. Distribution System Model

We consider a distribution system[1] comprising $N+1$ nodes, labeled by $\{0, 1, \ldots, N\}$. Node 0 represents the substation (or point of common coupling), whereas $\mathcal{N} := \{1, \ldots, N\}$ contains the remaining nodes; these nodes may feature a mix of uncontrollable loads and controllable DERs. We focus on a steady-state representation in which currents and voltages are modeled as complex phasors. For each node $k \in \mathcal{N}$, let the line-to-ground voltage phasor be $v_k = \nu_k e^{j\,\delta_k} \in \mathbb{C}$, with magnitude $\nu_k = |v_k|$ and angle δ_k. The phasorial representation of the current injected at node k is $i_k = |i_k| e^{j\,\psi_k} \in \mathbb{C}$. At the substation, the voltage is denoted as $v_0 = V_0 e^{j\,\delta_0}$ [26].

As usual, applying Ohm's and Kirchhoff's Laws in the phasor domain yields the relationship

$$\begin{bmatrix} i_0 \\ \boldsymbol{i} \end{bmatrix} \;=\; \begin{bmatrix} y_0 & \bar{\boldsymbol{y}}^\top \\ \bar{\boldsymbol{y}} & \boldsymbol{Y} \end{bmatrix} \begin{bmatrix} v_0 \\ \boldsymbol{v} \end{bmatrix}, \tag{1}$$

where $\boldsymbol{i} = [\, i_1, \ldots, i_N \,]^\mathsf{T} \in \mathbb{C}^N$ and $\boldsymbol{v} = [\, v_1, \ldots, v_N \,]^\mathsf{T} \in \mathbb{C}^N$, and where the admittance matrix $\boldsymbol{Y} \in \mathbb{C}^{N \times N}$ and the vectors $\bar{\boldsymbol{y}} \in \mathbb{C}^N, y_0 \in \mathbb{C}$ are built based on the series and shunt parameters of the lines under a Π-model [26].

Suppose that there are G DERs in the network, each capable of generating or consuming active and reactive powers. Let $\boldsymbol{u} = [\, p_1, \ldots, p_G,\; q_1, \ldots, q_G \,]^\mathsf{T} \in \mathbb{R}^{2G}$ collect the DERs' active powers p_i and reactive powers q_i. For each DER $i \in \mathcal{G}$, the set of admissible active and reactive power setpoints is defined by a compact set $\mathcal{C}_i \subset \mathbb{R}^2$; the overall control domain is given by the Cartesian product $\mathcal{C} := \mathcal{C}_1 \times \mathcal{C}_2 \times \cdots \times \mathcal{C}_G \subset \mathbb{R}^{2G}$. Define the mapping $m : \{1, \ldots, G\} \to \mathcal{N}$ to indicate the node at which each DER is connected. Then, the net injections at node n can be written as $p_{\text{net},n} = \sum_{i \in \mathcal{G}_n} p_i \;-\; p_{\ell,n}$, $q_{\text{net},n} = \sum_{i \in \mathcal{G}_n} q_i \;-\; q_{\ell,n}$, where $\mathcal{G}_n = \{\, i \in \{1, \ldots, G\} : m(i) = n\}$ and with $p_{\ell,n}, q_{\ell,n}$ denoting the real and reactive loads (positive entries imply consumption). Let $\boldsymbol{p} \in \mathbb{R}^N$ and $\boldsymbol{q} \in \mathbb{R}^N$ collect the active and reactive powers from the DERs on nodes $n \in \mathcal{N}$. Then, from (1), one can derive the equation:

$$(\boldsymbol{p} - \boldsymbol{p}_l) + j(\boldsymbol{q} - \boldsymbol{q}_l) \;=\; \operatorname{diag}(\boldsymbol{v}) \left(\bar{\boldsymbol{y}}^* v_0^* \;+\; \boldsymbol{Y}^* \boldsymbol{v}^* \right), \tag{2}$$

where $\boldsymbol{s}_l := \boldsymbol{p}_l + j\boldsymbol{q}_l \in \mathbb{C}^N$ is a vector collecting the aggregate complex powers of non-controllable loads at each node. Finally, we consider a set of nodes $\mathcal{M} \subseteq \mathcal{N}$ where voltages are monitored, and let $M = |\mathcal{M}|$ denote the number of such nodes.

Given the controllable powers $\boldsymbol{u}$ and the loads $\boldsymbol{p}_l, \boldsymbol{q}_l$, one can employ numerical techniques to solve (2) for the voltages $\boldsymbol{v}$. It is important to note that the power flow equation (2) may admit zero, one, or multiple solutions [27]–[29]. If multiple solutions exist, we focus on *practical* solutions; i.e., the solution within the neighborhood of the nominal voltage profile that yields relatively high voltage magnitudes and low line currents. Due to the Implicit Function Theorem, we can define a map $(\boldsymbol{u}, \boldsymbol{s}_l) \mapsto \boldsymbol{v}(\boldsymbol{u}, \boldsymbol{s}_l)$ mapping loads and power from the DERs into complex voltages at the nodes. Additionally, based on the function $\boldsymbol{v}(\boldsymbol{u}, \boldsymbol{s}_l)$ and the topology of the network, we also define the function $(\boldsymbol{u}, \boldsymbol{s}_l) \mapsto \boldsymbol{i}(\boldsymbol{u}, \boldsymbol{s}_l)$ mapping loads and power from the DERs into line currents; we assume that we monitor a set $\mathcal{E}$ of L lines.

The functions $\boldsymbol{v}(\boldsymbol{u}, \boldsymbol{s}_l)$ and $\boldsymbol{i}(\boldsymbol{u}, \boldsymbol{s}_l)$ are utilized to formulate instances of the AC OPF. In the remainder of this paper, we proceed under the following practical assumption.

Assumption 1 (*Maps in a neighborhood of the nominal voltage profile*)**.** The functions $(\boldsymbol{u}, \boldsymbol{s}_l) \mapsto |\boldsymbol{v}(\boldsymbol{u}, \boldsymbol{s}_l)|$ and $(\boldsymbol{u}, \boldsymbol{s}_l) \mapsto |\boldsymbol{i}(\boldsymbol{u}, \boldsymbol{s}_l)|$ are unique and continuously differentiable in an open neighborhood of the nominal voltage profile. Additionally, their Jacobian matrices $\boldsymbol{J}_v(\boldsymbol{u}, \boldsymbol{s}_l) := \frac{\partial |\boldsymbol{v}(\boldsymbol{u}, \boldsymbol{s}_l)|}{\partial \boldsymbol{u}}$ and $\boldsymbol{J}_i(\boldsymbol{u}, \boldsymbol{s}_l) := \frac{\partial |\boldsymbol{i}(\boldsymbol{u}, \boldsymbol{s}_l)|}{\partial \boldsymbol{u}}$ are locally Lipschitz continuous over that neighborhood. □

[1] *Notation*. We use the following notational conventions throughout the paper. Boldface upper-case letters (e.g., $\mathbf{X}$) denote matrices, and boldface lower-case letters (e.g., $\mathbf{x}$) denote column vectors. The transpose of a vector or matrix is denoted by $(\cdot)^\top$, and the complex conjugate by $(\cdot)^*$. The imaginary unit is denoted by j, satisfying $j^2 = -1$, and the absolute value of a scalar is written as $|\cdot|$. For a real-valued vector $\mathbf{x} \in \mathbb{R}^N$, $\operatorname{diag}(\mathbf{x})$ returns an $N \times N$ diagonal matrix with the entries of $\mathbf{x}$ on the diagonal. The ℓ_2-norm of a vector $\mathbf{x} \in \mathbb{R}^n$ is denoted $\|\mathbf{x}\|$; for a matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$, $\|\mathbf{x}\|$ is the induced ℓ_2-norm. For two vectors $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^m$, the notation $(\mathbf{x}, \mathbf{u}) \in \mathbb{R}^{n+m}$ denotes their concatenation. The symbol $\mathbf{0}$ is used to denote vectors or matrices of zeros, with dimension determined from context. The set of complex numbers is denoted $\mathbb{C}$. For a complex vector $\boldsymbol{x} \in \mathbb{C}^N$, $\Re(\boldsymbol{x}) \in \mathbb{R}^N$ denotes its real part and $\Im(\boldsymbol{x}) \in \mathbb{R}^N$ its imaginary part. We denote by $\mathbb{N}_0$ the set of non-negative integers, and by $\mathbb{N}_{>0}$ the set of positive integers. The set of all integers is denoted by $\mathbb{Z}$.

This assumption is supported by the findings in, e.g., [27]–[29]. This assumption will be utilized only in the analysis of the algorithms; it will not play a role in the algorithmic design and practical implementations of the proposed methods.

Remark II.1 (*Model and notation*)**.** We note that the framework proposed in this paper is applicable to multi-phase distribution systems with both wye and delta connections under the same Assumption 1. However, to simplify the notation and to streamline the exposition, we outline the framework using a single-phase model. □

B. AC OPF Formulation

Several formulations for the AC OPF at the distribution level has been proposed in the literature; see, for example, the survey [1] and the representative works [3], [5], [30], [31]. In this section, we structure our presentation around an AC OPF formulation that includes constraints on node voltages, line currents, and operating ranges of DERs.

Recall that $\boldsymbol{u} = [p_1, \dots, p_G, q_1, \dots, q_G]^\top \in \mathbb{R}^{2G}$ represents the vector of DER active and reactive power injections, and recall that $\mathcal{M} \subseteq \mathcal{N}$ is a set of nodes where voltages are monitored and controlled. In particular, for the latter, let lower and upper bounds on the voltage magnitudes be denotes as $\underline{V}$ and $\overline{V}$, respectively. Additionally, let $\bar{I}$ be an ampacity limit for the L lines that are monitored. We then consider the following problem formulation to compute the DERs' power setpoints:

$$\begin{aligned} \mathsf{U}^*(\boldsymbol{s}_l, \boldsymbol{\theta}) := \arg\min_{\boldsymbol{u} \in \mathcal{C}} \quad & C_v(\boldsymbol{v}(\boldsymbol{u}; \boldsymbol{s}_l)) + C_p(\boldsymbol{u}) \\ \text{s.t.} \quad & \underline{V} \leq |\boldsymbol{v}(\boldsymbol{u}; \boldsymbol{s}_l)| \leq \overline{V}, \\ & |\boldsymbol{i}(\boldsymbol{u}; \boldsymbol{s}_l)| \leq \bar{I}, \\ & (p_i, q_i) \in \mathcal{C}_i(\boldsymbol{\theta}_{u,i}), \quad \forall i = 1, \dots, G, \end{aligned} \tag{3}$$

where $C_v : \mathbb{R}^M \to \mathbb{R}$ is a cost associated with the voltage profile, $C_p : \mathbb{R}^{2G} \to \mathbb{R}$ captures DER-specific costs, and the set $\mathcal{C}_i(\boldsymbol{\theta}_{u,i})$ encodes constraints for the ith DER, such as capacity and hardware limits as well as grid code requirements. The inequalities in the voltage and current constraints are taken entry-wise. We allow a parametric representation of the set through parameters $\boldsymbol{\theta}_{u,i}$; to this end, we assume the set $\mathcal{C}_i(\boldsymbol{\theta}_{u,i})$ can be expressed as

$$\mathcal{C}_i(\boldsymbol{\theta}_{u,i}) = \{(p_i, q_i) \in \mathbb{R}^2 : \ell_i(p_i, q_i, \boldsymbol{\theta}_{u,i}) \leq \mathbf{0}_{n_{c_i}}\} \tag{4}$$

where ℓ_i is a vector-valued function modeling power limits, and the inequality is taken entry-wise. The function ℓ_i is assumed to be differentiable. For example, if the ith DER is an inverter-interfaced controllable renewable source, then $\ell_i(p_i, q_i, \boldsymbol{\theta}_{u,i}) = [p_i^2 + q_i^2 - s_{n,i}^2, p_i - p_{\max,i}, -p_i]^\top$, where $\boldsymbol{\theta}_{u,i} = (p_{\max,i}, s_{n,i})$ with $s_{n,i}$ and $p_{\max,i}$ the inverter rated size and the maximum available active power, respectively. The overall set of inputs that parametrize the problem (3) is denoted as $\boldsymbol{\theta} := (\boldsymbol{\theta}_{u,i}, \dots, \boldsymbol{\theta}_{u,G}, \underline{V}, \overline{V}, \bar{I})$; these inputs are in addition to $\boldsymbol{s}_l$. We will use the notation $\mathcal{C} = \mathcal{C}_1 \times \ldots \times \mathcal{C}_G$ and we define the set $\mathcal{S}(\boldsymbol{s}_l) := \mathcal{S}_v(\boldsymbol{s}_l) \cap \mathcal{S}_i(\boldsymbol{s}_l)$, where:

$$\begin{aligned} \mathcal{S}_v(\boldsymbol{s}_l) &:= \{\boldsymbol{u} \in \mathcal{C} : \underline{V} \leq |\boldsymbol{v}(\boldsymbol{u}; \boldsymbol{s}_l)| \leq \overline{V}\} \\ \mathcal{S}_i(\boldsymbol{s}_l) &:= \{\boldsymbol{u} \in \mathcal{C} : |\boldsymbol{i}(\boldsymbol{u}; \boldsymbol{s}_l)| \leq \bar{I}\}. \end{aligned}$$

The feasible set of (3) is $\mathcal{S}_v(\boldsymbol{s}_l) \cap \mathcal{S}_i(\boldsymbol{s}_l) \cap \mathcal{C}$. In the following, for notational simplicity, we drop the dependence on $\boldsymbol{s}_l$.

It is well known that the AC OPF is nonconvex and may admit multiple globally optimal and locally optimal solutions. Accordingly, the function $(\boldsymbol{s}_l, \boldsymbol{\theta}) \mapsto \mathsf{U}^*(\boldsymbol{s}_l, \boldsymbol{\theta})$ that *maps parameters of the problem into globally optimal solutions to the AC OPF* is a *set-valued function*. Since identifying a solution $\boldsymbol{u}^* \in \mathsf{U}^*(\boldsymbol{s}_l, \boldsymbol{\theta})$ is in general difficult, we consider the (sub-)set of $\mathsf{U}^*(\boldsymbol{s}_l, \boldsymbol{\theta})$ that contains points that are local minimizers and isolated KKT points for (3); we denote such set as $\mathsf{U}^{\mathsf{lm}}(\boldsymbol{s}_l, \boldsymbol{\theta})$ (although we note that some local minimizers can also be global minimizers). In the following, we explain our approach to identify local minimizers in $\mathsf{U}^{\mathsf{lm}}(\boldsymbol{s}_l, \boldsymbol{\theta})$.

C. Proposed Mathematical Framework and Implementations

Our proposed technical approach is grounded on a mathematical model of the form:

$$\dot{\boldsymbol{u}} = \eta F(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}), \tag{5a}$$

$$\underbrace{\begin{bmatrix} \boldsymbol{\nu} \\ \boldsymbol{\iota} \end{bmatrix}}_{:=\boldsymbol{\xi}} = \underbrace{\begin{bmatrix} |\boldsymbol{v}(\boldsymbol{u}; \boldsymbol{s}_l)| \\ |\boldsymbol{i}(\boldsymbol{u}; \boldsymbol{s}_l)| \end{bmatrix}}_{:=H(\boldsymbol{u}; \boldsymbol{s}_l)} + \underbrace{\begin{bmatrix} \boldsymbol{n}_v \\ \boldsymbol{n}_i \end{bmatrix}}_{:=\boldsymbol{n}} \tag{5b}$$

where: *(a)* $F(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})$ is a given algorithmic map, utilized to seek local minimizers in $\mathsf{U}^{\mathsf{lm}}(\boldsymbol{s}_l, \boldsymbol{\theta})$; this map updates $\boldsymbol{u}$ based on voltages $\boldsymbol{v}$, currents $\boldsymbol{i}$, and the problem parameters $\boldsymbol{\theta} = (\boldsymbol{\theta}_{u,i}, \dots, \boldsymbol{\theta}_{u,G}, \underline{V}, \bar{V}, \bar{I})$. *(b)* $H(\boldsymbol{u}; \boldsymbol{s}_l)$ in (5b) represents a power flow solution map; in particular, given $\boldsymbol{u}, \boldsymbol{s}_l$, one solves for (2) to obtain voltages and currents (and then computes their absolute values). In (5b), $\boldsymbol{n}$ represents an error or a perturbation in the computation of $\boldsymbol{\xi}$.

We design $F(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})$ based on CBF tools [32] and the safe gradient flow [23], [24]. In particular, $F(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})$ is given by:

$$\dot{\boldsymbol{u}} = \eta F(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) \tag{6}$$

$$\begin{aligned} & F(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) := \\ & \arg\min_{\boldsymbol{z} \in \mathbb{R}^{2G}} \|\boldsymbol{z} + \nabla C_p(\boldsymbol{u}) + \boldsymbol{J}_v(\boldsymbol{u}; \boldsymbol{s}_l)^\top \nabla C_v(\boldsymbol{\nu})\|^2 \\ & \quad \text{s.t.} - \boldsymbol{J}_v(\boldsymbol{u}; \boldsymbol{s}_l)^\top \boldsymbol{z} \leq -\beta\left(\mathbf{1}\underline{V} - \boldsymbol{\nu}\right) \\ & \qquad \boldsymbol{J}_v(\boldsymbol{u}; \boldsymbol{s}_l)^\top \boldsymbol{z} \leq -\beta\left(\boldsymbol{\nu} - \bar{V}\mathbf{1}\right) \\ & \qquad \boldsymbol{J}_i(\boldsymbol{u}; \boldsymbol{s}_l)^\top \boldsymbol{z} \leq -\beta\left(\boldsymbol{\iota} - \bar{I}\mathbf{1}\right) \\ & \qquad \boldsymbol{J}_{\ell_i}(p_i, q_i)^\top \boldsymbol{z} \leq -\beta \ell_i(p_i, q_i), \qquad \forall i \in \mathcal{G} \end{aligned} \tag{7}$$

where $\boldsymbol{J}_{\ell_i}(p_i, q_i)$ is the Jacobian of $(p_i, q_i) \mapsto \ell_i(p_i, q_i)$, $\beta > 0$ is a design parameter, and $\eta > 0$ is the controller gain and is a design parameter. As discussed in [23], the controller in (6) serves as an approximation of the projected gradient flow. This approximation, which leverages CBF models, ensures that the feasible set of (3) is forward invariant. This invariance property is a key motivation behind initiating our design from (7), and it is supported by our recent work in [24], where the function $F(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}_F)$ was specifically designed to address an AC OPF problem with voltage constraints.

On the other hand, the update in (5b) lends itself to two distinct practical implementations as outlined below:

△ *Online feedback-based implementation:* Once the update (5a) is performed, the power setpoints $\boldsymbol{u}$ are transmitted

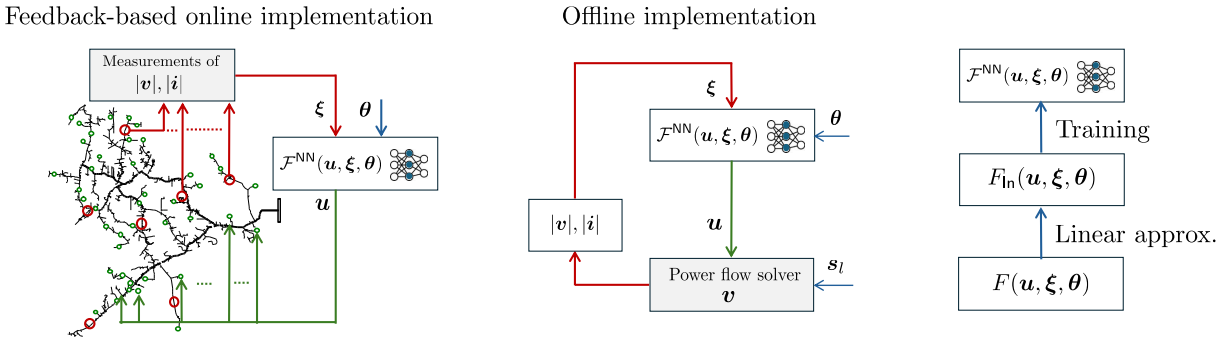


Fig. 1: *(Left)* Feedback-based online implementation leveraging measurements from the network. *(Center)* Offline implementation with power-flow solver. *(Right)* Design process.

to (and implemented by) the DERs; then, the system operator collects measurements of actual voltages and currents from the system, or leverages pseudo-measurements. This implementation is aligned with existing works on feedback-based optimization [2]–[5], [24], [30], and it is illustrated in Figure 1.

△ *Model-based offline implementation:* In this case, (5b) represents the solution to the AC power flow equations (2) via numerical methods. For example, given $\boldsymbol{s}_l$, voltages $\boldsymbol{v}$ can be found using the fixed-point method [28], [33]. This leads to the offline solution of (3) illustrated in Figure 1.

When $F(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})$ involves the solution of a quadratic program (QP) as in (7), the process in (5) can be computationally intensive; the time required to identify a solution may not align with the time scales at which loads $\boldsymbol{s}_l$ and system parameters $\boldsymbol{\theta}$ evolve [8]. More broadly, similar arguments would apply to cases where $F(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})$ is designed using different algorithmic approaches involving projections onto manifolds [34] or inversions of (potentially large) matrices as in Newton-type methods [16]. The idea is then to train a neural network to approximate $F(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})$. Letting $\mathcal{F}^{\mathsf{NN}} : \mathbb{R}^{2G} \times \mathbb{R}^{M+L} \times \mathbb{R}^{n_\theta} \to \mathbb{R}^{2G}$ the neural network map, we consider the following modification of (5):

$$\dot{\boldsymbol{u}} = \eta \mathcal{F}^{\mathsf{NN}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}), \tag{8a}$$

$$\boldsymbol{\xi} = H(\boldsymbol{u}; \boldsymbol{s}_l) + \boldsymbol{n} \tag{8b}$$

where we recall that $\boldsymbol{\xi}$ represents either a solution to the power flow equations via numerical methods or measurements of voltages and currents. Based on the model (8) the problem addressed in the remainder of the paper is as follows.

Problem 1. *Design and train a neural network $\mathcal{F}^{\mathsf{NN}}$ to emulate $F(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})$ in* (7) *so that the algorithm* (8)*: (a) converges to a solution $\boldsymbol{u}^* \in \mathsf{U}^{\mathsf{lm}}(\boldsymbol{\theta})$ of the AC OPF problem* (3)*; (b) ensures that voltage, current, and DER constraints are satisfied at any time during the execution of the algorithm.* □

The term "at any time" refers to the fact that the algorithm (8) is expected to produce points that are practically feasible for (3) even if it is terminated before convergence. This is a key for online AC OPF implementations as in Figure 1(left), and a desirable feature of offline methods. We provide some remarks to support our design approach.

D. Motivations and Rationale

An approach different than the one proposed in (8) in the context of learning for the AC OPF problems is to train a neural network to directly identify optimal solutions in $\mathsf{U}^*(\boldsymbol{s}_l, \boldsymbol{\theta})$, solutions in $\mathsf{U}^{\mathsf{lm}}(\boldsymbol{s}_l, \boldsymbol{\theta})$, or the set of KKT points $\mathsf{U}^{\mathsf{kkt}}(\boldsymbol{s}_l, \boldsymbol{\theta})$; see, for example, [7], [12]–[14] and [19]. We provide a comparison next.

△ *Mapping vs set-valued mapping*

- For any given $\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}$, $F(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})$ is defined as the *unique* optimal solution of the convex QP (7). Moreover, under some mild assumptions, the mapping $F(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})$ is locally Lipschitz (in all its arguments) [23], [24]. Therefore, in our approach, the neural network approximates a mapping that is continuous in its arguments. Accordingly, our learning problem is well posed.
- On the other hand, $\mathsf{U}^*(\boldsymbol{s}_l, \boldsymbol{\theta})$, $\mathsf{U}^{\mathsf{lm}}(\boldsymbol{s}_l, \boldsymbol{\theta})$, and $\mathsf{U}^{\mathsf{kkt}}(\boldsymbol{s}_l, \boldsymbol{\theta})$ are in general *sets*; therefore, $(\boldsymbol{s}_l, \boldsymbol{\theta}) \mapsto \mathsf{U}^*(\boldsymbol{\theta})$ is a set-valued mapping. Such a set-valued mapping cannot be approximated with the mapping of the neural network; see the discussion in, e.g. [21] and [12]. One workaround suggested in [21] is when the number of solutions to the AC OPF is finite and all the solutions can be identified; in this case, letting as an example $\boldsymbol{u}^{\mathsf{kkt}}(\boldsymbol{s}_l, \boldsymbol{\theta})$ be a vector collecting all the KKT points for given parameters $(\boldsymbol{s}_l, \boldsymbol{\theta})$, one can use a neural network to approximate $(\boldsymbol{s}_l, \boldsymbol{\theta}) \mapsto \boldsymbol{u}^{\mathsf{kkt}}(\boldsymbol{s}_l, \boldsymbol{\theta})$. However, the solutions of the AC OPF cannot be, in general, enumerated.

△ *Number of inputs*

- To approximate $F(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})$, the inputs to the training are the voltages at the monitored nodes, the currents at the monitored lines, the current setpoints of the DERs, and the parameters $\boldsymbol{\theta}$. Here, the training does not include the loads $\boldsymbol{s}_l$ as inputs.
- To approximate $\mathsf{U}^*(\boldsymbol{s}_l, \boldsymbol{\theta})$, $\mathsf{U}^{\mathsf{lm}}(\boldsymbol{s}_l, \boldsymbol{\theta})$, or $\mathsf{U}^{\mathsf{kkt}}(\boldsymbol{s}_l, \boldsymbol{\theta})$ directly, the inputs are the loads $\boldsymbol{s}_l$ throughout the network and the inputs $\boldsymbol{\theta}$. When the number of loads is larger than the controlled DERs and the monitored voltages and currents, this leads to more inputs in the training task.

△ *Feasibility guarantees*

- As shown in Section V, our method ensures that iterates $\boldsymbol{u}$ are practically feasible; i.e., we characterize the worst-case violation of a constraint. With this information, and by tightening the constraints during the training process,

one can ensure that our method generates points that are feasible for the AC OPF.
- Existing methods that "emulate" solutions in $\mathsf{U}^*(\boldsymbol{s}_l, \boldsymbol{\theta})$, $\mathsf{U}^{\mathsf{lm}}(\boldsymbol{s}_l, \boldsymbol{\theta})$, or $\mathsf{U}^{\mathsf{kkt}}(\boldsymbol{s}_l, \boldsymbol{\theta})$ do not guarantee feasibility of the generated outputs. Post-processing could adjust the solution to make it feasible, but that may involve heuristics that do not have feasibility guarantees.

III. Neural Network-based OPF Pursuit

In this section, we provide details on the algorithmic design and we discuss its implementation.

A. Algorithmic Design

The map $F(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})$ in (6) requires computing the Jacobian matrices of function $H(\boldsymbol{u}, \boldsymbol{s}_l)$. To favor a lower complexity training procedure, we rely on a linear approximation of the power flow equations (2); several linear approximation approaches can be found in the literature; see for example, [27], [28], [31] and references therein. In general, one can find linear approximations of the form

$$|\boldsymbol{v}(\boldsymbol{u}; \boldsymbol{s}_l)| \approx \boldsymbol{\Gamma}_v \boldsymbol{u} + \bar{\boldsymbol{v}}(\boldsymbol{s}_l), \quad |\boldsymbol{i}(\boldsymbol{u}; \boldsymbol{s}_l)| \approx \boldsymbol{\Gamma}_i \boldsymbol{u} + \bar{\boldsymbol{i}}(\boldsymbol{s}_l), \quad (9)$$

where the matrices $\boldsymbol{\Gamma}_v, \boldsymbol{\Gamma}_i$ and the vectors $\bar{\boldsymbol{v}}(\boldsymbol{s}_l), \bar{\boldsymbol{i}}(\boldsymbol{s}_l)$ can be computed using the methods in [27], [28], [31]. The matrices $\boldsymbol{\Gamma}_v, \boldsymbol{\Gamma}_i$ can be precomputed, as they do not depend on $\boldsymbol{u}$ or $\boldsymbol{s}_l$. Using (9), we can utilize the following approximation of (7):

$$\dot{\boldsymbol{u}} = \eta F_{\mathsf{ln}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) \quad (10)$$

$$\begin{aligned} F_{\mathsf{ln}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) := \arg \min_{\boldsymbol{z} \in \mathbb{R}^{2G}} \ & \|\boldsymbol{z} + \nabla C_p(\boldsymbol{u}) + \boldsymbol{\Gamma}_v^\top \nabla C_v(\boldsymbol{\nu})\|^2 \\ \text{s.t.} \ & -\boldsymbol{\Gamma}_v^\top \boldsymbol{z} \leq -\beta \left(\mathbf{1}\underline{V} - \boldsymbol{\nu}\right) \\ & \boldsymbol{\Gamma}_v^\top \boldsymbol{z} \leq -\beta \left(\boldsymbol{\nu} - \bar{V}\mathbf{1}\right) \\ & \boldsymbol{\Gamma}_i^\top \boldsymbol{z} \leq -\beta \left(\boldsymbol{\iota} - \bar{I}\mathbf{1}\right) \\ & \boldsymbol{J}_{\ell_i}(\boldsymbol{u}_i)^\top \boldsymbol{z} \leq -\beta \ell_i(\boldsymbol{u}), \ i \in \mathcal{G} \end{aligned} \quad (11)$$

where we have replaced the Jacobian matrices of the power flow equations with the ones of the linear approximations. In our forthcoming analysis in Section V, we quantify the effect of the linear approximation error in the overall performance.

Similarly to (7), (11) is a convex QP with a unique optimal solution. Moreover, from [35, Theorem 3.6], it follows that $\boldsymbol{u} \mapsto F_{\mathsf{ln}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})$ is locally Lipschitz over $\mathcal{B}(\boldsymbol{u}, r_1) := \{\boldsymbol{z} : \|\boldsymbol{z} - \boldsymbol{u}\| < r_1\}$ of $\boldsymbol{u}$, for any $\boldsymbol{\xi}$ and $\boldsymbol{\theta}$.

The next step, as illustrated in Figure 1(right), is to consider a neural network $\mathcal{F}^{\mathsf{NN}} : \mathbb{R}^{2G} \times \mathbb{R}^{M+L} \times \mathbb{R}^{n_\theta} \to \mathbb{R}^{2G}$, which will be trained to approximate the mapping $(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) \mapsto F_{\mathsf{ln}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})$. In particular, consider a fully connected feedforward neural network (FNN), defined recursively as:

$$\begin{aligned} \boldsymbol{y} &= \mathcal{F}^{\mathsf{NN}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) := W^{(H)} \boldsymbol{\varphi}^{(H)} + b^{(H)}, \\ \boldsymbol{\varphi}^{(i)} &= \Phi^{(i)} \left(W^{(i-1)} \boldsymbol{\varphi}^{(i-1)} + b^{(i-1)} \right), \quad i = 1, \ldots, H, \\ \boldsymbol{\varphi}^{(0)} &= [\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}], \end{aligned} \quad (12)$$

where H is the number of hidden layers, $W^{(i)} \in \mathbb{R}^{n_{i+1} \times n_i}$ and $b^{(i)} \in \mathbb{R}^{n_{i+1}}$ are the weights and biases, and $\Phi^{(i)}$ is a Lipschitz-continuous activation function (e.g., ReLU, leaky ReLU, or sigmoid) The network outputs in (12) are $\boldsymbol{y} = \dot{\boldsymbol{u}}$.

For the training procedure, suppose that N_{train} training points are available, and they are taken from a compact set $(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) \in \mathcal{C}_{\text{train}} \times \mathcal{E}_{\text{train}} \times \Theta_{\text{train}}$, where $\mathcal{C}_{\text{train}}$ is a superset of the feasible region of (3), $\mathcal{E}_{\text{train}}$ is an inflation of the set of operational voltages, and Θ_{train} is formed based on inverters' operating conditions. Thus, each training point is given by the input $(\boldsymbol{u}^{(k)}, \boldsymbol{\xi}^{(k)}, \boldsymbol{\theta}^{(k)})$ and the corresponding output $\boldsymbol{y}^{(k)} = F_{\mathsf{ln}}(\boldsymbol{u}^{(k)}, \boldsymbol{v}^{(k)}, \boldsymbol{\theta}^{(k)})$, for $k = 1, \ldots, N_{\text{train}}$. Then, we consider minimizing the following loss function:

$$\mathcal{L}(\boldsymbol{W}, \boldsymbol{b}) := \frac{1}{N_{\text{train}}} \sum_{n=1}^{N_{\text{train}}} \left\| \boldsymbol{y}^{(k)} - \mathcal{F}^{\mathsf{NN}}(\boldsymbol{u}^{(k)}, \boldsymbol{v}^{(k)}, \boldsymbol{\theta}^{(k)}) \right\|_2^2 \quad (13)$$

where the dependence of $\mathcal{F}^{\mathsf{NN}}$ on $\boldsymbol{W}, \boldsymbol{b}$ is dropped for notational convenience. The training routine is presented in Algorithm 1.

Algorithm 1 Offline training

Generate or collect training points
  For each time instant or episode $\{t_k\}_{k=1}^{N_{\text{train}}}$:
    Obtain $\boldsymbol{v}^{(k)}$, $\boldsymbol{i}^{(k)}$, and $\boldsymbol{u}^{(k)}$
    Obtain parameters: $\boldsymbol{\theta}^{(k)}$
    Compute: $\boldsymbol{y}^{(k)} = F_{\mathsf{ln}}(\boldsymbol{u}^{(k)}, \boldsymbol{\xi}^{(k)}, \boldsymbol{\theta}^{(k)})$
Train neural network
  Solve $\min_{\boldsymbol{W}, \boldsymbol{b}} \mathcal{L}(\boldsymbol{W}, \boldsymbol{b})$

We note that the training dataset can be generated offline by repeating step [S1] for a given set of values for voltages, currents, and DERs' powers, or it can be formed online by collecting measurements from the distribution grid. The trained FNN $\mathcal{F}^{\mathsf{NN}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})$ is then used in (8) to solve the AC OPF online (cf. Figure 1(left) or offline (cf. Figure 1(center)).

B. Online and Offline Implementations

In this section, we provide more details on the online and offline implementations of our proposed method. The feedback-based online implementation is illustrated in Figure 1(left); here, the parameters $\boldsymbol{\theta}(t)$ are time-varying since they include the power available from renewable-based DERs, which may change with evolving ambient conditions [3]. The overall algorithm is tabulated as Algorithm 2. Similar to existing feedback-based algorithms, Algorithm 2 does not require any information about the loads $\boldsymbol{s}_l$. In this implementation, the error term $\boldsymbol{n}$ in (5a) and (8a) represents errors in the measurements of voltages and currents, or in the computation of pseudo-measurements; these errors are small or even negligible [36].

On the other hand, the offline implementation of Figure 1(center) is tabulated as Algorithm 3. For this offline implementation, one needs information about the loads $\boldsymbol{s}_l$. In [S1a], solutions to the power flow (PF) equations can be identified using, for example, sweeping methods [26] or fixed-point methods [28]. In this implementation, the error $\boldsymbol{n}$ in (5a) and (8a) represents the numerical accuracy of the PF method. The algorithm is executed until convergence, or for a prescribed amount of time t_d.

Algorithm 2 Online feedback-based implementation

Initialization
  Load pretrained model $\mathcal{F}^{\mathsf{NN}}$, pick $\eta > 0$.
Real-Time operation $t \geq 0$**:**
  Measure DERs' setpoints $\boldsymbol{u}(t)$
  Measure $|\boldsymbol{v}(t)|$ and $|\boldsymbol{i}(t)|$ from selected locations
  Obtain parameters $\boldsymbol{\theta}(t)$
  Perform update: $\dot{\boldsymbol{u}}(t) = \eta \mathcal{F}^{\mathsf{NN}}\left(\boldsymbol{u}(t), \boldsymbol{\xi}(t), \boldsymbol{\theta}(t)\right)$
  Send $\boldsymbol{u}(t)$ to DERs and go to 4.

Algorithm 3 Offline implementation

Initialization
  Load pretrained model $\mathcal{F}^{\mathsf{NN}}$, pick $\eta > 0$.
  Load parameters $\boldsymbol{\theta}$, load $\boldsymbol{s}_l$, set $\boldsymbol{u}(0)$.
Perform until convergence or until t_d**:**
  Given $\boldsymbol{s}_l, \boldsymbol{u}(\tau)$, solve PF equations to get $\boldsymbol{v}(\tau)$
  Compute $|\boldsymbol{v}(\tau)|$ and $|\boldsymbol{i}(\tau)|$ from selected locations
  Perform update: $\dot{\boldsymbol{u}}(\tau) = \eta \mathcal{F}^{\mathsf{NN}}\left(\boldsymbol{u}(\tau), \boldsymbol{\xi}(\tau), \boldsymbol{\theta}\right)$
  Go to 5.

IV. Experimental Results in a Distribution Feeder

We test the proposed method illustrated in Figure 1(left) – which in this section we refer to as neural network-based safe gradient flow (NN-SGF) in short - on a voltage regulation problem.

We consider the medium voltage network (20 kV) shown in Figure 2 (see [25]). The network contains photovoltaic (PV) inverters at selected buses capable of adjusting both active and reactive power. Each inverter $i \in \mathcal{G}$ injects power $\boldsymbol{u}_i = (p_i, q_i) \in \mathbb{R}^2$ within a feasible set:

$$\mathcal{C}_i(\boldsymbol{\theta}_{u,i}) = \left\{ (p_i, q_i) \in \mathbb{R}^2 \;\middle|\; \begin{bmatrix} p_i^2 + q_i^2 - s_{n,i}^2 \\ p_i - p_{\max,i} \\ -p_i \\ -0.44\, s_{n,i} - q_i \\ q_i - 0.44\, s_{n,i} \end{bmatrix} \leq \mathbf{0} \right\}, \quad (14)$$

where $s_{n,i}$ represents the inverter's nominal apparent power rating, randomly selected from the set $\{490, 620, 740\}$ kVA to capture the range of deployment scales observed in practice. The upper bound $p_{\max,i}$ denotes the maximum available active power at time t. Together, the pair $\boldsymbol{\theta}_{u,i} = (p_{\max,i}, s_{n,i})$ defines the parameterization of the set $\mathcal{C}_i(\boldsymbol{\theta}_{u,i})$. This limit is consistent with practical deployment settings and in accordance with IEEE Std 1547-2018. We also assume that $p_{\max,i}$ is known at the DER level (via maximum power point tracking). The cost function in the AC OPF is defined as $C_p(\boldsymbol{u}) = \sum_{i\in\mathcal{G}} c_p \left(\frac{s_{n,i}-p_i}{s_{n,i}}\right)^2 + c_q \left(\frac{q_i}{s_{n,i}}\right)^2$, with $c_p = 3$ and $c_q = 1$. This cost function aims to minimize active power curtailment and inverter losses. The first term promotes operation near the available active power, while the second penalizes reactive power usage, which contributes to higher current magnitudes and associated Joule losses.

The voltage magnitudes at monitored buses are denoted by $\boldsymbol{\nu}$ (cf. (5b)) and are obtained from the `pandapower` power flow solver[2]. The aggregated non-controllable loads and maximum available active power for PV plants is from the Open Power System Data[3]; the data has a granularity of 10 seconds, and the values have been modified to match the initial loads and PV plants nominal values present in the network. The reactive power demand is set such that the power factor is 0.9 (lagging). The voltage service limits $\bar{V}$ and $\underline{V}$ are set to 1.05 and 0.95 p.u., respectively.

With the considered data and simulation setup, we obtain the voltage profiles illustrated in Figure 3 for the case of *no control*; this is a case where a protection scheme of the PV plants disconnects the inverters if the voltage level is too high. The disconnection scheme is inspired from the CENELEC EN50549-2 standard; the PV plant changes status from running to disconnected if: (i) the voltage at the point of connection goes above 1.06 pu, (ii) the root mean square value of the voltages measured at the point of connection for the past 10 minutes goes above 1.05 pu (the voltages are measured every 10 seconds). The switching from disconnected to connected occurs randomly in the interval [1min, 10min].

As shown in Figure 3, the proposed method is tested against a challenging voltage regulation problem. We compare the solutions obtained with the following strategies:
△ *(s1)* A solution of the AC OPF every 10 seconds to match the granularity of the Open Power System Data. Here, we use the nonlinear branch flow model [37] and the solver `IPOPT`. We refer to this case as batch optimization (BO).
△ *(s2)* Our solution strategy in (8) deployed in the online feedback-based configuration in Figure 1(left). Here, we run *one iteration* of the NN-SGF time a measurement is collected (as in standard feedback optimization methods).
△ *(s3)* A strategy similar to, e.g., [7], [10], [11] where a neural network is used to approximate $\mathsf{U}^{\mathsf{lm}}(\boldsymbol{s}_l, \boldsymbol{\theta})$; i.e., to emulate solutions of the BO directly. We refer to this as NN-BO.

A. Dataset Generation and Training

We simulate $N_{\text{training}} = 6{,}000$ independent operating conditions, each defined by a distinct grid configuration with varying load profiles, DER capacities, and voltage regulation constraints. Each condition is associated with a randomly sampled time instant $t_n \sim \mathcal{U}(t_{\min}, t_{\max})$, Where $t_{\min} =$ 06:00, $t_{\max} =$ 20:00. For each sampled time t_n, we simulate power flow using `pandapower` to obtain voltages $\boldsymbol{\nu}(t_n) = \{V_j(t_n)\}_{j\in\mathcal{M}}$, along with DER setpoints $\{p_i(t_n), q_i(t_n)\}_{i\in\mathcal{G}}$, constraint parameters $\{p_{\max,i}(t_n), s_{n,i}\}_{i\in\mathcal{G}}$, and voltage bounds $\{\underline{V}_j, \overline{V}_j\}_{j\in\mathcal{M}}$. For each operating condition, we run $N_{\text{iter}} = 10$ iterations of $F_{\mathsf{ln}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})$ using a forward Euler discretization, with $\eta = 0.2$. At each iteration $k \in \{1, \ldots, N_{\text{iter}}\}$, we record the DER setpoints $\boldsymbol{u}^{(n,k)} = \{p_i^{(k)}(t_n), q_i^{(k)}(t_n)\}_{i\in\mathcal{G}}$, the voltage measurements $\boldsymbol{\nu}^{(n,k)} = \{V_j^{(k)}(t_n)\}_{j\in\mathcal{M}}$, and the constraint parameters $p_{\max,i}(t_n)$, $s_{n,i}$, $\underline{V}_j$, and $\overline{V}_j$. We also extract the current magnitudes $\boldsymbol{\iota}^{(n,k)} \in \mathbb{R}^L$ at monitored lines. These are used to construct the state vector $\boldsymbol{\xi}^{(n,k)} = [(\boldsymbol{\nu}^{(n,k)})^\top, (\boldsymbol{\iota}^{(n,k)})^\top]^\top \in \mathbb{R}^{M+L}$, consistent with the online and offline implementation setup. The full constraint parameter vector is denoted

[2]See https://www.pandapower.org.
[3]https://data.open-power-system-data.org/household_data/2020-04-15

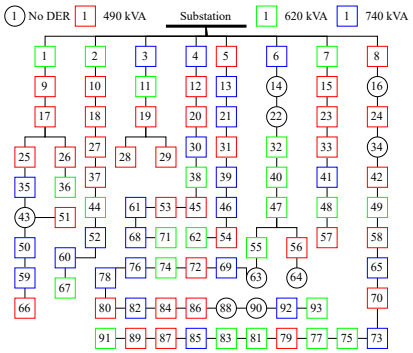


Fig. 2: Distribution network used in the simulations [25].

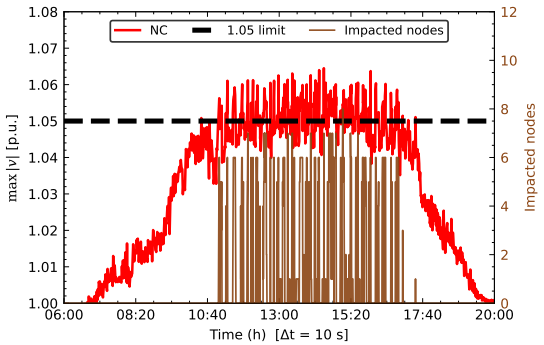


Fig. 3: Overvoltage events and number of nodes impacted with the considered simulation setup, when no control actions are implemented.

$\boldsymbol{\theta}^{(n)} = (\boldsymbol{\theta}^{(n)}_{u,1}, \ldots, \boldsymbol{\theta}^{(n)}_{u,G}, \underline{V}, \overline{V}, \overline{I})$, and is treated as fixed across SGF iterations at time t_n. The control update is computed as $\boldsymbol{y}^{(n,k)} = F_{\mathsf{ln}}(\boldsymbol{u}^{(n,k)}, \boldsymbol{\xi}^{(n,k)}, \boldsymbol{\theta}^{(n)}) \in \mathbb{R}^{2G}$ and used as the training label. The corresponding input vector $\mathbf{x}^{(n,k)} \in \mathbb{R}^{3G+M}$ is constructed by concatenating the normalized active power deviations $\left\{(p_i^{(k)}(t_n) - p_{\max,i}(t_n))/s_{n,i}, i \in \mathcal{G}\right\}$, normalized reactive powers $\left\{q_i^{(k)}(t_n)/s_{n,i}, i \in \mathcal{G}\right\}$, normalized voltage magnitudes $\left\{(V_j^{(k)}(t_n) - \underline{V}_j)/(\overline{V}_j - \underline{V}_j), j \in \mathcal{M}\right\}$, and the active DER limits $\{p_{\max,i}(t_n), i \in \mathcal{G}\}$. This normalization ensures all features lie in comparable ranges and are scaled relative to their physical limits (e.g., $s_{n,i}$ and voltage bounds), improving numerical stability. This process results in a dataset of $N_{\text{train}} = N_{\text{training}} \cdot N_{\text{iter}} = 60{,}000$ input-output pairs $\mathcal{D}_{\text{train}} = \left\{\left(\mathbf{x}^{(n,k)}, \mathbf{y}^{(n,k)}\right)\right\}_{n,k} \subset \mathbb{R}^{3G+M} \times \mathbb{R}^{2G}$. For evaluation, we construct a disjoint test set $\mathcal{D}_{\text{test}} = \left\{\left(\mathbf{x}^{(n,k)}, \mathbf{y}^{(n,k)}\right)\right\}_{n,k} \subset \mathbb{R}^{3G+M} \times \mathbb{R}^{2G}$ using $N_{\text{testing}} = 1000$ randomly sampled times $t_n \sim \mathcal{U}(06{:}00,\ 20{:}00)$, with the same SGF update procedure repeated for each test case. We ensure that $\mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{test}} = \emptyset$.

Learning Model, Training, and Evaluation. We train a fully connected FNN $\mathcal{F}^{\mathsf{NN}} : \mathbb{R}^{2G} \times \mathbb{R}^{M+L} \times \mathbb{R}^{n_\theta} \to \mathbb{R}^{2G}$; the FNN has an architecture of the form $[3G + M, h, h, h, 2G]$, with hidden width set to $h = \alpha(3G + M)$; we use $\alpha = 2$, yielding $h = 524$ for $G = 84$ and $M = 10$. The network is implemented in `PyTorch` and trained offline using the Adam optimizer[4] with learning rate 0.001, batch size 256, dropout 0.2, and up to 500 epochs with early stopping based on validation loss from a 10% held-out subset. The loss function is the mean squared error (MSE) $\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{n,k} \left\| \mathcal{F}^{\mathsf{NN}}(\boldsymbol{u}^{(n,k)}, \boldsymbol{\xi}^{(n,k)}, \boldsymbol{\theta}^{(n)}; \theta) - \boldsymbol{y}^{(n,k)} \right\|_2^2$ over the dataset $\mathcal{D}_{\text{train}} = \left\{\left(\boldsymbol{u}^{(n,k)}, \boldsymbol{\xi}^{(n,k)}, \boldsymbol{\theta}^{(n)}, \boldsymbol{y}^{(n,k)}\right)\right\}_{n,k}$. Performance on a disjoint test set $\mathcal{D}_{\text{test}}$ is evaluated using the MSE ℓ_2-norm prediction error, defined as $\varepsilon^{\mathsf{NN}} = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{n,k} \left\| \mathcal{F}^{\mathsf{NN}}(\boldsymbol{u}^{(n,k)}, \boldsymbol{\xi}^{(n,k)}, \boldsymbol{\theta}^{(n)}) - \boldsymbol{y}^{(n,k)} \right\|_2^2$,yielding $\varepsilon^{\mathsf{NN}} = 1.7 \times 10^{-6}$ and RMSE $= 0.0013$. During online deployment, the FNN runs in inference mode at each control step t, with sampling interval $\Delta t = 10$ seconds (to match the variability of the load and PV data). Given current DER setpoints $\boldsymbol{u}(t)$, measurements $\boldsymbol{\xi}(t)$, and parameters $\boldsymbol{\theta}(t)$, the update is approximated as $\boldsymbol{u}(t + \Delta t) = \boldsymbol{u}(t) + \eta \Delta t \cdot \mathcal{F}^{\mathsf{NN}}(\boldsymbol{u}(t), \boldsymbol{\xi}(t), \boldsymbol{\theta}(t))$, with $\eta = 0.02$; setpoints are restricted to the set $\mathcal{C}$, via a projection, if not feasible to reflect hardware constraints. Given the setpoints, updated voltage magnitudes are computed via AC power flow using `pandapower`, yielding the new vector $\boldsymbol{\xi}(t + \Delta t)$ for the next control step.

For the method *(s3)* used for comparison, the training of an FNN $\mathcal{F}^{\mathsf{NN}}_{\mathsf{batch}} : (\boldsymbol{s}_l, \boldsymbol{\theta}) \mapsto \boldsymbol{u}^*$ to emulate solutions to the BO was computationally heavier as we had to increase the size of the training set to obtain acceptable performance. The training set consists of inputs $\boldsymbol{s}_l^{(k)}, \boldsymbol{\theta}^{(k)}$, with corresponding outputs $\boldsymbol{u}^{*(k)}$ obtained by solving the AC OPF using IPOPT. To generate the dataset, we sample $N_{\text{cases}} = 8{,}000$ time instants $t_n \sim \mathcal{U}(06{:}00, 20{:}00)$, record the uncontrollable loads $\boldsymbol{s}_l(t_n) := (\boldsymbol{p}_l(t_n), \boldsymbol{q}_l(t_n))$ and inverter limits $\boldsymbol{p}_{\max}(t_n)$, and apply perturbations $\boldsymbol{s}_l^{(n,k)} := (1 + \epsilon_k)\boldsymbol{s}_l(t_n)$ with $\epsilon_k \sim \mathcal{U}(-0.05, 0.05)$ for $k = 1, \ldots, 10$, introducing up to $\pm 5\%$ variability to enable localized sampling of the solution space without violating OPF feasibility at t_n. Solving the AC OPF for each perturbed point yields the target $\boldsymbol{u}^{*(n,k)} \in \mathsf{U}^{\mathsf{lm}}(\boldsymbol{s}_l^{(n,k)}, \boldsymbol{\theta}))$, giving $N_{\text{train}} = 80{,}000$ training pairs. The loss minimized is $\mathcal{L}(\theta) = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} \left\| \mathcal{F}^{\mathsf{NN}}_{\mathsf{batch}}(\boldsymbol{s}_l^{(i)}, \boldsymbol{\theta}; \theta) - \boldsymbol{u}^{*(i)} \right\|_2^2$.

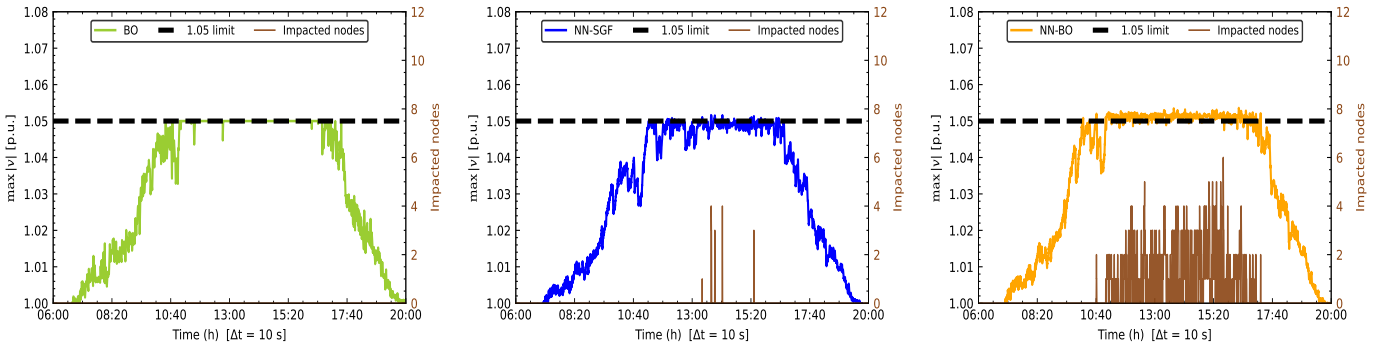
TABLE I: Training of NN-SGF (s2) and NN-BO (s3)

Method	Training points	Mean Squared Error test
NN-SGF (s2)	60,000	1.7×10^{-6}
NN-BO (s3)	80,000	8.3×10^{-5}

B. Voltage regulation and over-voltage duration

Figures 4 and 5 illustrate how different strategies perform in regulating voltage magnitudes within the bounds $[0.95, 1.05]$ p.u.; Figure 4 considers the *maximum voltage profile* across the system at every time step, as well as the *number of nodes that experience overvoltages*. The proposed NN–SGF method maintains voltages tightly bounded across all monitored nodes, with only brief and mild excursions slightly above 1.05 p.u. These short-duration deviations are well within the tolerance accepted by distribution utilities and do not compromise protection schemes. The BO method, which solves the full AC OPF offline to convergence, is used as a benchmark. The NN-BO approach, trained to emulate the BO setpoints directly, exhibits significantly more overvoltage excursions than the NN-SGF, reflected in its higher $\max T_{1.05}$ and $\text{mean}\ T_{1.05}$ as shown in Figure 5.

[4]See: https://docs.pytorch.org/docs/stable/generated/torch.optim.Adam.html



(a) AC OPF solution via IPOPT. (b) Online NN-SGF in Algorithm 2 and Fig. 1(left). (c) Neural network trained to emulate $\mathsf{U}^{\mathsf{lm}}(\boldsymbol{s}_l, \boldsymbol{\theta})$.

Fig. 4: Highest voltage profile and number of nodes experiencing overvoltages with different optimization methods : *(a)* Batch optimization (BO) using IPOPT; *(b)* the proposed NN-SGF (8), implemented in an online feedback configuration as in Figure 1(left); *(c)* method where a neural network is trained to emulate BO solutions (NN-BO).

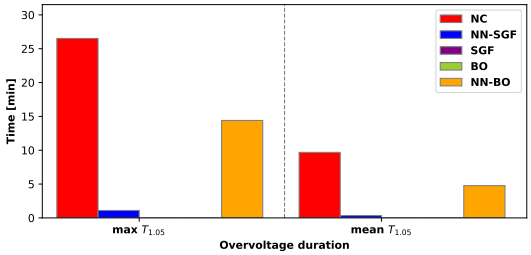


Fig. 5: Overvoltage duration for the three strategies *(s1)–(s3)* compared in Figure (4). Additionally, we consider the no control (NC) setup as in Figure 3 and the SFG.

This confirms that approaches that attempts to learn solutions to the OPF directly cannot ensure feasibility. Importantly, NN-SGF delivers effective online voltage regulation without any iterative optimization and even outperforms widely used schemes such as Volt/Var Control (VVC) and online primal dual methods investigated in [24], which typically suffer from slower response or larger transient violations. Overall, NN-SGF strikes the best balance among voltage compliance, computational efficiency, and system protection, all without introducing operational concerns.

C. Computational times

We assess the computational time of the proposed method. In Table II, we first consider an online implementation of the SGF and of the NN-SGF. We recall that the "Online step" refers to the setup in Figure 1(left) where the one evaluation of the SGF (resp., the NN-SGF) is used to generate new setpoints $\boldsymbol{u}(t)$, and the setpoints are sent to the inverters. We averaged the runtime over the full simulation horizon $t \in [06{:}00, 20{:}00]$. Obviously, the computational time of the NN-SGF is much lower, as the SGF involves solving the constrained QP in (11) to obtain the setpoint $\dot{\boldsymbol{u}}(t_n) = \eta F_{\mathsf{in}}(\boldsymbol{u}(t_n), \boldsymbol{\xi}(t_n), \boldsymbol{\theta}(t_n))$.

As a point of comparison, we consider the average computational time required by `IPOPT` to solve the AC OPF for the network in Figure 2, which is reported is Table II.

Overall, an *online* implementation of the NN-SGF achieves a $\sim 89\times$ speedup over BO and a $\sim 11\times$ speedup over the online version of the SGF proposed in [24].

We also consider the offline implementation. Here, the "Offline solution" refers to Figure 1(center), where each iteration involves one evaluation of the SGF (resp., the NN-SGF) and one solution to the PF. Again, the PF equations are solved using `pandapower`, and the average execution time of `pandapower` was 0.021 seconds. On average, the proposed scheme implemented in an offline fashion required less than 10 iterations, yielding the upper bounds provided in Table II.

We note that the proposed NN-SGF requires measurements of the voltages, while the BO and NN-BO require measurements of all the loads in the network; therefore, the actual time required by BO and NN-BO is much larger in practice [36].

TABLE II: Average computation times (in seconds). The times do not include the delay in measuring voltages (for SGF) or loads (for BO).

Method	Online step Fig. 1(left)	Offline implementation Fig. 1(center)	Offline solution
SGF	0.116	1.181	–
NN-SGF	0.011	0.308	–
BO (`IPOPT`)	–	–	0.962
NN-BO	–	–	0.010

V. Theoretical Analysis

In this section, we analyze the convergence and the ability to generate feasible points of our proposed method (8). We start with the following assumption, which imposes some mild regularity assumptions on a neighborhood of a strict locally optimal solution of the AC OPF.

Assumption 2 (*Regularity of isolated solutions*)**.** Assume that (3) is feasible and let $\boldsymbol{u}^*$ be a local minimizer and an isolated KKT point for (3), for given $\boldsymbol{p}_l, \boldsymbol{q}_l$. Assume that:
i) Strict complementarity slackness [38] and the linear independence constraint qualification (LICQ) [39] hold at $\boldsymbol{u}^*$.
ii) The maps $\boldsymbol{u} \mapsto C_p(\boldsymbol{u})$, $\boldsymbol{u} \mapsto C_v(|\boldsymbol{v}(\boldsymbol{u}; \boldsymbol{p}_l, \boldsymbol{q}_l)|)$, $\boldsymbol{u} \mapsto |\boldsymbol{v}(\boldsymbol{u}; \boldsymbol{p}_l, \boldsymbol{q}_l)|$, and $\boldsymbol{u} \mapsto |\boldsymbol{i}(\boldsymbol{u}; \boldsymbol{p}_l, \boldsymbol{q}_l)|$ are twice continuously differentiable over $\mathcal{B}(\boldsymbol{u}^*, r_1)$, and their Hessian matrices are positive semi-definite at $\boldsymbol{u}^*$.
iii) The Hessian $\nabla^2 C_p(\boldsymbol{u}^*)$ is positive definite. □

This assumption is supported by the results of [39] and used in [24]. We also impose the following assumptions on the approximation and training errors.

Assumption 3 (*Jacobian errors*)**.** $\exists\ E_v < +\infty, E_{J_v} < +\infty$ such that $\| |\boldsymbol{v}(\boldsymbol{u}; \boldsymbol{s}_l)| - (\boldsymbol{\Gamma}_v \boldsymbol{u} + \bar{\boldsymbol{v}}(\boldsymbol{s}_l)) \| \leq E_v$ and $\| \boldsymbol{\Gamma}_v - J_v(\boldsymbol{u}, \boldsymbol{s}_l) \| \leq E_{J_v}$ for any $\boldsymbol{u} \in \mathcal{B}(\boldsymbol{u}^*, r_1)$. □

Assumption 4 (*Measurement errors*)**.** $\exists\ \epsilon_n < +\infty$ such that $\|\boldsymbol{n}\| \leq \epsilon_n$. Let $\epsilon_v < +\infty$ and $\epsilon_i < +\infty$ such that $\|\boldsymbol{n}_v\| \leq \epsilon_v$ and $\|\boldsymbol{n}_i\| \leq \epsilon_i$, respectively. □

Assumption 5 (*Training errors*)**.** $\exists\ E^{\mathsf{NN}} < +\infty$ such that $\|\mathcal{F}^{\mathsf{NN}}(\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{\theta}) - F_{\ln}(\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{\theta}\| \leq E^{\mathsf{NN}}$ for all $(\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{\theta}) \in \mathcal{C}_{\text{train}} \times \mathcal{E}_{\text{train}} \times \Theta_{\text{train}}$. □

Since the line currents $\boldsymbol{i}$ can be computed from $\boldsymbol{v}$ via Ohm's Law, Assumption 3 implies that $\exists\ E_i < +\infty, E_{J_i} < +\infty$ such that $\|\|\boldsymbol{i}(\boldsymbol{u};\boldsymbol{s}_l)| - (\boldsymbol{\Gamma}_i\boldsymbol{u} + \bar{\boldsymbol{i}}(\boldsymbol{s}_l))\| \leq E_i$ and $\|\boldsymbol{\Gamma}_i - J_i(\boldsymbol{u},\boldsymbol{s}_l)\| \leq E_{J_i}$ for any $\boldsymbol{u} \in \mathcal{B}(\boldsymbol{u}^*, r_1)$. Assumptions 3-4 are motivated by the fact that the error of the linear approximation is small in a neighborhood of the optimizer [24], [27]), and that in realistic monitoring and SCADA systems, the measurement of the voltage magnitudes are affected by a negligible error [36]. Lastly, Assumption 5 follows from the approximation capabilities of neural networks over compact sets [40], [41].

To proceed, denote as $\boldsymbol{\Psi}_v := \boldsymbol{\Gamma}_v - \boldsymbol{J}_v(\boldsymbol{u};\boldsymbol{s}_l)$ and $\boldsymbol{\Psi}_i := \boldsymbol{\Gamma}_i - \boldsymbol{J}_i(\boldsymbol{u};\boldsymbol{s}_l)$ the errors in the computation of the Jacobian, and define the sets $\mathcal{E}_v = \{\boldsymbol{\Psi}_v : \|\boldsymbol{\Psi}_v\| \leq E_{J_v}\}$ and $\mathcal{E}_i = \{\boldsymbol{\Psi}_i : \|\boldsymbol{\Psi}_i\| \leq E_{J_i}\}$. Let $\mathcal{E}_n := \{\boldsymbol{n} : \|\boldsymbol{n}\| \leq \epsilon_n\}$. Define the map $F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\boldsymbol{\Psi}_v,\boldsymbol{\Psi}_i)$ as

$$
\begin{aligned}
&F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\boldsymbol{\Psi}_v,\boldsymbol{\Psi}_i) \qquad (15)\\
&:= \arg\min_{\boldsymbol{z}\in\mathbb{R}^{2G}} \|\boldsymbol{z} + \nabla C_p(\boldsymbol{u}) + (\boldsymbol{J}_v(\boldsymbol{u};\boldsymbol{s}_l) + \boldsymbol{\Psi}_v)^\top \nabla C_v(\boldsymbol{\nu})\|^2\\
&\quad \text{s.t.} -(\boldsymbol{J}_v(\boldsymbol{u};\boldsymbol{s}_l) + \boldsymbol{\Psi}_v)^\top \boldsymbol{z} \leq -\beta\left(\mathbf{1}\underline{V} - (|\boldsymbol{v}| + \boldsymbol{n}_v)\right)\\
&\qquad (\boldsymbol{J}_v(\boldsymbol{u};\boldsymbol{s}_l) + \boldsymbol{\Psi}_v)^\top \boldsymbol{z} \leq -\beta\left((|\boldsymbol{v}| + \boldsymbol{n}_v) - \bar{V}\mathbf{1}\right)\\
&\qquad (\boldsymbol{J}_i(\boldsymbol{u};\boldsymbol{s}_l) + \boldsymbol{\Psi}_i)^\top \boldsymbol{z} \leq -\beta\left((|\boldsymbol{i}| + \boldsymbol{n}_i) - \bar{I}\mathbf{1}\right)\\
&\qquad \boldsymbol{J}_{\ell_i}(\boldsymbol{u}_i)^\top \boldsymbol{z} \leq -\beta\ell_i(\boldsymbol{u}),\ i \in \mathcal{G}
\end{aligned}
$$

which is a representation of $F_{\mathsf{lm}}(\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{\theta})$ emphasizing the dependence on the errors; note also that $F(\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{\theta}) = F_{\mathsf{m}}(\boldsymbol{u},\mathbf{0},\mathbf{0},\mathbf{0})$. With this notation, we assume the following.

Assumption 6 (*Regularity*)**.** For any $\boldsymbol{u} \in \mathcal{B}(\boldsymbol{u}^*, r_1)$, and any $\boldsymbol{\Psi}_v$, $\boldsymbol{\Psi}_i$, and $\mathbf{e}$ satisfying Assumptions 3-4, the problem (15) is feasible, and satisfies the Mangasarian-Fromovitz Constraint Qualification and the constant-rank condition [35]. □

Next, we present the following intermediate result.

Lemma V.1 (*Lipschitz continuity*)**.** Let Assumption 6 hold, and assume that $\boldsymbol{u} \mapsto C_p(\boldsymbol{u})$, $\boldsymbol{\nu} \mapsto C_v(\boldsymbol{\nu})$ are twice continuously differentiable over $\mathcal{B}(\boldsymbol{u}^*, r_1)$ and for any $\boldsymbol{\nu}$ Then:
(i) For any $\boldsymbol{n} \in \mathcal{E}_n$, $\boldsymbol{\Psi}_v \in \mathcal{E}_v$, and $\boldsymbol{\Psi}_i \in \mathcal{E}_i$, $\boldsymbol{u} \mapsto F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\boldsymbol{\Psi}_v,\boldsymbol{\Psi}_i)$ is locally Lipschitz at $\boldsymbol{u}$, $\boldsymbol{u} \in \mathcal{B}(\boldsymbol{u}^*, r_1)$.
(ii) For any $\boldsymbol{u} \in \mathcal{B}(\boldsymbol{u}^*, r_1)$, $\boldsymbol{\Psi}_v \in \mathcal{E}_v$, and $\boldsymbol{\Psi}_i \in \mathcal{E}_i$, $\boldsymbol{n} \mapsto F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\boldsymbol{\Psi}_v,\boldsymbol{\Psi}_i)$ is Lipschitz with constant $\ell_n \geq 0$ over $\mathcal{E}_n$.
(iii) For any $\boldsymbol{u} \in \mathcal{B}(\boldsymbol{u}^*, r_1)$, $\boldsymbol{n} \in \mathcal{E}_n$, $\boldsymbol{\Psi}_i \in \mathcal{E}_i$, $\boldsymbol{\Psi}_v \mapsto F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\boldsymbol{\Psi}_v,\boldsymbol{\Psi}_i)$ is ℓ_{J_v}-Lipschitz over $\mathcal{E}_v$.
(iv) For any $\boldsymbol{u} \in \mathcal{B}(\boldsymbol{u}^*, r_1)$, $\boldsymbol{n} \in \mathcal{E}_n$, $\boldsymbol{\Psi}_v \in \mathcal{E}_v$, $\boldsymbol{\Psi}_i \mapsto F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\boldsymbol{\Psi}_v,\boldsymbol{\Psi}_i)$ is ℓ_{J_i}-Lipschitz over $\mathcal{E}_i$. □

Lemma V.1 follows from [35, Theorem 3.6], and by the compactness of the sets $\mathcal{E}_n$, $\mathcal{E}_v$ and $\mathcal{E}_i$. To state the main result, recall that $\boldsymbol{u}^*$ is a local optimizer of (3). Define $\boldsymbol{\xi}^* := H(\boldsymbol{u}^*;\boldsymbol{s}_l)$, $\boldsymbol{J}_F := \frac{\partial F(\boldsymbol{u},H(\boldsymbol{u};\boldsymbol{s}_l),\boldsymbol{\theta})}{\partial \boldsymbol{u}}\,|_{\boldsymbol{u}=\boldsymbol{u}^*}$, $e_1 := -\lambda_{\max}(\boldsymbol{J}_F)$, and $e_2 := -\lambda_{\min}(\boldsymbol{J}_F)$. Then, from [23], we can write the dynamics as $F(\boldsymbol{u}, H(\boldsymbol{u};\boldsymbol{s}_l),\boldsymbol{\theta}) = \boldsymbol{J}_F(\boldsymbol{u} - \boldsymbol{u}^*) + g(\boldsymbol{u})$, where g satisfies $\|g(\boldsymbol{u})\| \leq L\|\boldsymbol{u} - \boldsymbol{u}^*\|^2$, $\forall \boldsymbol{u} \in \mathcal{B}(\boldsymbol{u}^*, r_2)$, for some $L > 0$ and $r_2 > 0$. Define $r := \min\{r_1, r_2\}$ and $s_{\min}$ as: $s_{\min} = 0$ if $r \geq \frac{e_1}{L}$, and $s_{\min} = 1 - \frac{rL}{e_1}$ if $r \geq \frac{e_1}{L}$. We are ready to state the main result.

Theorem V.2 (*Stability and convergence*)**.** Consider the OPF problem (3) satisfying Assumption 1. Let Assumptions 3–5 hold for the linear model and the training, and let Assumption 6 hold for (11). Let $\boldsymbol{u}(t)$, $t \geq t_0$, be the unique trajectory of (8). Let $\epsilon := \ell_{J_v}E_{J_v} + \ell_{J_i}E_{J_i} + \ell_n\epsilon_n + \epsilon^{\mathsf{NN}}$, and assume that the set $\mathcal{R} := \left\{s : s_{\min} < s \leq 1,\ e_1^{-3}e_2LE < s - s^2\right\}$ is not empty. Then, for any $s \in \mathcal{R}$, it holds that

$$
\|\boldsymbol{u}(t) - \boldsymbol{u}^*\| \leq \sqrt{\frac{e_2}{e_1}}e^{-e_1\eta s(t-t_0)}\|\boldsymbol{u}(t_0) - \boldsymbol{u}^*\| + \frac{e_2\epsilon}{se_1^2}, \quad (16)
$$

for any $\boldsymbol{u}(t_0)$ such that $\|\boldsymbol{u}(t_0) - \boldsymbol{u}^*\| \leq \sqrt{\frac{e_1}{e_2}}\frac{e_1}{L}(1-s)$. △

From Theorem V.2, one can see that the asymptotic error can be reduced by improving the approximation accuracy of the neural network (i.e., reducing ϵ^{NN}) or by improving the linear approximation (i.e., reducing E_{J_v} and E_{J_i}). Practically, the errors in the voltages and currents (i.e., ϵ_n) are negligible. As a technical detail, the requirement that $\mathcal{R}$ is not empty guarantees that $\boldsymbol{u}(t)$ never exits the region of attraction of $\boldsymbol{u}^*$.

The following result characterizes the feasibility of $\boldsymbol{u}(t)$.

Proposition V.3 (*Practical forward invariance*)**.** Let the conditions in Theorem V.2 be satisfied, and let $\boldsymbol{u}(t)$, $t \geq t_0$, be the unique trajectory of (8). Define the set $\mathcal{S}_e := \mathcal{S}_{e,v} \cap \mathcal{S}_{e,i}$,

$$
\begin{aligned}
\mathcal{S}_{e,v} &:= \{\boldsymbol{u} \in \mathcal{C} : \underline{V}_e \leq |\boldsymbol{v}(\boldsymbol{u};\boldsymbol{s}_l)| \leq \overline{V}_e\}\\
\mathcal{S}_{e,i} &:= \{\boldsymbol{u} \in \mathcal{C} : |\boldsymbol{i}(\boldsymbol{u};\boldsymbol{s}_l)| \leq \overline{I}_e\}
\end{aligned}
$$

where $\underline{V}_e = \underline{V} - \epsilon_v - 2E_v - \beta^{-1}\|\boldsymbol{\Gamma}_v\|\epsilon^{\mathsf{NN}}$, $\overline{V}_e = \overline{V} + \epsilon_v + 2E_v + \beta^{-1}\|\boldsymbol{\Gamma}_v\|\epsilon^{\mathsf{NN}}$, and $\overline{I}_e = \overline{I} + \epsilon_i + 2E_i + \|\boldsymbol{\Gamma}_i\|\epsilon^{\mathsf{NN}}$. Then, the neural network-based algorithm (8) renders a set $\mathcal{S}_s$, with $\mathcal{S} \subseteq \mathcal{S}_s \subseteq \mathcal{S}_e$, forward invariant. △

We note that $\mathcal{S}_s$ in Proposition V.3, is an inflation of the set of feasible voltages and currents $\mathcal{S}$ specified in the AC OPF. Hence the terminology "practical feasibility". Indeed, when these errors are small, the constraint violation is practically negligible. We provide the following remarks:

i) When ϵ_v, ϵ_i, E_v, E_i, and ϵ^{NN} are available or are estimated numerically, the constraints of the original AC OPF (3) can be tightened so that (8) can render the feasible set $\mathcal{S}$ forward invariant.
ii) If $\boldsymbol{u}(t_0) \in \mathcal{S}_s$, then $\boldsymbol{u}(t) \in \mathcal{S}_s$ for all $t \geq t_0$. This implies that the solution offered by (8) is practically feasible for both the online implementation in Figure 1(left) and when the offline procedure in Figure 1(center) is terminated before convergence.
iii) Since $\|\boldsymbol{\Gamma}_v\|$ and $\|\boldsymbol{\Gamma}_i\|$ are generally small (less than 0.0982 in our numerical experiments), the constraint violation due to the neural network approximation is practically negligible.

The proofs of all results are provided in the extended version of this manuscript [42].

VI. Conclusions

We have proposed a solution method for solving the AC OPF where a neural network is used to approximate the solution of a convex QP defining the safe gradient flow. Our approach is shown to lead to both feedback-based online implementations and offline solutions based on power flow computations. Compared to existing methods that rely on neural networks, our algorithm ensures that the DERs' set-points are practically feasible and it ensures convergence to a neighborhood of a strict local optimizer of the AC OPF. These guarantees are important for power systems optimization tasks, as operating limits must be satisfied for a reliable and safe power delivery.

References

[1] D. K. Molzahn, F. Dörfler, H. Sandberg, S. H. Low, S. Chakrabarti, R. Baldick, and J. Lavaei, "A survey of distributed optimization and control algorithms for electric power systems," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 2941–2962, 2017.

[2] L. Gan and S. H. Low, "An online gradient algorithm for optimal power flow on radial networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 625–638, 2016.

[3] E. Dall'Anese and A. Simonetto, "Optimal power flow pursuit," *IEEE Transactions on Smart Grid*, vol. 9, no. 2, pp. 942–952, 2016.

[4] A. Bernstein and E. Dall'Anese, "Real-time feedback-based optimization of distribution grids: A unified approach," *IEEE Transactions on Control of Network Systems*, vol. 6, no. 3, pp. 1197–1209, 2019.

[5] M. Picallo, L. Ortmann, S. Bolognani, and F. Dörfler, "Adaptive real-time grid operation via online feedback optimization with sensitivity estimation," *Electric Power Systems Research*, vol. 212, p. 108405, 2022.

[6] A. Venzke, G. Qu, S. Low, and S. Chatzivasileiadis, "Learning optimal power flow: Worst-case guarantees for neural networks," in *IEEE SmartGridComm*, pp. 1–7, IEEE, 2020.

[7] A. S. Zamzam and K. Baker, "Learning optimal solutions for extremely fast AC optimal power flow," in *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pp. 1–6, IEEE, 2020.

[8] J. A. Taylor, S. V. Dhople, and D. S. Callaway, "Power systems without fuel," *Renewable and Sustainable Energy Reviews*, vol. 57, pp. 1322–1336, 2016.

[9] M. K. Singh, V. Kekatos, and G. B. Giannakis, "Learning to solve the AC-OPF using sensitivity-informed deep neural networks," *IEEE Transactions on Power Systems*, vol. 37, no. 4, pp. 2833–2846, 2021.

[10] R. Nellikkath and S. Chatzivasileiadis, "Physics-informed neural networks for AC optimal power flow," *Electric Power Systems Research*, vol. 212, p. 108412, 2022.

[11] X. Pan, M. Chen, T. Zhao, and S. H. Low, "DeepOPF: A feasibility-optimized deep neural network approach for AC optimal power flow problems," *IEEE Systems Journal*, vol. 17, no. 1, pp. 673–683, 2022.

[12] X. Pan, W. Huang, M. Chen, and S. H. Low, "DeepOPF-AL: Augmented learning for solving AC-OPF problems with a multi-valued load-solution mapping," in *Proceedings of the 14th ACM International Conference on Future Energy Systems*, pp. 42–47, 2023.

[13] S. Park, W. Chen, T. W. Mak, and P. Van Hentenryck, "Compact optimization learning for AC optimal power flow," *IEEE Transactions on Power Systems*, vol. 39, no. 2, pp. 4350–4359, 2023.

[14] F. Fioretto, T. W. Mak, and P. Van Hentenryck, "Predicting ac optimal power flows: Combining deep learning and lagrangian dual methods," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, pp. 630–637, 2020.

[15] Q. Tran, J. Mitra, and N. Nguyen, "Learning model combining of convolutional deep neural network with a self-attention mechanism for AC optimal power flow," *Electric Power Systems Research*, vol. 231, p. 110327, 2024.

[16] K. Baker, "A learning-boosted quasi-Newton method for ac optimal power flow," *arXiv preprint arXiv:2007.06074*, 2020.

[17] K. Chen, S. Bose, and Y. Zhang, "Physics-informed gradient estimation for accelerating deep learning-based AC-OPF," *IEEE Transactions on Industrial Informatics*, 2025.

[18] J. Wang and P. Srikantha, "Fast optimal power flow with guarantees via an unsupervised generative model," *IEEE Transactions on Power Systems*, vol. 38, no. 5, pp. 4593–4604, 2022.

[19] H. F. Hamann *et al.*, "Foundation models for the electric power grid," *Joule*, vol. 8, no. 12, pp. 3245–3258, 2024.

[20] M. Li, S. Kolouri, and J. Mohammadi, "Learning to solve optimization problems with hard linear constraints," *IEEE Access*, vol. 11, pp. 59995–60004, 2023.

[21] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos, "Learning to optimize: Training deep neural networks for interference management," *IEEE Transactions on Signal Processing*, vol. 66, no. 20, pp. 5438–5453, 2018.

[22] F. Zhou, J. Anderson, and S. H. Low, "The optimal power flow operator: Theory and computation," *IEEE Transactions on Control of Network Systems*, vol. 8, no. 2, pp. 1010–1022, 2020.

[23] A. Allibhoy and J. Cortés, "Control barrier function-based design of gradient flows for constrained nonlinear programming," *IEEE Transactions on Automatic Control*, vol. 69, no. 6, 2024.

[24] A. Colot, Y. Chen, B. Cornélusse, J. Cortés, and E. Dall'Anese, "Optimal power flow pursuit via feedback-based safe gradient flow," *IEEE Transactions on Control Systems Technology*, vol. 33, no. 2, pp. 658–670, 2025.

[25] D. Sarajlić and C. Rehtanz, "Low voltage benchmark distribution network models based on publicly available data," in *IEEE PES Innovative Smart Grid Technologies Europe*, 2019.

[26] W. H. Kersting, *Distribution System Modeling and Analysis*. 2nd ed., Boca Raton, FL: CRC Press, 2007.

[27] S. Bolognani and S. Zampieri, "On the existence and linear approximation of the power flow solution in power distribution networks," *IEEE Transactions on Power Systems*, vol. 31, no. 1, pp. 163–172, 2015.

[28] A. Bernstein, C. Wang, E. Dall'Anese, J.-Y. Le Boudec, and C. Zhao, "Load flow in multiphase distribution networks: Existence, uniqueness, non-singularity and linear models," *IEEE Transactions on Power Systems*, vol. 33, no. 6, pp. 5832–5843, 2018.

[29] C. Wang, A. Bernstein, J.-Y. Le Boudec, and M. Paolone, "Existence and uniqueness of load-flow solutions in three-phase distribution networks," *IEEE Transactions on Power Systems*, vol. 32, no. 4, pp. 3319–3320, 2017.

[30] S. Bolognani, R. Carli, G. Cavraro, and S. Zampieri, "Distributed reactive power feedback control for voltage regulation and loss minimization," *IEEE Transactions on Automatic Control*, vol. 60, no. 4, pp. 966–981, 2014.

[31] L. Gan and S. H. Low, "Convex relaxations and linear approximation for optimal power flow in multiphase radial networks," in *Power Systems Computation Conference*, IEEE, 2014.

[32] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *European control conference*, pp. 3420–3431, 2019.

[33] L. Chen and J. W. Simpson-Porco, "A fixed-point algorithm for the ac power flow problem," in *2023 American Control Conference (ACC)*, pp. 4449–4456, IEEE, 2023.

[34] A. Hauswirth, S. Bolognani, G. Hug, and F. Dorfler, "Projected gradient descent on Riemannian manifolds with applications to online power system optimization," in *54th Annual Allerton Conference on Communication, Control, and Computing*, pp. 225–232, Sept 2016.

[35] J. Liu, "Sensitivity analysis in nonlinear programs and variational inequalities via continuous selections," *SIAM Journal on Control and Optimization*, vol. 33, no. 4, pp. 1040–1060, 1995.

[36] A. Angioni, T. Schlösser, F. Ponci, and A. Monti, "Impact of pseudo-measurements from new power profiles on state estimation in low-voltage grids," *IEEE Transactions on Instrumentation and Measurement*, vol. 65, no. 1, pp. 70–77, 2015.

[37] M. Baran and F. Wu, "Optimal capacitor placement on radial distribution systems," *IEEE Transactions on Power Delivery*, vol. 4, no. 1, pp. 725–734, 1989.

[38] A. V. Fiacco, "Sensitivity analysis for nonlinear programming using penalty methods," *Mathematical programming*, vol. 10, no. 1, pp. 287–311, 1976.

[39] A. Hauswirth, S. Bolognani, G. Hug, and F. Dörfler, "Generic existence of unique lagrange multipliers in ac optimal power flow," *IEEE Control Systems Letters*, vol. 2, no. 4, pp. 791–796, 2018.

[40] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[41] Y. Duan, G. Ji, Y. Cai, *et al.*, "Minimum width of leaky-relu neural networks for uniform universal approximation," in *International Conference on Machine Learning*, pp. 19460–19470, PMLR, 2023.

[42] D. Ajeyemi, Y. Chen, A. Colot, J. Cortés, and E. Dall'Anese, "Learning to pursue ac optimal power flow solutions with feasibility guarantees," 2025. Extended version:.

APPENDIX

A. Proof of Theorem V.2

Recall that $\boldsymbol{\nu} := |\boldsymbol{v}(\boldsymbol{u};\boldsymbol{s}_l)| + \boldsymbol{n}_v$, $\boldsymbol{i} := |\boldsymbol{i}(\boldsymbol{u};\boldsymbol{s}_l)| + \boldsymbol{n}_i$, $\boldsymbol{\xi} = (\boldsymbol{\nu}, \boldsymbol{\iota})$; to streamline notation, we will use $|\boldsymbol{v}|$ and $|\boldsymbol{i}|$ to denote the error-free measurements or computations of voltage magnitudes currents magnitudes. Recall that $F(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) = F_{\mathsf{m}}(\boldsymbol{u}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0})$. We express the NN-SGF controller as $\dot{\boldsymbol{u}} = \mathcal{F}^{\mathsf{NN}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})$, where $\boldsymbol{\theta} = (\boldsymbol{\theta}_{u,1}, \ldots, \boldsymbol{\theta}_{u,G}, \underline{V}, \overline{V}, \overline{I})$ contains the constraint parameters of the AC OPF. Rewrite the NN-SGF controller as:

$$\begin{aligned}
\dot{\boldsymbol{u}} &= \eta\, \mathcal{F}^{\mathsf{NN}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) \\
&= \eta\, \underbrace{F_{\mathsf{m}}(\boldsymbol{u}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0})}_{\text{nominal}} \\
&\quad + \eta\, \underbrace{[F_{\mathsf{m}}(\boldsymbol{u}, \boldsymbol{n}, \boldsymbol{\Psi}_v, \boldsymbol{\Psi}_i) - F_{\mathsf{m}}(\boldsymbol{u}, \boldsymbol{n}, \boldsymbol{0}, \boldsymbol{0})]}_{\text{Jacobian error}} \\
&\quad + \eta\, \underbrace{[F_{\mathsf{m}}(\boldsymbol{u}, \boldsymbol{n}, \boldsymbol{0}, \boldsymbol{0}) - F_{\mathsf{m}}(\boldsymbol{u}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0})]}_{\text{measurement error}} \\
&\quad + \eta\, \underbrace{\left[\mathcal{F}^{\mathsf{NN}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) - F_{\mathsf{ln}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})\right]}_{\text{NN training error}}
\end{aligned}$$

where we stress that $F_{\mathsf{m}}(\boldsymbol{u}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0}) = F(\boldsymbol{u}, (|\boldsymbol{v}|, |\boldsymbol{i}|), \boldsymbol{\theta})$ is the nominal controller (7).The NN-SGF controller is thus interpreted as a perturbation of the nominal gradient flow $F_{\mathsf{m}}(\boldsymbol{u}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0})$, which is differentiable at the strict local minimizer $\boldsymbol{u}^*$ (Assumption 2); its Jacobian is defined as

$$\boldsymbol{J}_F := \left.\frac{\partial F_{\mathsf{m}}(\boldsymbol{u}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0})}{\partial \boldsymbol{u}}\right|_{\boldsymbol{u}=\boldsymbol{u}^*},$$

and is negative definite. Let $e_1 := -\lambda_{\max}(\boldsymbol{J}_F)$, $e_2 := -\lambda_{\min}(\boldsymbol{J}_F)$, and define the matrix

$$P := \int_0^\infty e^{\boldsymbol{J}_F^\top \zeta} e^{\boldsymbol{J}_F \zeta}\, d\zeta,$$

which satisfies the Lyapunov equation $P\boldsymbol{J}_F + \boldsymbol{J}_F^\top P = -I$. The matrix P satisfies the bounds:

$$\frac{1}{2e_2}\|\boldsymbol{u} - \boldsymbol{u}^*\|^2 \le (\boldsymbol{u} - \boldsymbol{u}^*)^\top P(\boldsymbol{u} - \boldsymbol{u}^*) \le \frac{1}{2e_1}\|\boldsymbol{u} - \boldsymbol{u}^*\|^2.$$

Define the Lyapunov function

$$V_1(\boldsymbol{u}) := (\boldsymbol{u} - \boldsymbol{u}^*)^\top P(\boldsymbol{u} - \boldsymbol{u}^*).$$

We compute:

$$\begin{aligned}
\dot{V}_1(\boldsymbol{u}) &= 2(\boldsymbol{u} - \boldsymbol{u}^*)^\top P \dot{\boldsymbol{u}} \\
&= 2\eta(\boldsymbol{u} - \boldsymbol{u}^*)^\top P\, \mathcal{F}^{\mathsf{NN}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) \\
&= 2\eta(\boldsymbol{u} - \boldsymbol{u}^*)^\top P\, F_{\mathsf{m}}(\boldsymbol{u}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0}) \\
&\quad + 2\eta(\boldsymbol{u} - \boldsymbol{u}^*)^\top P\, [F_{\mathsf{m}}(\boldsymbol{u}, \boldsymbol{n}, \boldsymbol{\Psi}_v, \boldsymbol{\Psi}_i) - F_{\mathsf{m}}(\boldsymbol{u}, \boldsymbol{n}, \boldsymbol{0}, \boldsymbol{0})] \\
&\quad + 2\eta(\boldsymbol{u} - \boldsymbol{u}^*)^\top P\, [F_{\mathsf{m}}(\boldsymbol{u}, \boldsymbol{n}, \boldsymbol{0}, \boldsymbol{0}) - F_{\mathsf{m}}(\boldsymbol{u}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0})] \\
&\quad + 2\eta(\boldsymbol{u} - \boldsymbol{u}^*)^\top P\, \left[\mathcal{F}^{\mathsf{NN}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) - F_{\mathsf{ln}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})\right].
\end{aligned}$$

Next, we analyze each term. As for the nominal controller, by first-order Taylor expansion [23]:

$$\begin{aligned}
F_{\mathsf{m}}(\boldsymbol{u}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0}) &= F_{\mathsf{m}}(\boldsymbol{u}^*, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0}) \\
&\quad + \left.\frac{\partial F_{\mathsf{m}}(\boldsymbol{u}, \boldsymbol{0}, \boldsymbol{0}, \boldsymbol{0})}{\partial \boldsymbol{u}}\right|_{\boldsymbol{u}=\boldsymbol{u}^*} (\boldsymbol{u} - \boldsymbol{u}^*) + g(\boldsymbol{u}).
\end{aligned}$$

Additionally, one has that $\|g(\boldsymbol{u})\| \leq L\|\boldsymbol{u}-\boldsymbol{u}^*\|^2$, for some $L \geq 0$. Then, $F_{\mathsf{m}}(\boldsymbol{u},\mathbf{0},\mathbf{0},\mathbf{0}) = \boldsymbol{J}_F(\boldsymbol{u}-\boldsymbol{u}^*) + \hat{g}(\boldsymbol{u})$.

The quadratic form evaluates as:

$$\begin{aligned}&2\eta(\boldsymbol{u}-\boldsymbol{u}^*)^\top P F_{\mathsf{m}}(\boldsymbol{u},\mathbf{0},\mathbf{0},\mathbf{0})\\ &= (\boldsymbol{u}-\boldsymbol{u}^*)^\top \left(P\boldsymbol{J}_F + \boldsymbol{J}_F^\top P\right)(\boldsymbol{u}-\boldsymbol{u}^*) + 2\eta(\boldsymbol{u}-\boldsymbol{u}^*)^\top P\hat{g}(\boldsymbol{u}).\end{aligned}$$

Using the Lyapunov identity $P\boldsymbol{J}_F + \boldsymbol{J}_F^\top P = -I$, the bound $\|P\| \leq \frac{1}{2e_1}$, and $\|g(\boldsymbol{u})\| \leq L\|\boldsymbol{u}-\boldsymbol{u}^*\|^2$, we conclude:

$$2\eta(\boldsymbol{u}-\boldsymbol{u}^*)^\top P F_{\mathsf{m}}(\boldsymbol{u},\mathbf{0},\mathbf{0},\mathbf{0}) \leq -\eta\|\boldsymbol{u}-\boldsymbol{u}^*\|^2 + \frac{\eta L}{e_1}\|\boldsymbol{u}-\boldsymbol{u}^*\|^3.$$

We now focus on the term related to the error in the Jacobian. Using the triangle inequality we get:

$$\begin{aligned}&\|F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\boldsymbol{\Psi}_v,\boldsymbol{\Psi}_i) - F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\mathbf{0},\mathbf{0})\|\\ &\quad= \|F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\boldsymbol{\Psi}_v,\boldsymbol{\Psi}_i) - F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\mathbf{0},\boldsymbol{\Psi}_i)\\ &\qquad + F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\mathbf{0},\boldsymbol{\Psi}_i) - F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\mathbf{0},\mathbf{0})\|\\ &\quad\leq \|F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\boldsymbol{\Psi}_v,\boldsymbol{\Psi}_i) - F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\mathbf{0},\boldsymbol{\Psi}_i)\|\\ &\qquad + \|F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\mathbf{0},\boldsymbol{\Psi}_i) - F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\mathbf{0},\mathbf{0})\|.\end{aligned}$$

By Lemma V.1 and Assumption 3, there exist constants ℓ_{J_v}, ℓ_{J_i} such that:

$$\begin{aligned}\|F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\boldsymbol{\Psi}_v,\boldsymbol{\Psi}_i) - F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\mathbf{0},\boldsymbol{\Psi}_i)\| &\leq \ell_{J_v}E_{J_v},\\ \|F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\mathbf{0},\boldsymbol{\Psi}_i) - F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\mathbf{0},\mathbf{0})\| &\leq \ell_{J_i}E_{J_i}.\end{aligned}$$

Hence,

$$\|F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\boldsymbol{\Psi}_v,\boldsymbol{\Psi}_i) - F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\mathbf{0},\mathbf{0})\| \leq \ell_{J_v}E_{J_v} + \ell_{J_i}E_{J_i}.$$

Using the fact that $\|P\| \leq \frac{1}{2e_1}$, one has that:

$$\begin{aligned}&2\eta(\boldsymbol{u}-\boldsymbol{u}^*)^\top P\left(F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\boldsymbol{\Psi}_v,\boldsymbol{\Psi}_i) - F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\mathbf{0},\mathbf{0})\right)\\ &\quad\leq \frac{2\eta}{2e_1}\|F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\boldsymbol{\Psi}_v,\boldsymbol{\Psi}_i) - F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\mathbf{0},\mathbf{0})\|\cdot\|\boldsymbol{u}-\boldsymbol{u}^*\|\\ &\quad= \frac{\eta}{e_1}\left(\ell_{J_v}E_{J_v} + \ell_{J_i}E_{J_i}\right)\cdot\|\boldsymbol{u}-\boldsymbol{u}^*\|.\end{aligned}$$

Next, by Lemma V.1 and Assumption 4, there exists ℓ_n such that

$$\|F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\mathbf{0},\mathbf{0}) - F_{\mathsf{m}}(\boldsymbol{u},\mathbf{0},\mathbf{0},\mathbf{0})\| \leq \ell_n\epsilon_n.$$

Then:

$$\begin{aligned}&2\eta(\boldsymbol{u}-\boldsymbol{u}^*)^\top P\left(F_{\mathsf{m}}(\boldsymbol{u},\boldsymbol{n},\mathbf{0},\mathbf{0}) - F_{\mathsf{m}}(\boldsymbol{u},\mathbf{0},\mathbf{0},\mathbf{0})\right)\\ &\quad\leq \frac{\eta}{e_1}\ell_n\epsilon_n\|\boldsymbol{u}-\boldsymbol{u}^*\|.\end{aligned}$$

Finally, by Assumption 5, the approximation error satisfies

$$\left\|\mathcal{F}^{\mathsf{NN}}(\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{\theta}) - F_{\mathsf{ln}}(\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{\theta})\right\| \leq \epsilon^{\mathsf{NN}}.$$

Hence,

$$2\eta(\boldsymbol{u}-\boldsymbol{u}^*)^\top P\left[\mathcal{F}^{\mathsf{NN}}(\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{\theta}) - F_{\mathsf{ln}}(\boldsymbol{u},\boldsymbol{\xi},\boldsymbol{\theta})\right] \leq \frac{\eta}{e_1}\epsilon^{\mathsf{NN}}\|\boldsymbol{u}-\boldsymbol{u}^*\|.$$

Putting all terms together, we get:

$$\begin{aligned}\dot{V}_1(\boldsymbol{u}) &\leq -\eta\|\boldsymbol{u}-\boldsymbol{u}^*\|^2 + \frac{\eta L}{e_1}\|\boldsymbol{u}-\boldsymbol{u}^*\|^3\\ &\quad+ \frac{\eta}{e_1}\left(\ell_{J_v}E_{J_v} + \ell_{J_i}E_{J_i} + \ell_n\epsilon_n + \epsilon^{\mathsf{NN}}\right)\|\boldsymbol{u}-\boldsymbol{u}^*\|.\end{aligned}$$

We rewrite the inequality by factoring $\|\boldsymbol{u}-\boldsymbol{u}^*\|^2$ from the first two terms:

$$\begin{aligned}\dot{V}_1(\boldsymbol{u}) &\leq \|\boldsymbol{u}-\boldsymbol{u}^*\|^2\left(-\eta + \frac{\eta L}{e_1}\|\boldsymbol{u}-\boldsymbol{u}^*\|\right)\\ &\quad+ \frac{\eta}{e_1}\left(\ell_{J_v}E_{J_v} + \ell_{J_i}E_{J_i} + \ell_n\epsilon_n + \epsilon^{\mathsf{NN}}\right)\|\boldsymbol{u}-\boldsymbol{u}^*\|.\end{aligned}$$

This inequality holds if $\|\boldsymbol{u}-\boldsymbol{u}^*\| \leq \frac{e_1}{L}(1-s)$, for any $s \in (s_{\min}, 1]$. Then, the dominant terms yield:

$$\begin{aligned}\dot{V}_1(\boldsymbol{u}) &\leq -\eta s\|\boldsymbol{u}-\boldsymbol{u}^*\|^2\\ &\quad+ \frac{\eta}{e_1}\left(\ell_{J_v}E_{J_v} + \ell_{J_i}E_{J_i} + \ell_n\epsilon_n + \epsilon^{\mathsf{NN}}\right)\|\boldsymbol{u}-\boldsymbol{u}^*\|.\end{aligned}$$

Define $V_2(\boldsymbol{u}) := \sqrt{V_1(\boldsymbol{u})}$. Then, using the chain rule,

$$\dot{V}_2(\boldsymbol{u}) = \frac{\dot{V}_1(\boldsymbol{u})}{2V_2(\boldsymbol{u})}.$$

Substituting the bound on $\dot{V}_1(\boldsymbol{u})$ yields:

$$\begin{aligned}\dot{V}_2(\boldsymbol{u}) &\leq -e_1\eta s V_2(\boldsymbol{u})\\ &\quad+ \frac{\eta\sqrt{2e_2}}{2e_1}\left(\ell_{J_v}E_{J_v} + \ell_{J_i}E_{J_i} + \ell_n\epsilon_n + \epsilon^{\mathsf{NN}}\right).\end{aligned}$$

Let $b = e_1\eta s$, and define

$$a = \frac{\eta\sqrt{2e_2}}{2e_1}\left(\ell_{J_v}E_{J_v} + \ell_{J_i}E_{J_i} + \ell_n\epsilon_n + \epsilon^{\mathsf{NN}}\right).$$

Then the differential inequality becomes $\dot{V}_2(\boldsymbol{u}) \leq -bV_2(\boldsymbol{u}) + a$, and by Grönwall's inequality:

$$V_2(t) \leq V_2(t_0)e^{-b(t-t_0)} + \frac{a}{b}\left(1 - e^{-b(t-t_0)}\right).$$

Using the bounds $\|\boldsymbol{u}(t)-\boldsymbol{u}^*\| \leq \sqrt{2e_2}V_2(t)$ and $V_2(t_0) \leq \frac{1}{\sqrt{2e_1}}\|\boldsymbol{u}(t_0)-\boldsymbol{u}^*\|$, we obtain:

$$\begin{aligned}\|\boldsymbol{u}(t)-\boldsymbol{u}^*\| &\leq \sqrt{\frac{e_2}{e_1}}\|\boldsymbol{u}(t_0)-\boldsymbol{u}^*\|e^{-b(t-t_0)}\\ &\quad+ \frac{\sqrt{2e_2}}{b}\cdot a\left(1 - e^{-b(t-t_0)}\right).\end{aligned}$$

Define the aggregate error:

$$\epsilon := \ell_{J_v}E_{J_v} + \ell_{J_i}E_{J_i} + \ell_n\epsilon_n + \epsilon^{\mathsf{NN}},$$

By substituting the definitions of a and b and simplifying, we obtain:

$$\frac{\sqrt{2e_2}}{b}\cdot a = \frac{\sqrt{2e_2}}{e_1\eta s}\cdot\frac{\eta\sqrt{2e_2}}{2e_1}\epsilon = \frac{e_2}{e_1^2 s}\epsilon.$$

then the final bound becomes:

$$\begin{aligned}\|\boldsymbol{u}(t)-\boldsymbol{u}^*\| &\leq \sqrt{\frac{e_2}{e_1}}\|\boldsymbol{u}(t_0)-\boldsymbol{u}^*\|e^{-e_1\eta s(t-t_0)}\\ &\quad+ \frac{e_2}{e_1^2 s}\epsilon\left(1 - e^{-e_1\eta s(t-t_0)}\right).\end{aligned}$$

Evaluating the limit as $t \to +\infty$ yields the desired local exponential stability result. △

B. Proof of Theorem V.3

The proof leverages Nagumo's Theorem. Consider the SGF controller $F_{\mathsf{ln}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta})$ in (11); let $\hat{v}_j := |\boldsymbol{\Gamma}_{v,j}\boldsymbol{u} + \bar{v}_j(\boldsymbol{s}_l)|$ denote the linearized voltage magnitude at index $j \in \mathcal{M}$, and let the measurement noise $\boldsymbol{n}_v$ satisfy $\|\boldsymbol{n}_v\| \leq \epsilon_v$, as in Assumption 4. Then, for each $j \in \mathcal{M}$:

$$\begin{aligned} -\boldsymbol{\Gamma}_{v,j}^\top F_{\mathsf{ln}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) &\leq -\beta\left(\underline{V} - |\hat{v}_j| - \epsilon_v - E_v\right), \\ \boldsymbol{\Gamma}_{v,j}^\top F_{\mathsf{ln}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) &\leq -\beta\left(|\hat{v}_j| - (\overline{V} + \epsilon_v + E_v)\right), \end{aligned}$$

where E_v bounds the voltage linearization error. Now consider the NN-SGF:

$$F^{\mathsf{NN}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) = F_{\mathsf{ln}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) + \Delta F, .$$

with $\|\Delta F\| \leq \epsilon^{\mathsf{NN}}$. Next:

$$\begin{aligned} -\boldsymbol{\Gamma}_{v,j}^\top F^{\mathsf{NN}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) &= -\boldsymbol{\Gamma}_{v,j}^\top F_{\mathsf{ln}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) - \boldsymbol{\Gamma}_{v,j}^\top \Delta F \\ &\leq -\beta(\underline{V} - |\hat{v}_j| - \epsilon_v - E_v) + \|\boldsymbol{\Gamma}_{v,j}\| \cdot \epsilon^{\mathsf{NN}}, \\ \boldsymbol{\Gamma}_{v,j}^\top F^{\mathsf{NN}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) &= \boldsymbol{\Gamma}_{v,j}^\top F_{\mathsf{ln}}(\boldsymbol{u}, \boldsymbol{\xi}, \boldsymbol{\theta}) + \boldsymbol{\Gamma}_{v,j}^\top \Delta F \\ &\leq -\beta(|\hat{v}_j| - \overline{V} - \epsilon_v - E_v) + \|\boldsymbol{\Gamma}_{v,j}\| \cdot \epsilon^{\mathsf{NN}}. \end{aligned}$$

To ensure the vector field is inward-pointing at the boundary of the voltage constraint set, we then require:

$$|\hat{v}_j| \in \left[\underline{V} - \epsilon_v - E_v - \frac{\|\boldsymbol{\Gamma}_{v,j}\|\epsilon^{\mathsf{NN}}}{\beta},\ \overline{V} + \epsilon_v + E_v + \frac{\|\boldsymbol{\Gamma}_{v,j}\|\epsilon^{\mathsf{NN}}}{\beta}\right].$$

A similar argument for the current constraints yields:

$$|\hat{\imath}_j| := |\boldsymbol{\Gamma}_{i,j}\boldsymbol{u} + \bar{\imath}_j(\boldsymbol{s}_l)| \leq \overline{I} + \epsilon_i + E_i + \frac{\|\boldsymbol{\Gamma}_{i,j}\|\epsilon^{\mathsf{NN}}}{\beta}.$$

Define the inflated constraint bounds:

$$\begin{aligned} \underline{V}_e &:= \underline{V} - \epsilon_v - E_v - \frac{\|\boldsymbol{\Gamma}_{v,j}\|\epsilon^{\mathsf{NN}}}{\beta}, \\ \overline{V}_e &:= \overline{V} + \epsilon_v + E_v + \frac{\|\boldsymbol{\Gamma}_{v,j}\|\epsilon^{\mathsf{NN}}}{\beta}, \\ \overline{I}_e &:= \overline{I} + \epsilon_i + E_i + \frac{\|\boldsymbol{\Gamma}_{i,j}\|\epsilon^{\mathsf{NN}}}{\beta}. \end{aligned}$$

We define the inflated feasible sets as:

$$\begin{aligned} \mathcal{S}_{s,\hat{v}} &:= \left\{\boldsymbol{u} \in \mathcal{C} : \underline{V}_s \leq |\hat{\boldsymbol{v}}(\boldsymbol{u}; \boldsymbol{s}_l)| \leq \overline{V}_s\right\}, \\ \mathcal{S}_{s,\hat{\imath}} &:= \left\{\boldsymbol{u} \in \mathcal{C} : |\hat{\boldsymbol{\imath}}(\boldsymbol{u}; \boldsymbol{s}_l)| \leq \overline{I}_s\right\}, \\ \mathcal{S}_s &:= \mathcal{S}_{s,\hat{v}} \cap \mathcal{S}_{s,\hat{\imath}}. \end{aligned}$$

Since the NN-SGF vector field is strictly inward-pointing, Nagumo's Theorem implies that $\mathcal{S}_s$ is forward invariant under (8). To conclude the proof, we note that $|\hat{v}_j| - E_v \leq |v_j| \leq |\hat{v}_j| + E_v$ and $|\hat{\imath}_j| \leq |\imath_j| + E_i$, and thus $\mathcal{S}_s \subseteq \mathcal{S}_e$.

$\triangle$