

AC Power Flow Contingency Analysis Using a Single Deep Neural Network

Md Obaidur Rahman, Junjie Qin, and Vassilis Kekatos
Elmore Family School of Electrical and Computer Engineering
Purdue University, West Lafayette, IN 47906
rahma160, jq, kekatos@purdue.edu

Abstract

Contingency analysis using the AC power flow (AC-PF) model is a critical tool for accurate grid security assessment, but its computational burden increases with the number of operating scenarios and outage configurations to evaluate. Recent ML-based approaches typically require outage-specific training data, leading to offline training costs that scale with the number of contingencies. This work proposes a framework that reuses a single ML model trained solely on basecase AC-PF data to estimate post-contingency operating states under arbitrary single-line outages. The proposed approach formulates post-contingency state prediction as a fixed-point iteration. If the ML model is a deep neural network (DNN), we derive sufficient conditions that guarantee convergence and develop semidefinite programming (SDP) formulations to certify these conditions for a given DNN. Numerical tests on the IEEE 118-bus system demonstrate that the proposed SDP formulations are tight, that the certified conditions hold for all tested contingencies, and that the resulting method produces accurate post-contingency state estimates within only a few iterations.

1. Introduction

Contingency analysis is a fundamental tool for ensuring the secure and reliable operation of power systems and for evaluating future grid expansion plans. The need for scalable contingency analysis is growing as rapid load growth, renewable integration, transportation electrification, and large data centers expand the range of operating conditions that must be assessed. Moreover, corrective switching and topology optimization may lead to more dynamic network configurations. Since contingency analysis relies on repeated power-flow computations across many operating scenarios and outage configurations, its computational burden can become substantial. In this context, we propose a novel framework for predicting post-contingency power system states using a single

machine-learning model that can be reused across multiple outage scenarios.

To alleviate the computational burden of AC contingency analysis, operators often rely on sensitivity-based approximations, such as power-transfer and line-outage distribution factors (PTDFs and LODFs) [1, 2]. Numerous works have improved the efficiency of these methods through direct LODF calculations, graph-theoretic formulations, and extensions to multiple-line outages [3, 4, 5]. However, they cannot accurately capture voltage violations, reactive power redistribution, network losses, or apparent power limit violations. To improve accuracy, several works have developed AC sensitivity factors and linearized AC-PF models [6, 7, 8], yet their accuracy deteriorates under topology changes or major shifts in the operating point, limiting their applicability to screening and approximate contingency analysis.

Another line of work seeks to accelerate exact AC contingency analysis. Warm-started Newton–Raphson methods initialize the post-contingency solve using the basecase voltage profile, thereby reducing the number of iterations required for convergence [9]. Other approaches exploit the fact that a single-line outage induces a low-rank modification of the AC-PF Jacobian. Using the matrix inversion lemma, these methods update the inverse of the basecase Jacobian using outage information rather than recomputing the solution from scratch [10, 11]. Holomorphic embedding methods employ analytic continuation to determine whether a post-contingency solution exists and to avoid Newton–Raphson divergence near stressed operating points [12], but their complexity can still be a concern.

These challenges have motivated the development of ML-based surrogates for power-system computations. DNNs and related ML models have been successfully applied to a variety of power-system tasks, such as the optimal power flow, emergency control, and security assessment [13, 14, 15, 16, 17, 18, 19]. For the AC-PF and contingency analysis, physics-informed neural networks incorporate AC-PF residuals, Kirchhoff’s law

constraints, or network topology information during training to improve voltage predictions [20, 21]. Other works train ML models to predict post-contingency voltages, line loadings, and security indices directly from pre-contingency operating conditions [22]. To account for topology changes, graph neural networks explicitly encode buses and transmission lines for contingency prediction and screening [23], while input-convex DNNs have been proposed for reliable $N - k$ contingency screening with guarantees against false negatives [24].

Despite recent progress, existing ML-based approaches for contingency analysis often require outage-specific training data, increasing offline costs for data generation and model training as the number of contingencies grows. To overcome this limitation, we propose an ML-based fixed-point framework for AC contingency analysis. The contributions are threefold: *c1*) We derive a formulation that encodes line outages as input perturbations, enabling a single DNN trained on the basecase topology to be reused across contingencies; *c2*) We establish sufficient conditions guaranteeing convergence of the iterative scheme; and *c3*) We develop SDP formulations for certifying these conditions. Numerical tests on the IEEE 118-bus system demonstrate that the proposed SDPs derive tight bounds, that the certified conditions hold for all tested single-line contingencies, and that the proposed method produces accurate post-contingency state estimates within only a few iterations.

The rest of the paper is organized as follows. Section 2 reviews the AC-PF model and basecase DNN surrogate. Section 3 introduces the fixed-point framework for single-line contingencies. Section 4 derives convergence conditions, while Section 5 develops SDP-based bounds to certify them. Section 6 evaluates the method on the IEEE 118-bus system.

2. ML Surrogate of an AC-PF Solver

This section reviews the AC power flow (AC-PF) problem and explains how a deep neural network (DNN) can be trained to predict the solutions of an AC-PF solver under the basecase power system topology.

An AC power transmission system can be represented by a graph $\mathcal{G} = (\mathcal{N}, \mathcal{L}_0)$, where $\mathcal{N} := \{1, \dots, N\}$ is the set of buses and $\mathcal{L}_0$ is the set of lines or transformers. We use the subscript 0 to indicate that this graph corresponds to the basecase power system topology. The set $\mathcal{N}$ is partitioned into the set of generators or PV buses $\mathcal{V}$; the set of load or PQ buses $\mathcal{P}$; and the reference bus. Zero-injection buses are included in $\mathcal{P}$. For each $n \in \mathcal{N}$, let $v_n = V_n e^{j\theta_n}$

and $s_n = p_n + jq_n$ denote the voltage phasor and complex power injection into bus n . Vectors $\mathbf{v} \in \mathbb{C}^N$ and $\mathbf{s} \in \mathbb{C}^N$ collect the voltage phasors and complex power injections for all buses, respectively.

The notation $\bar{\mathbf{v}} \in \mathbb{C}^N$ denotes the complex conjugate of $\mathbf{v}$, whereas $\tilde{\mathbf{v}}$ collects the real/imaginary parts of $\mathbf{v}$:

$$\tilde{\mathbf{v}} := \begin{bmatrix} \text{Re}(\mathbf{v}) \\ \text{Im}(\mathbf{v}) \end{bmatrix} \in \mathbb{R}^{2N}.$$

The notation $\ell = (m, n)$ means that line $\ell \in \mathcal{L}$ runs between buses m and n . For each line, let y_ℓ denote the series admittance and let y_ℓ^s denote the half-line shunt admittance. The bus admittance matrix corresponding to the basecase topology can be expressed as

$$\mathbf{Y}_0 = \sum_{\ell=(m,n) \in \mathcal{L}_0} y_\ell \mathbf{a}_\ell \mathbf{a}_\ell^\top + y_\ell^s \mathbf{e}_m \mathbf{e}_m^\top + y_\ell^s \mathbf{e}_n \mathbf{e}_n^\top, \quad (1)$$

where $\mathbf{e}_m$ is the m -th column of the identity, and $\mathbf{a}_\ell := \mathbf{e}_m - \mathbf{e}_n$. Given $\mathbf{Y}_0$, the complex power injections across all buses satisfy the AC power flow (AC-PF) equations

$$\mathbf{s} = \mathbf{v} \odot (\bar{\mathbf{Y}}_0 \bar{\mathbf{v}}) \quad (2)$$

where barred symbols denote complex conjugation and $\odot$ denotes entrywise multiplication.

System operators solve the AC-PF problem to compute the power system state. Given two specified quantities for each bus, the AC-PF task amounts to solving the nonlinear equations in (2) to find $\mathbf{v}$. For each PV bus $n \in \mathcal{V}$, we specify (p_n, V_n) . For each PQ bus $n \in \mathcal{P}$, we specify (p_n, q_n) . For the reference bus, we specify V_0 and set $\theta_0 = 0$. Let us stack the specified quantities in vector $\mathbf{c} \in \mathbb{R}^{2N}$. When an AC-PF solution exists and is unique within the operating region of interest, we can define the mapping from the vector of specifications $\mathbf{c}$ to the AC-PF solution

$$G : \mathbb{R}^{2N} \rightarrow \mathbb{R}^{2N}, \quad \tilde{\mathbf{v}} = G(\mathbf{c}). \quad (3)$$

To accomplish various planning and operation tasks, system operators must repeatedly solve the AC-PF problem for numerous scenarios, including diverse loading conditions and generation schedules. To alleviate the computational burden, one obvious approach would be to replace the AC-PF solver with a ML-based surrogate. One can train a deep neural network (DNN) to predict an AC-PF solution once presented with the specification vector at its input.

One can use a standard feed-forward DNN architecture where the input $\mathbf{c}$ is propagated through K layers to approximate the related AC-PF solution $G(\mathbf{c})$ at the DNN output $F_{\mathbf{w}}(\mathbf{c})$ by feeding $\mathbf{x}_0 = \mathbf{c}$:

$$\mathbf{z}_k = \mathbf{W}_k \mathbf{x}_k + \mathbf{b}_k, \quad k = 0, \dots, K-1 \quad (4a)$$

$$\mathbf{x}_{k+1} = \sigma_k(\mathbf{z}_k), \quad k = 0, \dots, K-1 \quad (4b)$$

$$F_{\mathbf{w}}(\mathbf{c}) = \mathbf{W}_K \mathbf{x}_K + \mathbf{b}_K, \quad (4c)$$

where $\mathbf{W}_k$ and $\mathbf{b}_k$ is the weight matrix and bias vector of layer k , and $\sigma_k(\cdot)$ is the vector-valued activation function at layer k . Vector $\mathbf{w}$ collects all the trainable DNN parameters, that is, the vectorized weight matrices and bias vectors across all layers. Mirroring the mapping of the AC-PF solver in (3), let us denote the input/output mapping of the DNN as

$$F_{\mathbf{w}} : \mathbb{R}^{2N} \rightarrow \mathbb{R}^{2N}, \quad \tilde{\mathbf{v}} = F_{\mathbf{w}}(\mathbf{c}). \quad (5)$$

To avoid the computational burden of generating a large labeled dataset through repeated AC-PF solves, the DNN is trained in an unsupervised way. Rather than solving (2), an AC-PF solution can be found by minimizing the squared residual error of the AC-PF equations as [25, 26]

$$\min_{\tilde{\mathbf{v}} \in \mathbb{C}^N} \sum_{m=1}^{2N} \left(\tilde{\mathbf{v}}^\top \mathbf{H}_m \tilde{\mathbf{v}} - c_m \right)^2, \quad (6)$$

where the symmetric matrix $\mathbf{H}_m$ corresponding to specification m can be derived from $\mathbf{Y}_0$. An AC-PF solution yields zero cost because it satisfies $\tilde{\mathbf{v}}^\top \mathbf{H}_m \tilde{\mathbf{v}} = c_m$ for all m , and so it is a minimizer of (6).

Building on (6), the DNN can be trained by minimizing the AC-PF residual errors across S specification scenarios $\{\mathbf{c}^s\}_{s=1}^S$:

$$\frac{1}{S} \sum_{s=1}^S \sum_{m=1}^{2N} \left(F_{\mathbf{w}}^\top(\mathbf{c}^s) \mathbf{H}_m F_{\mathbf{w}}(\mathbf{c}^s) - c_m^s \right)^2,$$

where c_m^s is the m -th entry of vector $\mathbf{c}^s$. In this way, the optimal DNN parameters $\mathbf{w}^*$ can be found without requiring any AC-PF solutions. Once the DNN parameters have been learned, a single forward pass of the DNN for a given $\mathbf{c}$ approximates the basecase AC-PF solution map

$$F(\mathbf{c}) := F_{\mathbf{w}^*}(\mathbf{c}) \simeq G(\mathbf{c}). \quad (7)$$

To unclutter notation, we will often drop the dependence of F on the optimal DNN parameters. Given the computational effort invested in training this DNN, one might wonder whether it could also infer AC-PF solutions under line contingencies. We next explain how to incorporate this DNN into a fixed-point iteration to infer AC-PF solutions under almost every single-line contingency.

3. Proposed Methodology

To ensure grid reliability, operators must evaluate the system state under any single-line contingency. In a system with L lines, contingency analysis entails solving L additional instances of the AC-PF problem. Each instance involves a contingent topology in which a line has been removed from the basecase topology. Under preventive contingency management, the AC-PF specifications remain unaltered between the basecase and the contingent topologies. Under corrective contingency management, however, the operator may change generator setpoints or curtail load to handle contingencies more effectively. Contingency analysis typically ignores the outage of *critical lines*. A line is deemed critical if, once removed, the power system graph becomes disconnected.

As with the basecase topology, the operator may consider using an ML model to replace AC-PF solvers, thereby alleviating the computational burden associated with AC contingency analysis. Unfortunately, the operator should train L additional ML models, one for each of the L contingent topologies. The offline computational overhead exacerbates further if the operator considers double-line contingencies. Leveraging the structure of the AC-PF problem across the basecase and contingent topologies, we propose an iterative scheme that uses a single DNN trained solely on the basecase topology to predict system states for any single-line contingency.

Let contingency ℓ be associated with the outage of line $\ell = (m, n) \in \mathcal{L}$. The buses m and n , which are adjacent to the outaged line, will be henceforth termed *terminal buses* of contingency ℓ . Similar to (2), the AC-PF equations under contingency ℓ read as

$$\mathbf{s} = \mathbf{v} \odot (\bar{\mathbf{Y}}_\ell \bar{\mathbf{v}}). \quad (8)$$

A key observation is that the bus admittance matrix $\bar{\mathbf{Y}}_\ell$ under contingency ℓ relates to $\bar{\mathbf{Y}}_0$ as

$$\mathbf{Y}_\ell = \mathbf{Y}_0 - \Delta_\ell, \quad (9)$$

where $\Delta_\ell := y_\ell \mathbf{a}_\ell \mathbf{a}_\ell^\top + y_\ell^s \mathbf{e}_m \mathbf{e}_m^\top + y_\ell^s \mathbf{e}_n \mathbf{e}_n^\top$. Plugging (9) into (8) and rearranging provides

$$\mathbf{s} + \check{\delta}_\ell(\mathbf{v}) = \mathbf{v} \odot (\bar{\mathbf{Y}}_0 \bar{\mathbf{v}}), \quad (10)$$

where we define the mapping

$$\check{\delta}_\ell(\mathbf{v}) := \mathbf{v} \odot (\bar{\Delta}_\ell \bar{\mathbf{v}}). \quad (11)$$

Equation (10) reveals that the AC-PF equations under contingency ℓ can be thought of as the AC-PF equations under the basecase topology with the complex power

injections properly modified. It is easy to verify that vector $\Delta_\ell \mathbf{v}$ has only two non-zero entries: the entries m and n related to the buses adjacent to the outaged line ℓ . Due to the entrywise multiplication, vector $\tilde{\delta}_\ell(\mathbf{v})$ has non-zero values only at the terminal bus locations m and n . Therefore, we modify only the complex power injections at terminal buses for the associated topology. Although sparse, the modification vector $\tilde{\delta}_\ell(\mathbf{v})$ depends on the unknown system state $\mathbf{v}$, so the model in (10) seems to be of no practical use.

Nonetheless, we can leverage (10) to develop an iterative scheme that finds the AC-PF solution under contingency ℓ using the basecase AC-PF solver:

$$\mathbf{s} + \tilde{\delta}_\ell(\mathbf{v}_t) = \mathbf{v}_{t+1} \odot (\bar{\mathbf{Y}}_0 \bar{\mathbf{v}}_{t+1}). \quad (12)$$

According to (10), each iteration t involves two steps. The first step computes the modified complex power injections on the left-hand side (LHS) of (12) using the most recent state estimate $\mathbf{v}_t$. In the second step, we find $\mathbf{v}_{t+1}$ as the solution of the basecase AC-PF solver fed with the modified AC-PF specifications. If the iterative scheme converges, its equilibrium coincides with the AC-PF solution under contingency ℓ in (8).

We elaborate on the modified injections at terminal buses. If bus $m \in \mathcal{P}$ is a PQ bus, then $\tilde{\delta}_\ell(\mathbf{v}_t)$ modifies both of its specifications. If bus $m \in \mathcal{V}$ is a PV bus, $\tilde{\delta}_\ell(\mathbf{v}_t)$ modifies only its active power injection, leaving the voltage specification intact. If bus m is the reference, no modification is made. Bus n is handled similarly.

The iterative scheme in (12) can be equivalently expressed in terms of the basecase AC-PF mapping as

$$\tilde{\mathbf{v}}_{t+1} = G(\mathbf{c} + \delta_\ell(\tilde{\mathbf{v}}_t)), \quad (13)$$

where $\mathbf{c}$ is the original specifications for the AC-PF problem under contingency ℓ in (8), and the vector $\delta_\ell(\tilde{\mathbf{v}}_t)$ models how these specifications are modified. Specifically, vector $\delta_\ell(\tilde{\mathbf{v}}_t)$ collects the real part of the entries of $\tilde{\delta}_\ell(\mathbf{v}_t)$ corresponding to PQ and PV buses; the imaginary part of the entries of $\tilde{\delta}_\ell(\mathbf{v}_t)$ corresponding to PQ buses; and contains zeros at the entries corresponding to voltage magnitude specifications. Depending on the types of terminal buses, vector $\delta_\ell(\tilde{\mathbf{v}}_t)$ may have four nonzero entries (PQ/PQ); three (PQ/PV); two (PQ/reference or PV/PV); or one (PV/reference). We will denote the number of nonzero entries of $\delta_\ell(\tilde{\mathbf{v}}_t)$ by $N_\ell \in \{1, 2, 3, 4\}$.

Inspired by (13), we propose predicting AC-PF solutions under line outages by replacing the AC-PF solver with the related DNN. The proposed scheme incorporates DNN into the next fixed-point iteration:

$$\tilde{\mathbf{v}}_{t+1} = F(\mathbf{c} + \delta_\ell(\tilde{\mathbf{v}}_t)). \quad (14)$$

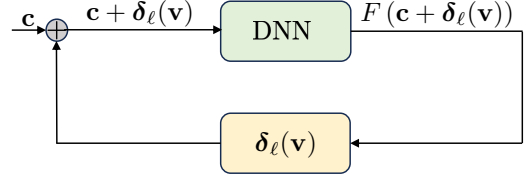


Figure 1. An AC-PF solution under line contingency ℓ can be found through a fixed-point iteration including a single DNN in a feedback loop.

Figure 1 illustrates the scheme. The scheme involves a single DNN, the one trained to solve the basecase AC-PF problem. The advantage of (14) over solving (8) using an AC-PF solver is that the computational complexity can be reduced from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2)$ if the DNN depth does not scale with N and the fixed-point iteration in (14) converges fast. We address these two questions numerically through the tests of Section 6.

Before providing numerical evidence, we develop a computational framework to verify whether (14) is convergent. The proposed framework can certify that the iterative scheme converges. It can also determine its region of convergence and the number of iterations required for convergence.

4. Convergence Certification of the Fixed-Point Iteration

Suppose a system operator has already trained a DNN to predict AC-PF solutions under the basecase topology. Since the DNN parameters $\mathbf{w}$ are fixed after training, we denote the resulting mapping simply by $\tilde{\mathbf{v}} = F(\mathbf{c})$. To predict the AC-PF solution under contingency ℓ , the operator applies the fixed-point iteration shown in (14). A natural question is whether this iteration converges to a fixed point. This section develops a computational framework for addressing this question. Given the power system model and a trained DNN, the proposed framework can be applied offline to certify the convergence of (14) for each contingency ℓ . Once convergence has been certified for a particular contingency, the operator can confidently use (14) to predict post-contingency AC-PF solutions.

Although our analysis ignores shunt admittances by setting $y_\ell^s = 0$ in (1) for all ℓ for simplicity, the tests of Section 6 involve the detailed line models. According to the Banach fixed-point theorem, the iterative scheme in (14) converges to a fixed point if the mapping

$$H_\ell(\tilde{\mathbf{v}}; \mathbf{c}) := F(\mathbf{c} + \delta_\ell(\tilde{\mathbf{v}})) \quad (15)$$

satisfies the following two properties across a set $\mathcal{R}$ of voltage vectors and a set $\mathcal{C}$ of specification vectors $\mathbf{c}$:

- *Self mapping:* If $\tilde{\mathbf{v}} \in \mathcal{R}$, then $H_\ell(\tilde{\mathbf{v}}; \mathbf{c}) \in \mathcal{R}$ for all $\mathbf{c} \in \mathcal{C}$; and
- *Contraction:* There exists $L_{H_\ell} < 1$ so that

$$\|H_\ell(\tilde{\mathbf{v}}_1; \mathbf{c}) - H_\ell(\tilde{\mathbf{v}}_2; \mathbf{c})\| \leq L_{H_\ell} \|\tilde{\mathbf{v}}_1 - \tilde{\mathbf{v}}_2\|$$

for all $\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2 \in \mathcal{R}$ and all $\mathbf{c} \in \mathcal{C}$.

We define sets $\mathcal{R}$ and $\mathcal{C}$ as the Euclidean balls:

$$\mathcal{R} := \{\tilde{\mathbf{v}} : \|\tilde{\mathbf{v}} - \tilde{\mathbf{v}}_0\| \leq R\}, \quad (16a)$$

$$\mathcal{C} := \{\mathbf{c} : \|\mathbf{c} - \mathbf{c}_0\| \leq C\}. \quad (16b)$$

The voltage center is set to be the so-termed *flat voltage profile* $\tilde{\mathbf{v}}_0 := [\mathbf{1}^\top \mathbf{0}^\top]^\top$, corresponding to an unloaded power system with zero power injections ($\mathbf{s}_0 = \mathbf{0}$). The specification center $\mathbf{c}_0$ corresponds to this unloaded power system scenario, so that all power injection specifications in $\mathbf{c}_0$ are set to zero, and all voltage magnitude specifications are set to one. We will assume that the DNN has been trained to perfectly predict the flat voltage profile, i.e., $F(\mathbf{c}_0) = \tilde{\mathbf{v}}_0$.

We first study the self-mapping property of the mapping $H_\ell(\tilde{\mathbf{v}}; \mathbf{c})$. This property is established in two steps. In the first step, we show that if $\mathbf{c} \in \mathcal{C}$ and $\tilde{\mathbf{v}} \in \mathcal{R}$, the contingency-corrected specification vector $\mathbf{c} + \delta_\ell(\tilde{\mathbf{v}})$ belongs to a Euclidean ball $\mathcal{C}'$ with the same center as $\mathcal{C}$ but a larger radius, so that $\mathcal{C} \subset \mathcal{C}'$. This is formalized in Lemma 1; all proofs are in the appendix.

Lemma 1. *If $\mathbf{c} \in \mathcal{C}$ and $\tilde{\mathbf{v}} \in \mathcal{R}$, the modified specification vector lies in the Euclidean ball*

$$\mathbf{c} + \delta_\ell(\tilde{\mathbf{v}}) \in \mathcal{C}' := \{\mathbf{c} : \|\mathbf{c} - \mathbf{c}_0\| \leq C'\} \quad (17)$$

where $C' = C + 2\sqrt{2}\bar{y}R(1 + R)$ and $\bar{y} := \max_\ell |y_\ell|$.

Lemma 1 is practically important because it implies that for the iterations in (14) to work properly, the DNN has to be trained over a larger set of specification scenarios $\mathcal{C}'$. Building on Lemma 1, the next lemma establishes the self-mapping property of $H_\ell(\tilde{\mathbf{v}}; \mathbf{c})$.

Lemma 2. *If the DNN Lipschitz constant L_G satisfies*

$$L_G := \sup_{\mathbf{c} \in \mathcal{C}'} \|\nabla_{\mathbf{c}} F(\mathbf{c})\| \leq \frac{R}{C'}, \quad (18)$$

then $H_\ell(\tilde{\mathbf{v}}; \mathbf{c}) \in \mathcal{R}$ for all $\tilde{\mathbf{v}} \in \mathcal{R}$, $\mathbf{c} \in \mathcal{C}$, and ℓ .

Lemma 2 establishes that $H_\ell(\tilde{\mathbf{v}}; \mathbf{c})$ is a self-mapping provided the DNN sensitivity L_G with respect to its input is sufficiently small across $\mathcal{C}'$. This result is practically important, as it shows that the DNN must be trained and certified to possess a sufficiently small L_G across the enlarged set $\mathcal{C}'$.

Having established the self-mapping property, we proceed with the contraction property. The Lipschitz constant L_{H_ℓ} of the mapping H_ℓ can be computed as

$$L_{H_\ell} = \sup_{\tilde{\mathbf{v}} \in \mathcal{R}, \mathbf{c} \in \mathcal{C}} \|\nabla_{\tilde{\mathbf{v}}} H_\ell(\tilde{\mathbf{v}}; \mathbf{c})\|. \quad (19)$$

From the chain rule, the Jacobian matrix in (19) can be expressed as the product of two $2N \times 2N$ matrices as

$$\nabla_{\tilde{\mathbf{v}}} H_\ell(\tilde{\mathbf{v}}; \mathbf{c}) = \underbrace{\nabla_{\mathbf{c} + \delta_\ell(\tilde{\mathbf{v}})} F(\mathbf{c} + \delta_\ell(\tilde{\mathbf{v}}))}_{:= \mathbf{J}_\ell(\tilde{\mathbf{v}}; \mathbf{c})} \nabla_{\tilde{\mathbf{v}}} \delta_\ell(\tilde{\mathbf{v}}). \quad (20)$$

Recall that vector $\delta_\ell(\tilde{\mathbf{v}})$ has only N_ℓ nonzero entries with $N_\ell \in \{1, 2, 3, 4\}$, and that those entries depend only on the terminal bus voltages. Therefore, the $2N \times 2N$ Jacobian matrix $\nabla_{\tilde{\mathbf{v}}} \delta_\ell(\tilde{\mathbf{v}})$ has only N_ℓ nonzero rows and 4 nonzero columns. Leveraging this structure, the next lemma provides a factorization of $\nabla_{\tilde{\mathbf{v}}} \delta_\ell(\tilde{\mathbf{v}})$, which will be helpful for bounding L_{H_ℓ} .

Lemma 3. *The Jacobian matrix of the mapping $\delta_\ell(\tilde{\mathbf{v}})$ with $\ell = (m, n)$ can be factorized as*

$$\nabla_{\tilde{\mathbf{v}}} \delta_\ell(\tilde{\mathbf{v}}) = \mathbf{U}_\ell \mathbf{K}_\ell(\tilde{\mathbf{v}}) \mathbf{V}_\ell^\top, \quad (21)$$

where $\mathbf{V}_\ell := [\mathbf{e}_m \ \mathbf{e}_n \ \mathbf{e}_{m+N} \ \mathbf{e}_{n+N}]$; $\mathbf{U}_\ell$ is a $2N \times N_\ell$ matrix formed by keeping N_ℓ columns of $\mathbf{V}_\ell$ depending on the types of buses m and n ; and $\mathbf{K}_\ell(\tilde{\mathbf{v}})$ is the $N_\ell \times 4$ Jacobian of the nonzero entries of $\delta_\ell(\tilde{\mathbf{v}})$ with respect to terminal bus voltages satisfying

$$\|\mathbf{K}_\ell(\tilde{\mathbf{v}})\| \leq 4|y_\ell|, \quad \forall \tilde{\mathbf{v}} \in \mathcal{R}.$$

Plugging (20) and (21) into (19) provides

$$\begin{aligned} L_{H_\ell} &= \sup_{\tilde{\mathbf{v}} \in \mathcal{R}, \mathbf{c} \in \mathcal{C}} \|\mathbf{J}_\ell(\tilde{\mathbf{v}}; \mathbf{c}) \mathbf{U}_\ell \mathbf{K}_\ell(\tilde{\mathbf{v}}) \mathbf{V}_\ell^\top\| \\ &\leq 4|y_\ell| \sup_{\tilde{\mathbf{v}} \in \mathcal{R}, \mathbf{c} \in \mathcal{C}} \|\mathbf{J}_\ell(\tilde{\mathbf{v}}; \mathbf{c}) \mathbf{U}_\ell\| \end{aligned} \quad (22)$$

because of the sub-multiplicative norm property and $\|\mathbf{K}_\ell(\tilde{\mathbf{v}}) \mathbf{V}_\ell^\top\| = \|\mathbf{K}_\ell(\tilde{\mathbf{v}})\|$ since $\mathbf{V}_\ell$ is orthonormal.

According to (22), the mapping $H_\ell(\mathbf{c} + \delta_\ell(\tilde{\mathbf{v}}))$ is a contraction for all $\tilde{\mathbf{v}} \in \mathcal{R}$ and $\mathbf{c} \in \mathcal{C}$ if

$$L_{G_\ell} := \sup_{\tilde{\mathbf{v}} \in \mathcal{R}, \mathbf{c} \in \mathcal{C}} \|\mathbf{J}_\ell(\tilde{\mathbf{v}}; \mathbf{c}) \mathbf{U}_\ell\| \leq \frac{1}{4|y_\ell|}. \quad (23)$$

In summary, the mapping $H_\ell(\mathbf{c} + \delta_\ell(\tilde{\mathbf{v}}))$ satisfies the assumptions of the fixed-point theorem if it satisfies (18) and (23). The bound in (18) is on the DNN sensitivity to a general input, while the bound in (23) is on the DNN sensitivity to inputs of particular structure depending on contingency ℓ . Section 5 explains how to certify

whether a given DNN satisfies these bounds by solving SDPs. The numerical tests of Section 6 demonstrate that the DNN sensitivities are sufficiently small for the benchmark systems, so that system operators can confidently use the proposed iterative scheme in (14) to predict post-contingency system states.

5. Bounding DNN Sensitivities via SDPs

We present two SDP formulations to bound the Lipschitz constants L_G and L_{G_ℓ} for a DNN already trained to predict basecase AC-PF solutions. Computing the exact Lipschitz constant of a DNN is known to be NP-hard [27]. We adopt recent works that propose computing upper bounds of DNN Lipschitz constants [28, 29]. Consider the DNN described in (4). Its Lipschitz constant is upper bounded as $L_G \leq \sqrt{\rho}$, where ρ is the optimal value of the ensuing SDP [28]:

$$\begin{aligned} \min \quad & \rho \\ \text{over } \quad & \rho \geq 0, \text{ diagonal } \mathbf{A}_k \succeq 0, k = 0, \dots, K-1 \\ \text{s.to } \quad & \mathbf{M}(\mathbf{Q}_0, \dots, \mathbf{Q}_K) \preceq 0. \end{aligned} \quad (24)$$

Matrix $\mathbf{M}$ is defined as

$$\mathbf{M} := \sum_{k=0}^{K-1} \begin{bmatrix} \mathbf{W}_k \mathbf{E}_k \\ \mathbf{E}_{k+1} \end{bmatrix}^\top \mathbf{Q}_k \begin{bmatrix} \mathbf{W}_k \mathbf{E}_k \\ \mathbf{E}_{k+1} \end{bmatrix} - \begin{bmatrix} \mathbf{E}_0 \\ \mathbf{W}_K \mathbf{E}_K \end{bmatrix}^\top \mathbf{Q}_K \begin{bmatrix} \mathbf{E}_0 \\ \mathbf{W}_K \mathbf{E}_K \end{bmatrix},$$

and each $\mathbf{E}_k$ is a selection matrix so that $\mathbf{x}_k = \mathbf{E}_k \mathbf{x}$ for $k = 0, \dots, K$, if $\mathbf{x}^\top = [\mathbf{x}_0^\top \dots \mathbf{x}_K^\top]$. For $k = 0, \dots, K-1$, the matrix $\mathbf{Q}_k$ is defined as

$$\mathbf{Q}_k := \begin{bmatrix} -\mathbf{A}_k \text{dg}(\boldsymbol{\alpha}_k \odot \boldsymbol{\beta}_k) & \frac{1}{2} \mathbf{A}_k \text{dg}(\boldsymbol{\alpha}_k + \boldsymbol{\beta}_k) \\ \frac{1}{2} \mathbf{A}_k \text{dg}(\boldsymbol{\alpha}_k + \boldsymbol{\beta}_k) & -\mathbf{A}_k \end{bmatrix},$$

where vectors $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ carry lower and upper slope bounds of the activation σ_k . Matrix $\mathbf{Q}_K$ is defined as

$$\mathbf{Q}_K := \begin{bmatrix} \rho \mathbf{I} & 0 \\ 0 & -\mathbf{I} \end{bmatrix}.$$

The SDP depends on the DNN weight matrices $\{\mathbf{W}_k\}_{k=0}^K$, but not the bias vectors $\{\mathbf{b}_k\}_{k=0}^K$. The Lipschitz bound obtained holds for all inputs $\mathbf{c} \in \mathbb{R}^{2N}$. If the DNN has ReLUs as activations, the slope bounds are set as $\boldsymbol{\alpha}_k = \mathbf{0}$ and $\boldsymbol{\beta}_k = \mathbf{1}$ for all k . To improve the conservatism of the bound, subsequent works suggested limiting the ReLU slope bounds if the input lies in a compact set [28, 29]. We briefly summarize the result. If the i -th entry in the pre-activation vector $\mathbf{z}_k$ is bounded

as $z_k^i \in [l_k^i, u_k^i]$, the slope bounds for the associated ReLU are

$$(\alpha_k^i, \beta_k^i) = \begin{cases} (0, 0), & u_k^i \leq 0, \\ (1, 1), & l_k^i \geq 0, \\ \left(0, \frac{u_k^i}{u_k^i - l_k^i}\right), & l_k^i < 0 < u_k^i. \end{cases} \quad (25)$$

Using tighter slope bounds as $\{(\boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)\}_{k=0}^K$ in (24) has been observed to yield significantly tighter bounds on DNN Lipschitz constants [28, 29]. Algorithm 1 of [28] provides a computationally-efficient way for computing lower and upper bounds on the entries of $\mathbf{z}_k$, recursively from the input to the output layer. In our setting, the DNN input lies in the Euclidean ball $\mathcal{C}'$. If $\mathbf{w}_0^i$ is the i -th row of $\mathbf{W}_0$ and b_0^i the i -th entry of $\mathbf{b}_0$, the upper/lower bounds on the i -th entry of $\mathbf{z}_0 = \mathbf{W}_0 \mathbf{c} + \mathbf{b}_0$ can be found in closed form as $\mathbf{c}_0^\top \mathbf{w}_0^i + b_0^i \pm \mathcal{C}' \|\mathbf{w}_0^i\|$.

Switching to certifying the contraction property, we would like to compute an upper bound on L_{G_ℓ} and evaluate the condition in (23). Different from L_G , the constant L_{G_ℓ} depends on the partial sensitivity matrix $\nabla_{\mathbf{c}} F(\mathbf{c}) \mathbf{U}_\ell$, which has only the N_ℓ columns of $\nabla_{\mathbf{c}} F(\mathbf{c})$ associated with contingency ℓ . Interestingly, constant L_{G_ℓ} can be bounded via a neat modification of the SDP.

We will first modify (24) to compute L_{G_ℓ} over $\mathbf{c} \in \mathbb{R}^{2N}$ for generic slope bounds $\boldsymbol{\alpha}_k = \mathbf{0}$ and $\boldsymbol{\beta}_k = \mathbf{1}$. To this end, partition the contribution of the N_ℓ relevant entries of $\mathbf{c}$ on $\mathbf{z}_0$ and that of the remaining entries:

$$\mathbf{z}_0 = \mathbf{W}_0 \mathbf{P}_\ell \mathbf{c} + \mathbf{W}_0 \mathbf{P}_\ell^\perp \mathbf{c} + \mathbf{b}_0, \quad (26)$$

where $\mathbf{P}_\ell := \mathbf{U}_\ell \mathbf{U}_\ell^\top$ and $\mathbf{P}_\ell^\perp := \mathbf{I}_{2N} - \mathbf{P}_\ell$. To compute partial sensitivities, we vary $\mathbf{U}_\ell^\top \mathbf{c}$ and treat $\mathbf{W}_0 \mathbf{P}_\ell^\perp \mathbf{c} + \mathbf{b}_0$ as a modified bias vector. The original SDP in (24) does not depend on biases. Thus, we can upper bound $\sup_{\mathbf{c} \in \mathbb{R}^{2N}} \|\nabla_{\mathbf{c}} F(\mathbf{c}) \mathbf{U}_\ell\|$ using a modified SDP where $\mathbf{W}_0$ has been replaced by $\mathbf{W}_0 \mathbf{U}_\ell$. As with total sensitivities, this bound may be loose.

To tighten the bound on L_{G_ℓ} , we tighten the activation slope bounds in the modified SDP. One option would be to compute upper/lower bounds for each entry of $\mathbf{z}_0$ over $\mathbf{c} \in \mathcal{C}'$, as we did with total sensitivities. This approach does not utilize the structure $\mathbf{c} + \boldsymbol{\delta}_\ell(\tilde{\mathbf{v}})$ of the DNN input with $\mathbf{c} \in \mathcal{C}$ and $\tilde{\mathbf{v}} \in \mathcal{R}$. In the proof of Lemma 2, we showed that if $\tilde{\mathbf{v}} \in \mathcal{R}$, then $\|\boldsymbol{\delta}_\ell(\tilde{\mathbf{v}})\| \leq C_\ell := 2\sqrt{2}|y_\ell|R(1-R)$. Then, the i -th entry of $\mathbf{z}_0$ can be upper bounded by the optimal cost

$$u_0^i = \max_{\mathbf{c}, \boldsymbol{\delta}_\ell} (\mathbf{c} + \boldsymbol{\delta}_\ell)^\top \mathbf{w}_0^i + b_0^i \quad (27)$$

$$\text{s.to } \|\mathbf{c} - \mathbf{c}_0\| \leq C \text{ and } \|\boldsymbol{\delta}_\ell\| \leq C_\ell$$

for $i = 1, \dots, 2N$. The problem is separable over $\mathbf{c}$ and δ_ℓ . The optimal cost can be found to be

$$u_0^i = \mathbf{c}_0^\top \mathbf{w}_0^i + b_0^i + (C + C_\ell) \|\mathbf{w}_0^i\|.$$

By swapping maximization with minimization, the lower bound on z_0^i can be found as $l_0^i = \mathbf{c}_0^\top \mathbf{w}_0^i + b_0^i - (C + C_\ell) \|\mathbf{w}_0^i\|$.

In a nutshell, to compute an upper bound on L_G , we use the SDP in (24) with tightened slope bounds for $\mathbf{c} \in \mathcal{C}'$. To compute an upper bound on L_{G_ℓ} , we solve (24) for each ℓ with $\mathbf{W}_0$ replaced by $\mathbf{W}_0 \mathbf{U}_\ell$ and tightened activation slope bounds over $\mathbf{c} \in \mathcal{C}$ and $\|\delta_\ell\| \leq C_\ell$. Once the upper bounds on L_G and L_{G_ℓ} have been computed, we can certify whether the DNN satisfies conditions (18) and (23).

6. Numerical Tests

We evaluated the proposed AC contingency analysis on the IEEE 118-bus system. To assess its scalability, we also evaluated the proposed method on the larger 6,717-bus Texas7k system, with results reported in [30]. Among its 186 lines, we considered the 177 non-critical lines as candidate contingencies. We trained a generic fully connected DNN with two hidden layers, each containing 118 neurons, resulting in a number of trainable parameters that scales as N^2 . We generated 800 training and 200 testing basecase scenarios by perturbing the nominal specifications with zero-mean Gaussian noise. For PQ specifications, the noise standard deviation was 20% during training and 5% during testing. For voltage specifications, it was 0.1 pu and 0.05 pu, respectively. The DNN was trained using Adam with a rate of 4×10^{-5} , and training was terminated when the gradient norm fell below 10^{-3} .

The first test assessed whether the fixed-point iterations satisfy the convergence conditions. Given the trained DNN, we computed the constant L_G using the SDP in (24) and the constants $L_{H_\ell} = 4|y_\ell|L_{G_\ell}$ for all ℓ using the modified SDP. We set the voltage set radius to $\mathcal{R} = 0.5$, and determined the specification set radius from the testing set used to train the DNN. We observed that $L_G \leq L_{H_\ell}$ for all ℓ , so the subsequent analysis relies solely on L_{H_ℓ} . With the exception of three lines, the contraction condition $L_{H_\ell} \leq 1$ was satisfied. Nonetheless, the fixed-point iteration numerically converged even for those three lines. This is because the contraction condition is only sufficient, and the SDP bounds can be conservative.

The second test examined the relationship between the empirical convergence rate and the Lipschitz constant L_{H_ℓ} across contingencies. Figure 2 shows the Euclidean distance between $\tilde{\mathbf{v}}_t$ and the

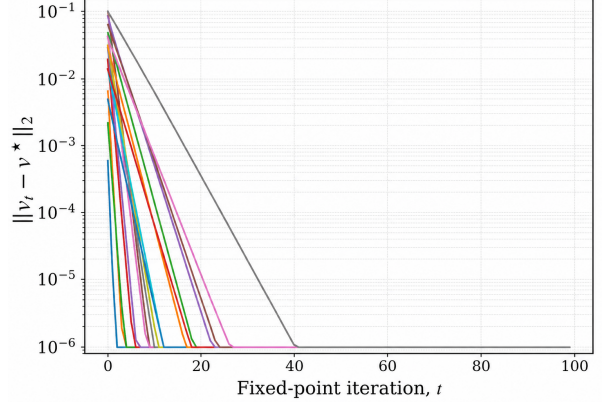


Figure 2. Mean Euclidean distance between the fixed-point voltage estimates and their equilibrium for selected contingencies across scenarios. For all lines, the method converges within 50 iterations.

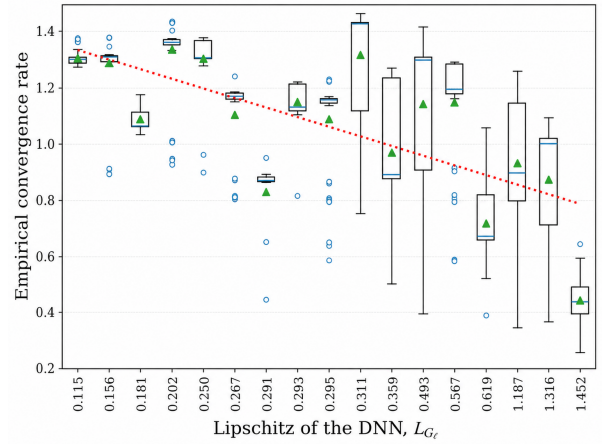


Figure 3. Actual convergence rate of the iterative scheme as a function of the Lipschitz constant L_{H_ℓ} .

equilibrium point for 17 selected contingencies at ranks $\{1, 10, 20, \dots, 160, 177\}$. The method converges within 50 iterations for all contingencies. Figure 3 plots the mean empirical convergence rate across scenarios for the same contingencies as a function of L_{H_ℓ} . The convergence rate decreases with the Lipschitz constant, consistent with the contraction analysis in (23). Variations in the empirical rates arise because the slope is estimated over a finite number of iterations and depends on the initial distance from equilibrium.

The third test assessed the accuracy of the predicted post-contingency states by evaluating the normalized mean squared error (NMSE) between the actual and predicted voltage vectors, and the normalized mean absolute error (NMAE) between the actual AC-PF specifications and those obtained by substituting the predicted voltage states into the AC-PF equations.

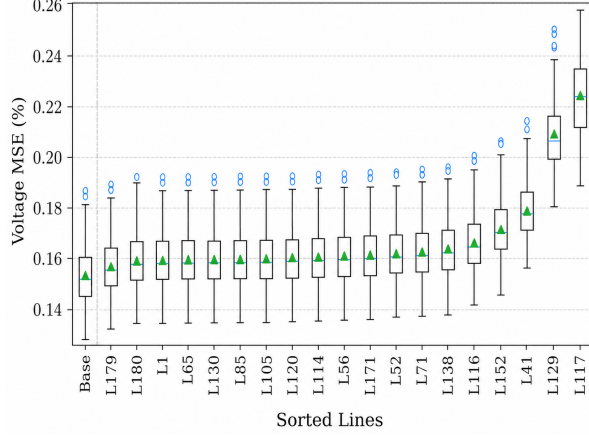


Figure 4. NMSE between actual and predicted voltages for 17 contingencies.

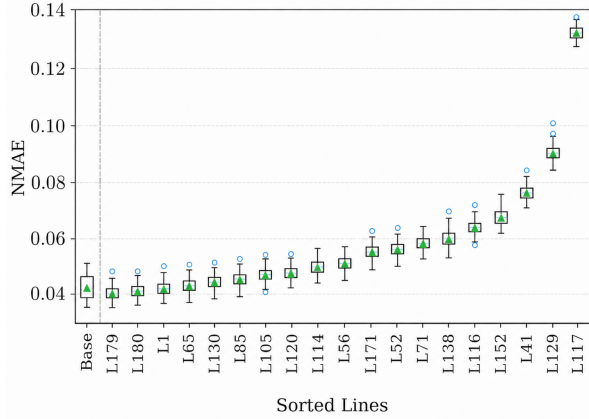


Figure 5. NMAE between actual and predicted PF specifications across 17 contingencies.

Figures 5 and 4 show box plots of these errors for the same selected line contingencies across the 200 testing scenarios, together with the corresponding errors under the basecase topology. For contingencies with smaller Lipschitz values, both metrics remain close to their basecase levels, with noticeable degradation occurring primarily for lines near the end of the Lipschitz ordering.

The final test compared the proposed method’s computational cost with conventional contingency analysis approaches, including NR with flat and warm starts, Jacobian factorization [11], and a DC-based solver. We conducted the experiments on a RunPod terminal equipped with an AMD EPYC 9355 32-Core processor and an NVIDIA RTX PRO 6000 Blackwell GPU. We ran the proposed method on the NVIDIA RTX PRO 6000 Blackwell GPU and all other methods on a CPU, since no GPU implementation is available.

Table 1 compares the total runtime and mean runtime per instance across all contingencies and testing

Table 1. Runtime comparison over 17,700 instances.

Method	Total (s)	Per instance (ms)
Proposed method	3.9369	0.2244
NR flat	9.5709	0.5407
DC model	0.0103	0.0006
NR warm start	8.7270	0.4931
Jacobian factorization	4.6770	0.2642

scenarios. Only the DC-based solver achieves lower runtimes than the proposed method, yet at the expense of linearization inaccuracies. Figure 2 explains the computational advantage of the proposed method. With most contingencies converging in about 30 iterations, our method’s total complexity is $O(N^2)$ in these tests, compared with $O(N^3)$ per iteration for dense NR. Sparse NR implementations can scale better, typically as $O(N^{1.7})$ – $O(N^{2.5})$, while the proposed DNN is highly parallelizable. Its $O(N^2)$ complexity could be reduced further using topology-aware architectures such as GNNs.

7. Conclusions and Future Work

This work has proposed a fixed-point formulation that enables a single DNN, trained solely on basecase AC-PF data, to predict AC-PF solutions for arbitrary single-line contingencies. To guarantee convergence of the fixed-point iterations, we established sufficient self-mapping and contraction conditions and developed SDP formulations to certify that the required DNN sensitivity bounds meet these conditions. Numerical tests on the IEEE 118-bus system validate both the convergence and the prediction accuracy of the proposed method, as the SDP-certified bounds satisfy the self-mapping and contraction conditions for almost all tested cases. The results also show an inverse relationship between the convergence rate and the DNN sensitivity. For many contingencies, the prediction errors remain close to the basecase performance, demonstrating that the basecase DNN can be effectively reused in fixed-point iteration. One interesting direction for future work is to extend the formulation to multi-line contingencies and topology reconfiguration. Another key question is whether the exact AC-PF fixed-point map converges. This would clarify whether the observed convergence behavior comes from the AC-PF formulation or from the mapping of the trained DNN.

8. Appendix

Proof of Lemma 1. If $\mathbf{c} \in \mathcal{C}$ and $\tilde{\mathbf{v}} \in \mathcal{R}$, we get that

$$\|\mathbf{c} + \delta_\ell(\tilde{\mathbf{v}}) - \mathbf{c}_0\| \leq \|\mathbf{c} - \mathbf{c}_0\| + \|\delta_\ell(\tilde{\mathbf{v}})\|$$

$$\leq C + \max_{\tilde{\mathbf{v}} \in \mathcal{R}} \|\delta_\ell(\tilde{\mathbf{v}})\|.$$

It suffices to upper bound $\max_{\tilde{\mathbf{v}} \in \mathcal{R}} \|\delta_\ell(\tilde{\mathbf{v}})\|$ for all ℓ . If we ignore the shunt admittance in (11), we get

$$\tilde{\delta}_\ell(\mathbf{v}) = \bar{y}_\ell (\bar{v}_m \bar{v}_n) \mathbf{v} \odot (\mathbf{e}_m - \mathbf{e}_n). \quad (28)$$

Recall that $\delta_\ell(\tilde{\mathbf{v}})$ is obtained by taking the real values of $\tilde{\delta}_\ell(\mathbf{v})$ related to PQ and PV terminal buses, and the imaginary values of $\tilde{\delta}_\ell(\mathbf{v})$ related to PQ terminal buses. Because $\delta_\ell(\tilde{\mathbf{v}})$ carries a subset or all the values of $\tilde{\delta}_\ell(\mathbf{v})$, we get $\|\delta_\ell(\tilde{\mathbf{v}})\| \leq \|\tilde{\delta}_\ell(\mathbf{v})\|$. From (28), we obtain

$$\|\tilde{\delta}_\ell(\mathbf{v})\| = |y_\ell| \cdot |v_m - v_n| \cdot \sqrt{|v_m|^2 + |v_n|^2}. \quad (29)$$

If $\tilde{\mathbf{v}} \in \mathcal{R}$, then $\|\mathbf{v} - \mathbf{1}\| \leq R$ and so $\|\mathbf{v} - \mathbf{1}\|_\infty \leq R$. This implies that $|v_m| \leq (1 + R)$, $|v_n| \leq (1 + R)$, and $|v_m - v_n| \leq 2R$ for all entries of $\mathbf{v}$. Hence, the norm in (29) is upper bounded by $2\sqrt{2}|y_\ell|R(1 + R)$ for all ℓ . The radius C' is a universal bound for all ℓ . $\square$

Proof of Lemma 2. The goal is to show that $\|H_\ell(\tilde{\mathbf{v}}; \mathbf{c}) - \tilde{\mathbf{v}}_0\| \leq R$ for all $\tilde{\mathbf{v}} \in \mathcal{R}$ and $\mathbf{c} \in \mathcal{C}$. To this end, we have

$$\begin{aligned} \|H_\ell(\tilde{\mathbf{v}}; \mathbf{c}) - \tilde{\mathbf{v}}_0\| &= \|F(\mathbf{c} + \delta_\ell(\tilde{\mathbf{v}})) - F(\mathbf{c}_0)\| \\ &\leq L_G \|\mathbf{c} + \delta_\ell(\tilde{\mathbf{v}}) - \mathbf{c}_0\| \leq L_G C'. \end{aligned}$$

The equality follows from the definition of H_ℓ in (15) and assuming that the DNN has been trained to predict $\tilde{\mathbf{v}}_0$ when presented $\mathbf{c}_0$. The first inequality follows from the DNN Lipschitz constant definition over set $\mathcal{C}'$. The second inequality follows from Lemma 2. The condition in (18) ensures that $L_G C' \leq R$. $\square$

Proof of Lemma 3. We first consider the case where both terminal buses are PQ buses. In this case, vector $\delta_\ell(\tilde{\mathbf{v}})$ has four nonzero entries, corresponding to the two PQ specifications per terminal bus:

$$\delta_{\ell,m}(\tilde{\mathbf{v}}) = \text{Re}\{\bar{y}_\ell v_m(\bar{v}_m - \bar{v}_n)\} \quad (30a)$$

$$\delta_{\ell,n}(\tilde{\mathbf{v}}) = -\text{Re}\{\bar{y}_\ell v_n(\bar{v}_m - \bar{v}_n)\} \quad (30b)$$

$$\delta_{\ell,m+N}(\tilde{\mathbf{v}}) = \text{Im}\{\bar{y}_\ell v_m(\bar{v}_m - \bar{v}_n)\} \quad (30c)$$

$$\delta_{\ell,n+N}(\tilde{\mathbf{v}}) = -\text{Im}\{\bar{y}_\ell v_n(\bar{v}_m - \bar{v}_n)\}. \quad (30d)$$

The four functions are quadratic functions of only four entries of $\tilde{\mathbf{v}}$, which can be collected in vector $\tilde{\mathbf{v}}_\ell := [\tilde{v}_m \ \tilde{v}_n \ \tilde{v}_{m+N} \ \tilde{v}_{n+N}]^\top$.

Given its sparsity pattern, matrix $\nabla_{\tilde{\mathbf{v}}} \delta_\ell(\tilde{\mathbf{v}})$ can be factorized as in (21) with $\mathbf{U}_\ell = \mathbf{V}_\ell$ and $\mathbf{K}_\ell(\tilde{\mathbf{v}})$ being the Jacobian of the four quadratic functions. Since the functions are quadratic, the Jacobian is linear in $\tilde{\mathbf{v}}_\ell$:

$$\mathbf{K}_\ell(\tilde{\mathbf{v}}) = \sum_{i=1}^4 \tilde{v}_{\ell,i} \mathbf{A}_\ell^i, \quad (31)$$

where matrices $\mathbf{A}_\ell^i$ depend on $y_\ell = g_\ell + jb_\ell$ as follows

$$\mathbf{A}_\ell^1 = \begin{bmatrix} 2g_\ell & -g_\ell & 0 & b_\ell \\ 0 & -g_\ell & 0 & -b_\ell \\ -2b_\ell & b_\ell & 0 & g_\ell \\ 0 & b_\ell & 0 & -g_\ell \end{bmatrix},$$

$$\mathbf{A}_\ell^2 = \begin{bmatrix} -g_\ell & 0 & -b_\ell & 0 \\ -g_\ell & 2g_\ell & b_\ell & 0 \\ b_\ell & 0 & -g_\ell & 0 \\ b_\ell & -2b_\ell & g_\ell & 0 \end{bmatrix},$$

$$\mathbf{A}_\ell^3 = \begin{bmatrix} 0 & -b_\ell & 2g_\ell & -g_\ell \\ 0 & b_\ell & 0 & -g_\ell \\ 0 & -g_\ell & -2b_\ell & b_\ell \\ 0 & g_\ell & 0 & b_\ell \end{bmatrix},$$

$$\mathbf{A}_\ell^4 = \begin{bmatrix} b_\ell & 0 & -g_\ell & 0 \\ -b_\ell & 0 & -g_\ell & 2g_\ell \\ g_\ell & 0 & b_\ell & 0 \\ -g_\ell & 0 & b_\ell & -2b_\ell \end{bmatrix}.$$

Matrix $\mathbf{K}_\ell(\tilde{\mathbf{v}})$ can be alternatively expressed as

$$\mathbf{K}_\ell(\tilde{\mathbf{v}}) = \mathbf{A}_\ell (\tilde{\mathbf{v}}_\ell \otimes \mathbf{I}_4), \quad (32)$$

where $\mathbf{A}_\ell := [\mathbf{A}_\ell^1 \ \mathbf{A}_\ell^2 \ \mathbf{A}_\ell^3 \ \mathbf{A}_\ell^4]$ and $\otimes$ is the Kronecker product. From the sub-multiplicative property of the spectral norm, it follows that $\|\mathbf{K}_\ell(\tilde{\mathbf{v}})\| \leq \|\mathbf{A}_\ell\| \cdot \|\tilde{\mathbf{v}}_\ell \otimes \mathbf{I}_4\| = \|\mathbf{A}_\ell\| \cdot \|\tilde{\mathbf{v}}_\ell\|$.

Since $\tilde{\mathbf{v}} \in \mathcal{R}$, we get $\|\mathbf{v} - \mathbf{1}\| = \|\tilde{\mathbf{v}} - \tilde{\mathbf{v}}_0\| \leq R$ and $\|\mathbf{v} - \mathbf{1}\|_\infty \leq \|\mathbf{v} - \mathbf{1}\| \leq R$ so that $|v_m| \leq 1 + R$ and $|v_n| \leq 1 + R$. It hence follows that $\|\tilde{\mathbf{v}}_\ell\| = \|\mathbf{v}_\ell\| \leq \sqrt{2}\|\mathbf{v}_\ell\|_\infty \leq \sqrt{2}(1 + R)$. Regarding $\mathbf{A}_\ell$, after some algebraic manipulations, it can be shown that the largest eigenvalue of $\mathbf{A}_\ell \mathbf{A}_\ell^\top$ is $16|y_\ell|^2$, so that $\|\mathbf{A}_\ell\| = 4|y_\ell|$.

Other combinations of terminal bus types can be handled by selecting a subset of rows from $\mathbf{K}_\ell(\tilde{\mathbf{v}})$ and by forming $\mathbf{U}_\ell$ by selecting the corresponding columns from $\mathbf{V}_\ell$. For example, if $m \in \mathcal{P}$ and $n \in \mathcal{V}$, the $(n + N)$ -th entry of $\delta_\ell(\tilde{\mathbf{v}})$ is zero, and so the quadratic function in (30d) is ignored. In this case, we have $N_\ell = 3$, $\mathbf{U}_\ell = [\mathbf{e}_m \ \mathbf{e}_n \ \mathbf{e}_{m+N}]$, and $\mathbf{K}_\ell(\tilde{\mathbf{v}})$ is obtained from (31) by dropping the fourth row from each $\mathbf{A}_\ell^i$. Because dropping rows from a matrix cannot increase its spectral norm, the bound $\|\mathbf{K}_\ell(\tilde{\mathbf{v}})\| \leq 4|y_\ell|$ remains valid. In general, if a terminal bus is a PQ bus, it contributes two rows to $\mathbf{K}_\ell(\tilde{\mathbf{v}})$; if it is a PV bus, it contributes one row; and if it is the reference bus, it contributes no row to $\mathbf{K}_\ell(\tilde{\mathbf{v}})$. $\square$

References

- [1] P. W. Sauer, "On the formulation of power distribution factors for linear load flow methods," *IEEE Trans. Power App. Syst.*, vol. PAS-100, no. 2, pp. 764–770, Feb. 1981.

- [2] A. J. Wood, B. F. Wollenberg, and G. B. Sheblé, *Power Generation, Operation, and Control*, 3rd ed. Hoboken, NJ, USA: Wiley, Dec. 2013.
- [3] J. Guo, Y. Fu, Z. Li, and M. Shahidehpour, "Direct calculation of line outage distribution factors," *IEEE Trans. Power Syst.*, vol. 24, no. 3, pp. 1633–1634, Aug. 2009.
- [4] A. F. Castelli and J. D. Lara, "Improved evaluation of large network matrices for linear power flow within optimization problems," in *Proc. IEEE Power & Energy Society General Meeting*, Seattle, WA, Jul. 2024, pp. 1–5.
- [5] T. Güler, G. Gross, and M. Liu, "Generalized line outage distribution factors," *IEEE Trans. Power Syst.*, vol. 22, no. 2, pp. 879–881, May 2007.
- [6] H. Qu, L. Roald, and G. Andersson, "Uncertainty margins for probabilistic AC security assessment," in *PowerTech*, Eindhoven, Netherlands, Jun. 2015, pp. 1–6.
- [7] S. V. Dhople, S. S. Guggilam, and Y. C. Chen, "Linear approximations to AC power flow in rectangular coordinates," in *Proc. Allerton Conf.*, Monticello, IL, Sep. 2015, pp. 211–217.
- [8] C. Coffrin and P. Van Hentenryck, "A linear-programming approximation of AC power flows," *INFORMS Journal on Computing*, vol. 26, no. 4, pp. 718–734, Nov. 2014.
- [9] J. Yu, Z. Li, J. Zhang, X. Bai, H. Ge, J. Zheng, and Q. H. Wu, "Efficient contingency analysis of power systems using linear power flow with generalized warm-start compensation," *International Journal of Electrical Power & Energy Systems*, vol. 156, p. 109692, Mar. 2024.
- [10] Y.-H. Yeung, A. Pothén, M. Halappanavar, and Z. Huang, "AMPS: An augmented matrix formulation for principal submatrix updates with application to power grids," *SIAM Journal on Scientific Computing*, vol. 39, no. 5, pp. S809–S827, Oct. 2017.
- [11] K. L. Lo and S. P. Zhu, "A newton-like method for line outage simulation," *IEEE Proceedings - Generation, Transmission and Distribution*, vol. 151, no. 6, pp. 706–710, Mar. 2004.
- [12] R. Yao, F. Qiu, and K. Sun, "Contingency analysis based on partitioned and parallel holomorphic embedding," *IEEE Trans. Power Syst.*, vol. 37, no. 2, pp. 1452–1462, Mar. 2022.
- [13] A. S. Zamzam and K. Baker, "Learning optimal solutions for extremely fast AC optimal power flow," in *SmartGridComm*, Tempe, AZ, Nov. 2020, pp. 1–6.
- [14] X. Pan, T. Zhao, M. Chen, and S. Zhang, "DeepOPF: A deep neural network approach for security-constrained DC optimal power flow," *IEEE Trans. Power Syst.*, vol. 36, no. 3, pp. 1725–1735, May 2021.
- [15] M. K. Singh, V. Kekatos, and G. B. Giannakis, "Learning to solve the AC-OPF using sensitivity-informed deep neural networks," *IEEE Trans. Power Syst.*, vol. 37, no. 4, pp. 2833–2846, Jul. 2022.
- [16] Y. Zhou, J. Park, and H. Zhu, "Scalable learning for optimal load shedding under power grid emergency operations," in *Proc. IEEE Power & Energy Society General Meeting*, Denver, CO, Jul. 2022, pp. 1–5.
- [17] S. Gupta, V. Kekatos, and S. Chatzivasileiadis, "Optimal design of volt/VAR control rules of inverters using deep learning," *IEEE Trans. Smart Grid*, vol. 15, no. 5, pp. 4731–4743, Sep. 2024.
- [18] W. Wang, N. Yu, J. Shi, and Y. Gao, "Safe off-policy deep reinforcement learning algorithm for volt/VAR control in power distribution systems," *IEEE Trans. Smart Grid*, vol. 11, no. 4, pp. 3008–3018, Jul. 2020.
- [19] A. Kody, J. Pabon, and D. K. Molzahn, "Learning to solve the $N - k$ interdiction problem considering load variability," in *Hawaii International Conference on System Sciences*, Lahaina, HI, Jan. 2023, pp. 2723–2732.
- [20] X. Hu, H. Hu, S. Verma, and Z.-L. Zhang, "Physics-guided deep neural networks for power flow analysis," *IEEE Trans. Power Syst.*, vol. 36, no. 3, pp. 2082–2092, May 2021.
- [21] V. Eeckhout, H. Fani, M. U. Hashmi, and G. Deconinck, "Improved physics-informed neural network based AC power flow for distribution networks," in *Proc. IEEE Conf. on Innovative Smart Grid Technologies*, Dubrovnik, Croatia, Oct. 2024, pp. 1–6.
- [22] Y. Du, F. Li, J. Li, and T. Zheng, "Achieving 100x acceleration for $N - 1$ contingency screening with uncertain scenarios using deep convolutional neural network," *IEEE Trans. Power Syst.*, vol. 34, no. 4, pp. 3303–3305, Jul. 2019.
- [23] A. M. Nakiganda, C. Cheylan, and S. Chatzivasileiadis, "Topology-aware neural networks for fast contingency analysis of power systems," *arXiv preprint arXiv:2310.04213*, Mar. 2025.
- [24] N. Christianson, W. Cui, S. Low, W. Yang, and B. Zhang, "Fast and reliable $N - k$ contingency screening with input-convex neural networks," in *Learning for Dynamics and Control Conference*, vol. 283, Ann Arbor, MI, Jun. 2025, pp. 1–12, proceedings of Machine Learning Research.
- [25] R. Madani, J. Lavaei, and R. Baldick, "Convexification of power flow problem over arbitrary networks," in *Proc. IEEE Conf. on Decision and Control*, Osaka, Japan, Dec. 2015.
- [26] V. Kekatos, G. Wang, H. Zhu, and G. B. Giannakis, "PSSE redux: Convex relaxation, decentralized, robust, and dynamic approaches," in *Advances in Power System State Estimation*, M. El-Hawary, Ed. Wiley, Mar. 2021, pp. 173–202.
- [27] A. Virmaux and K. Scaman, "Lipschitz regularity of deep neural networks: Analysis and efficient estimation," in *NeurIPS*, vol. 31, Montreal, Canada, Dec. 2018.
- [28] N. Hashemi, J. Ruths, and M. Fazlyab, "Certifying incremental quadratic constraints for neural networks via convex optimization," in *Learning for Dynamics and Control Conference*, ser. Proceedings of Machine Learning Research, vol. 144. Virtual: PMLR, Jun. 2021, pp. 842–853.
- [29] Y. Xu and S. Sivaranjani, "ECLipsE: Efficient compositional lipschitz constant estimation for deep neural networks," in *NeurIPS*, vol. 37, Vancouver, Canada, Dec. 2024.
- [30] M. Rahman and V. Kekatos, "Expediting AC contingency analysis with machine learning: A single-model framework," *IEEE Trans. Power App. Syst.*, Sep. 2026, submitted. [Online]. Available: <https://engineering.purdue.edu/~kekatos/papers/TPWRS2027b.pdf>

Acknowledgments

This work is supported by US NSF grant 2434502.