# Physics Informed Machine Learning for Power Flow Analysis: Injecting Knowledge via Pre-, In-, and Post-processing

Guido Parodi[1,2], Giulio Ferro[1], Michela Robba[1], Andrea Coraddu[3], Francesca Cipollini[2], Davide Anguita[1], and Luca Oneto[1(✉)]

[1] University of Genoa, Genoa, Italy
guido.parodi@edu.unige.it, {giulio.ferro,michela.robba, davide.anguita,luca.oneto}@unige.it
[2] aizoOn Technology Consulting, Turin, Italy
francesca.cipollini@aizoongroup.com
[3] Delft University of Technology, Delft, The Netherlands
a.coraddu@tudelft.nl

**Abstract.** Modern power grids are becoming increasingly complex with the integration of heterogeneous distributed energy resources, underscoring the need for accurate and efficient Power Flow Analysis to ensure stability, reliability, and market operations. Existing methods generally rely on iterative numerical techniques (INT) or machine learning (ML). While INT is physically consistent and highly accurate, it can be computationally expensive and vulnerable to slow or non-convergence. ML methods offer faster solutions but often require extensive data, suffer from limited extrapolation capabilities, and lack physical consistency. Physics-informed ML (PIML) bridges these gaps by embedding domain knowledge before, during, and after training. However, current PIML approaches typically do not leverage this full range of opportunities. In this paper, we propose a novel PIML framework for Power Flow Analysis that integrates physical insights at all three stages (pre, in, and post-processing) to achieve superior accuracy and efficiency. Notably, we introduce a new post-processing technique that partitions the power network into its mesh and radial components: the mesh portion is handled via PIML, while the radial portion is efficiently solved with a convex optimization approach informed by the PIML outputs. This approach is efficient with radial topologies, especially in power distribution networks where the radial part is predominant. Experiments on realistic power networks demonstrate that our method outperforms state-of-the-art approaches in both accuracy and computational performance.

**Keywords:** Physics Informed · Pre-Processing · In-Processing · Post-Processing · Machine Learning · Power Systems · Power Flow Analysis


I. Rojas et al. (Eds.): IWANN 2025, LNCS 16009, pp. 379–391, 2026.
https://doi.org/10.1007/978-3-032-02728-3_30

## 1 Introduction

Over the past decade, the power grid has been reshaped by distributed energy resources (DERs), including renewable energy resources (RERs), distributed generation, storage, microgrids, and flexible demand. The intermittent nature of RERs poses new challenges for grid planners and operators [19], driving the need for fast, accurate, and reliable decisions to maintain an efficient, affordable, and resilient grid [21]. In this context, power flow analysis is crucial for understanding and managing interactions among RERs, the grid, and conventional generation units [3]. By determining voltage magnitudes and phase angles under various system conditions [29], power flow analysis underpins stability, reliability, planning, and market operations in the face of growing RER integration [25].

The power flow problem (see Sect. 3) is formulated as a set of nonlinear equations that model power flow through branches and buses, accounting for power injections and line impedances. The goal is to satisfy power balance and voltage constraints at all nodes [6]. Commonly, solutions are found using iterative numerical techniques (INTs) such as Newton-Rhapson, Gauss-Seidel, and Fast Decoupled Load Flow [14]. However, these methods can converge slowly in large-scale or ill-conditioned systems and may fail to yield feasible solutions [11].

To overcome INT limitations, researchers have explored Machine Learning (ML) techniques by framing power flow analysis as a multi-output regression problem [1,8]. ML learns a function from historical or simulated data to estimate target variables, bypassing INTs. Approaches range from classical models (e.g., Ridge Regression, SVM) to complex neural networks (e.g., Graph Neural Networks). Although training is computationally and data-intensive, once complete, inference is extremely fast. However, ML models have limited extrapolation capabilities, reduced physical interpretability, and typically offer only average accuracy compared to INTs.

To address the relatively lower accuracy of ML methods compared to INTs, researchers have proposed Physics-Informed ML (PIML) [8,10], which leverages both data and domain knowledge to develop models that match the accuracy of INTs while maintaining the speed of ML. In the context of this work, the predicted voltages and currents must satisfy the power flow equations. PIML incorporates domain-specific knowledge into ML, enhancing performance and physical plausibility [15,27,31]. This integration can occur at different stages of the ML pipeline, commonly categorized into pre-processing, in-processing, and post-processing approaches. Pre-processing involves preparing and transforming data before feeding it into ML to ensure the highest-quality input. Techniques such as data cleaning, feature engineering, and data augmentation apply domain knowledge to enhance dataset relevance and quality. By leveraging domain expertise to structure and refine the data, pre-processing helps ML navigate complex data landscapes and reduces the distance they must traverse to generate accurate insights. In-processing entails directly embedding domain knowledge into the ML learning process. This may include integrating mathematical representations of physical laws, trends, or constraints into the model's architecture or introducing physics-inspired regularization terms, all while preserving desirable properties

like convexity and differentiability. The goal is to guide the model's internal learning mechanisms through physics-aware insights, thereby improving predictive accuracy at a more granular level. Post-processing refines the outputs of ML without altering the models themselves. Instead, it applies additional rules or models to enforce domain consistency. For example, model predictions can serve as inputs to well-established physical models, or logical rules can correct inconsistencies. Post-processing capitalizes on domain knowledge to contextualize and adjust ML outputs, aiming to mitigate errors and ensure alignment with established domain truths. By combining these three strategies, PIML systems can achieve both the accuracy associated with INTs and the computational efficiency characteristic of ML-based approaches.

Based on the state-of-the-art research on PIML for Power Flow Analysis reported in Sect. 2, no existing approach leverages all three stages (pre, in, and post-processing) simultaneously; most solutions focus on only one or, at best, two stages. In this paper, building on our previous work [24], we present a novel PIML approach that integrates all three stages to improve both accuracy and efficiency (see Sect. 4). Specifically, we introduce a new post-processing method that combines the reliability of INT with the speed of ML. The idea is straightforward: we split the power network into mesh and radial components. Because the Power Flow Equations are difficult to solve purely via INT for the mesh part, we apply PIML with pre- and in-processing methods [24], then we feed the output of the mesh analysis into the radial part by solving an exact convex optimization problem suitable for power distribution networks. There, the Power Flow Equations can be solved via a convex optimization problem [18] for a power distribution network, enabling effective and fast solutions. This divide-and-conquer approach in the post-processing stage fully exploits both domain knowledge and machine learning, resulting in a more effective and efficient PIML solution. Empirical results (see Sect. 5) on realistic power networks show that our method outperforms existing alternatives in terms of both accuracy and computational performance. We provide a fair comparison of INT, ML, and PIML under both interpolation and extrapolation scenarios, demonstrating that incorporating domain knowledge across multiple stages yields significantly better results than relying on a single stage. Such an in-depth analysis is seldom present in the current literature (see Sect. 2) and, to the best of our knowledge, has never been conducted with this level of detail.

## 2 Related Works

Many PIML proposals to solve Power Flow Analysis exist in the literature [8,10]. However, the current state of the art faces three main limitations.

First, existing research does not provide fair comparisons between ML and PIML approaches, often reporting only the accuracy of PIML or relying on different architectures for plain ML versus PIML [9,22]. For example, some works propose complex convolutional [2] or graph-based [5,12,13] neural networks but do not compare them against alternative methods that have demonstrated higher accuracy on equally or more intricate power grids [9,24].

Second, very few studies validate their approaches under extrapolation scenarios, going beyond simple interpolation. These methods are assessed either by testing on different ranges of input-output variables [9,22,30] or on electrical grids other than those used for training [5,12]. The former scenario is the most practical, as power networks evolve over time with the introduction of new sources or loads [19]. In contrast, the latter scenario, although interesting from an ML perspective, is often impractical in real applications, where there is little need to reuse the same model on entirely different grids and where current accuracy levels remain too low [7].

Lastly, to the best of the authors' knowledge, no work fully integrates physical power flow knowledge throughout all three stages of model creation (pre, in, and post-processing). The most advanced approaches address at most two of these stages, such as [22], which tackles pre- and post-processing, and [24], which focuses on pre- and in-processing.

Our methodology, presented in Sect. 4, addresses these gaps by proposing a PIML technique that incorporates physical insights at all three stages of training and that is validated fairly under both interpolation and extrapolation scenarios.

## 3 Problem Description and Available Data

In this section, we will first introduce the power flow problem[1] (Sect. 3.1), then we will show that for radial power distribution network the power flow problem can be solved via convex optimization (Sect. 3.2), and then we will describe the power networks and the data generated and leveraged to test the ML and the PIML (Sect. 3.3).

### 3.1 Power Flow Problem

The power grid can be represented as an undirected graph $\mathtt{G}(\mathcal{N}, \mathcal{E})$ where $\mathcal{N}$ is the set of nodes of the grid and $\mathcal{E}$ is the set of edges. The nodal quantities[2], i.e., the quantities of interest at node $n \in \mathcal{N}$, in this grid include the active power injection $P_n$, reactive power injection $Q_n$, voltage $V_n$ (real part $V_n^R$ and imaginary part $V_n^I$), and current injection $I_n$ (real part $I_n^R$ and imaginary part $I_n^I$). Since $P_n$ and $Q_n$ are quite easy to measure, the scope of the power flow model is to estimate $V_n$ and $I_n$ in all the nodes $n \in N$ [26]. Let us introduce the admittance matrix model, then the generators and loads modeling, and finally the Network Equations. All networks' lines, i.e., the edge $(i, k) \in \mathcal{E}$ that connects the nodes $i, k \in \mathcal{N}$, are modeled with a common branch model, consisting of a

[1] We denote the real and imaginary components of a complex number $C \in \mathbb{C}$ as $C^R$ and $C^I$, respectively. $j$ is the imaginary unit. The transpose and Hermitian of vector $C$ are represented as $C^T$ and $C^H$, respectively. The symbol $\odot$ is the Hadamard product, while $|C|$ and $\angle C$ denote the magnitude and phase of a complex number, respectively.

[2] It is worth noting that we adopt the active sign convention, as described by Kersting [16], where positive values are used to denote nodal injections.

standard $\pi$ line model, with series impedance $Z^{\mathcal{E}}_{i,k} = R^{\mathcal{E}}_{i,k} + jX^{\mathcal{E}}_{i,k}$ and admittance $Y^{\mathcal{E}}_{i,k} = (Z^{\mathcal{E}})^{-1}_{i,k}$. The network branch admittance matrix $Y^{br}$ is then defined as

$$Y^{br}_{i,k} = \begin{cases} \sum_{w:(i,w)\in\mathcal{E}} Y^{\mathcal{E}}_{i,w} & \text{if } i = k \\ -Y_{i,k} & \text{if } i \neq k \text{ and } (i,k) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}. \tag{1}$$

As for the shunt elements, a shunt-connected element such as a capacitor or inductor is modeled as a fixed impedance to ground at a bus. The shunt element's admittance at bus $n \in \mathcal{N}$ is given as $Y^{\mathcal{N}}_n = G^{\mathcal{N}}_n + jB^{\mathcal{N}}_n$ where $G^{\mathcal{N}}_n$ and $B^{\mathcal{N}}_n$ are the shunt conductance and susceptance at node $n \in N$. By defining the shunt matrix as

$$Y^{sh}_{i,k} = \begin{cases} Y^{\mathcal{N}}_i & \text{if } i = k \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

we can define the overall admittance matrix as $Y = Y^{br} + Y^{sh}$ where $G = Y^R$ and $B = Y^I$. For a node $n \in \mathcal{N}$, a generator is modeled as a complex power injection $S^g_n$ at a specific bus. The injection is $S^g_n = P^g_n + jQ^g_n$ where $P^g_n$ and $Q^g_n$ are active and reactive power injections. Instead, constant power loads are modeled as a specified quantity of active and reactive power consumed at a bus, $P^d_n$ and $Q^d_n$, and the load is $S^d_n = P^d_n + jQ^d_n$.

The power flow equations which describe the sinusoidal steady-state equilibrium of a power network are given by the following set of equations

$$I = YV, \quad S = V \odot I^H. \tag{3}$$

The nodal bus injections are then matched to the injections from loads and generators according to the nodal power balance equations, expressed as a function of the complex bus voltages and generator injections in complex matrix form as

$$P_n + P^d_n + P^g_n = 0, \quad Q_n + Q^d_n + Q^g_n = 0, \tag{4}$$

with $n \in N$. Equations (3) can be rewritten in terms of the real and imaginary parts of the voltage and the current obtaining the final set of equations

$$\begin{aligned} &I^R_n = \textstyle\sum_{i\in\mathcal{N}} \left(V^R_i G_{i,n} - V^I_i B_{i,n}\right), \quad I^I_n = \textstyle\sum_{i\in\mathcal{N}} \left(V^I_i G_{i,n} + V^R_i B_{i,n}\right), \\ &P_n = V^R_n I^R_n + V^I_n I^I_n, \quad Q_n = -V^R_n I^I_n + V^I_n I^R_n, \end{aligned} \tag{5}$$

with $n \in N$. The non-linear Eqs. (5) can be solved with different INT [14]. In our work we will leverage MATPOWER [32] as one state of the art package for this purpose.

### 3.2 Power Flow Problem for Radial Power Distribution Networks

In the following, a brief description of the branch flow model and its convex counterpart for radial power distribution grids is given. To continue, the following variables have to be listed (if not specified, we will consider $\forall (i,w) \in \mathcal{E}$ $i \to w$):

$I_{i,w}$ as the current from node $i$ to node $w$, $S_{w,k} = P_{w,k} + jQ_{w,k}$ as the complex power flow to the successors nodes, and $S_{i,w} = P_{i,w} + jQ_{i,w}$ as the complex power flow coming to the predecessor node. The following expression are the starting point for the branch flow model

$$V_i - V_w = Z^{\mathcal{E}}_{i,w} I_{i,w}, \quad S_{i,w} = V_i I^H_{i,w}, \quad s_w = \sum_{k:w\to k} S_{w,k} - \left(S_{i,w} - Z^{\mathcal{E}}_{i,w} |I_{i,w}|^2\right), \quad (6)$$

namely the Ohm's law, the complex power flow definition, and the power balance at each node. The last two are referred to complex powers so it is needed to split them in to real and imaginary parts. By manipulation it is possible to obtain the branch flow model

$$s_w = \sum_{k:w\to k} S_{w,k} - \left(S_{i,w} - Z^{\mathcal{E}}_{i,w} |I_{i,w}|^2\right), \quad |I_{i,w}|^2 = \left(P^2_{i,w} + Q^2_{i,w}\right) / |V_i|^2, \quad (7)$$

$$|V_w|^2 = |V_i|^2 + \left|Z^{\mathcal{E}}_{i,w}\right|^2 |I_{i,w}|^2 - 2\left(r_{i,w}P_{i,w} + x_{i,w}Q_{i,w}\right), \quad (8)$$

which is nonlinear and nonconvex. In order to obtain a convex model we need two more steps. The idea is to consider only the magnitude of the variables, eliminating the angles (this is possible only for radial networks). Calling $|V_i|^2 = v_i$ and $|I_{i,w}|^2 = l_i \ \forall (i,w) \in \mathcal{E}$ the angle relaxed branch flow equations are obtained

$$v_w = v_i + \left|Z^{\mathcal{E}}_{i,w}\right|^2 l_{i,w} - 2\left(r_{i,w}P_{i,w} + x_{i,w}Q_{i,w}\right), \quad (9)$$

$$s_w = \sum_{k:w\to k} S_{w,k} - \left(S_{i,w} - Z^{\mathcal{E}}_{i,w} l_{i,w}\right), \quad l_{i,w} = \left(P^2_{i,w} + Q^2_{i,w}\right) / v_i. \quad (10)$$

The last constrains are still not convex, so they have to be replaced

$$l_{i,w} \geq \left(P^2_{i,w} + Q^2_{i,w}\right) / v_i. \quad (11)$$

Thanks to the previous step, it is possible to state a convex optimization problem that allows to solve the power flow problem for a radial distribution network

$$\min_{l_{i,w}:(i,w)\in\mathcal{E}} \sum_{(i,w)\in\mathcal{E}} l_{i,w}, \quad \text{s.t.} \begin{cases} l_{i,w} \geq \left(P^2_{i,w} + Q^2_{i,w}\right) / v_i \\ s_w = \sum\limits_{k:w\to k} S_{w,k} - \left(S_{i,w} - Z^{\mathcal{E}}_{i,w} l_{i,w}\right) \\ v_w = v_i + \left|Z^{\mathcal{E}}_{i,w}\right|^2 l_{i,w} - 2\left(r_{i,w}P_{i,w} + x_{i,w}Q_{i,w}\right) \end{cases}. \quad (12)$$

Note that this problem allows for extremely fast solutions and completely eliminate the convergence problems.

### 3.3 Available Power Networks and Data

In this work, we employ two power networks (PN1 and PN2 in Fig. 1). PN1 is a modified version of the 17-bus radial distribution system originally introduced in [20] and made available in the MATPOWER repository [32]. To create PN1, a connection was added between nodes 9 and 11, yielding a system that can be

divided into two subnetworks: a mesh one (nodes 8 to 11) and a radial one (nodes 1 to 7 and 12 to 17). PN2 is a modified version of the 85-bus radial distribution system originally introduced in [4] and again available in the MATPOWER repository. To create PN2, two connections were added between nodes 7 and 25 and nodes 22 and 47, yielding a system that can be divided into five subnetworks: a mesh one (nodes 5 to 8, 18 to 34, 37 to 39, and 44 to 47) and four radial ones expanding from nodes 5, 8, 32, and 34.

For both PN1 and PN2, two datasets (DI and DE) were generated using the INT MATPOWER [32] package, corresponding to interpolation and extrapolation scenarios, respectively. The DI dataset was created by injecting uniform random noise of amplitude $\pm 0.2$ p.u. into the original active and reactive power values from the MATPOWER cases. A power base of 100 MVA was adopted, and the system was then solved to obtain voltages and currents at each node. The DE dataset was generated following the same procedure but with random noise drawn from the intervals $[-0.3, -0.2)$ and $(0.2, 0.3]$ p.u. In both DI and DE, a thousand samples were generated.

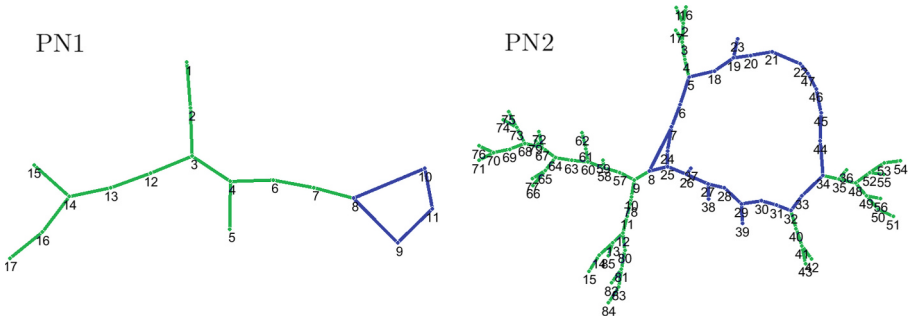


**Fig. 1.** Considered power networks (in green the radial parts)

## 4 Methodology

This work builds upon and extends the authors' previous work [24]. In this section, we first demonstrate how to build a surrogate model of the INT using machine learning (Sect. 4.1). We then present the PIML approach, which progressively integrates the physical knowledge from Sect. 3 into the ML framework. This integration is carried out through pre, in, and post-processing, discussed respectively in Sects. 4.2, 4.2, and 4.2. Finally, we describe how we conduct a fair performance evaluation of the different approaches, considering both interpolation and extrapolation scenarios.

### 4.1 Machine Learning Based Surrogate

The problem of surrogating the INT for power flow analysis, can be framed as a classical ML multi-output regression problem [28] where (see Sect. 3.1) an input space $\mathcal{X} = \mathbb{R}^{n_x}$ ($P_n$ and $Q_n$ $\forall n \in \mathcal{N}$) and a output space $\mathcal{Y} = \mathbb{R}^{n_y}$ ($V_n^R$, $V_n^I$, $I_n^R$, and $I_n^I$ $\forall n \in \mathcal{N}$) are defined and an ML model able to predict the outputs given the inputs $f_\Theta : \mathcal{X} \to \mathcal{Y}$ has to be learned ($\Theta$ are the learnable parameters). $\Theta$ are commonly learned solving the following optimization problem

$$\min_\Theta \frac{1}{|D|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \ell(f_\Theta(\boldsymbol{x}), \boldsymbol{y}), \tag{13}$$

where $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \cdots, (\boldsymbol{x}_{n_d}, \boldsymbol{y}_{n_d})\}$ is a series of $n_d$ observations of the input-output relation and $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is the loss function.

In line with our previous work [24], we opt for a single fully connected hidden-layer neural network, namely

$$f_{\Theta=\{W^o,W^h\}}(\boldsymbol{x}) = W^o\varphi(W^h\boldsymbol{x}), \tag{14}$$

where $W^o \in \mathbb{R}^{n_y \times n_h}$, $W^h \in \mathbb{R}^{n_h \times n_x}$, $\varphi$ is the activation function, and $n_h$ is the number of hidden units.

The data utilized for the training and model selection phases are randomly sampled from the DI dataset (see Sect. 3.3), consisting of 700 samples, 90% for learning and 10% for validation [23]. The remaining 300 samples of DI and the entire DE (see Sect. 3.3) are used in the performance estimation phase [23] namely for computing the interpolation and extrapolation errors respectively. In this last phase, the Mean Absolute Error (MAE) is employed.

During the training phase, the Mean Squared Error (MSE) is chosen as loss function and the ADAM algorithm [17] is used to solve Problem 13, empowered by a scheduler that decreases the learning rate $\rho$ by a factor $\rho_f$ when training error does not decrease more than a threshold $\rho_t$ for a given epoch number $\rho_e$. As a consequence, the hyperparameters of the learning procedure are the number of hidden units $n_h$, the activation function $\varphi$, the batch size of the optimizer $b$, the initial learning rate $\rho_0$, the learning factor of the scheduler $\rho_f$, the threshold of the scheduler $\rho_t$, the epoch number of the scheduler $\rho_e$, and the total epochs $n_e$.

Hyperparameters are optimized during the model selection phase performing an exhaustive search [23] within a grid resulting from the Cartesian product of the following values: $n_h \in \{10^2, 10^{2.2}, \cdots, 10^3\}$, $\varphi \in \{\texttt{relu}, \texttt{sigmoid}, \texttt{tanh}\}$ where `relu` is the rectified linear unit, `sigmoid` is the sigmoid function, and `tanh` is the hyperbolic tangent function, $b \in \{32, 64, 128\}$, $\rho_0 \in \{10^{-5}, 10^{-4}, \cdots, 10^{-2}\}$, $\rho_f \in \{0.90, 0.95, 0.99\}$, $\rho_t \in \{10^{-7}, 10^{-6}, \cdots, 10^{-4}\}$, $\rho_e \in \{10, 15, 20\}$, and $n_e \in \{500, 600, \cdots, 1000\}$.

### 4.2 From Machine Learning to Physics Informed Machine Learning

For the PIML technique, we build upon the ML framework and we empower it progressively with the physical knowledge via pre, in, and post-processing.

**In-processing.** For the in-processing approach, following [24], we construct a custom regularizer $R(\Theta)$ to be added in Problem (13) that leverages Eqs. (3). To do so, we consider the fact that Eqns. (3) tells us that the output ($V$ and $I$ namely $V_n^R$, $V_n^I$, $I_n^R$, and $I_n^I$, $\forall n \in \mathcal{N}$) of our model $f_\Theta$ based on its input ($S$ namely $P$ and $Q$ that are $P_n$ and $Q_n$, $\forall n \in \mathcal{N}$) needs to satisfy the constraint stated in the equations themselves, namely $\|I - YV\| + \|S - V \odot I^H\|$ should be as small as possible (ideally, it should tend to zero). For this reason, it is possible to formulate $R(\Theta)$ as follows

$$R(\Theta, \mathcal{D}) = \lambda \sum_{S \in \mathcal{D}} (\|I(f_\Theta(S)) - YV(f_\Theta(S))\| + \|S - V(f_\Theta(S)) \odot I^H(f_\Theta(S))\|), \tag{15}$$

where $\lambda \in [0, \infty)$ is a hyperparameter which regulates how much we want to enforce the effect of the regularizer and $V(f_\Theta(S))$ and $I(f_\Theta(S))$ are the $V$ and $I$ predicted by $f_\Theta$ when $S$ is provided as input, respectively. Note that $\lambda$, theoretically, should be as large as possible so as to enforce the physical knowledge as much as possible. This cannot be done in practice since it increases the complexity of the minimization problem. To address this issue, we also define a scheduler for $\lambda$. We start from $\lambda = 0$ and arrive to $\lambda = \lambda^{\max}$ increasing logarithmically $\lambda$ every $\eta n_e$ epochs with $\eta \in [0, 1]$. So, in model selection phase we follow the same procedure described for the ML technique enriching the search space by making a Cartesian product with $\lambda^{\max} \in \{10^{-2}, 10^{-1}, \cdots, 10^{1}\}$ $\eta = \{.1, .2, .5\}$.

**Pre-processing.** Noting that the regularizer of Eq. (15) can be computed without the knowledge of the actual output in the datasets but just the input (as the output is the one produced by the network) we can leverage, as in [24], this property to define a pre-processing approach using a simple data augmentation strategy. In particular, it is possible to modify the regularizer of Eq. (15) as follows

$$\tilde{R}(\Theta, \mathcal{D}) = R(\Theta, \mathcal{D})/2 + R(\Theta, \tilde{\mathcal{D}})/2, \quad \tilde{D} = \{S + \mathtt{N}_S(\sigma) : S \in \mathcal{D}\}, \tag{16}$$

where $\mathtt{N}_S(\sigma)$ is a Gaussian noise of variance $\sigma$ applied to all elements of $S$. Note that, compared to Regularizer (15), Regularizer (16) enables a more precise estimation and exploration of the behavior of $f_\Theta$. The hyperparameter $\sigma$ is tuned during the model selection phase searching for the best value in $\{0.001, 0.01, 0.1, 1, 1.5, 2, 3\}$.

**Post-processing.** For the post-processing approach, we rely on the fact that a power network can be decomposed into two main components: a mesh part and a radial part (see Fig. 1). This decomposition is performed by replacing the interface between these two sub-networks with an equivalent node with fixed voltage in magnitude and phase (coming from the PIML result), allowing each portion to be treated independently [32]. We employ the PIML approach for the mesh sub-network, which includes both pre-processing and in-processing. For the radial sub-network, we take the PIML output to construct the equivalent node,

then we feed this equivalent into the convex formulation described in Sect. 3.2. This procedure enables us to incorporate the relevant physical knowledge in a post-processing stage, increasing the overall precision and thereby reducing the burden on the PIML to learn the entire network. Consequently, a convex solver can solve the radial portion quickly and efficiently.

**Table 1.** Accuracy, in interpolation and extrapolation, and prediction time for ML and PIML with In, In+Pre, and In+Pre+Post processing on PN1 and PN2. Note that the INT takes approximately 2s for both PN1 and PN2.

| PN | Methods | | Pred. Time | Interpolation | | | Extrapolation | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | $[10^{-5}s]$ | MAE $[10^{-5}$ p.u.] | MAPE [%] | Imp↰ [%] | MAE $[10^{-5}$ p.u.] | MAPE [%] | Imp↰ [%] |
| PN1 | ML | | 3 | $4.05 \pm 0.62$ | $0.55 \pm 0.05$ | – | $35.6 \pm 5.3$ | $5.31 \pm 0.49$ | – |
| | PIML | (In) | 3 | $3.60 \pm 0.57$ | $0.49 \pm 0.05$ | 11 | $35.1 \pm 5.3$ | $5.26 \pm 0.53$ | 1 |
| | | (In+Pre) | 3 | $3.35 \pm 0.49$ | $0.47 \pm 0.04$ | 7 | $30.4 \pm 4.9$ | $4.49 \pm 0.45$ | 13 |
| | | (In+Pre+Post) | 356 | $1.51 \pm 0.38$ | $0.14 \pm 0.02$ | 55 | $19.0 \pm 4.2$ | $1.83 \pm 0.30$ | 38 |
| PN2 | ML | | 5 | $5.85 \pm 1.37$ | $0.45 \pm 0.05$ | – | $73.0 \pm 16.4$ | $5.45 \pm 0.51$ | – |
| | PIML | (In) | 9 | $4.90 \pm 1.2$ | $0.38 \pm 0.04$ | 16 | $70.6 \pm 15.6$ | $5.34 \pm 0.50$ | 3 |
| | | (In+Pre) | 8 | $4.21 \pm 1.0$ | $0.34 \pm 0.04$ | 14 | $37.4 \pm 8.9$ | $2.98 \pm 0.30$ | 47 |
| | | (In+Pre+Post) | 580 | $1.50 \pm 0.7$ | $0.16 \pm 0.02$ | 64 | $12.1 \pm 5.1$ | $1.13 \pm 0.17$ | 68 |

## 5 Experimental Results

In this section, we report the results of applying the methodology described in Sect. 4 to the problem introduced in Sect. 3, using the data presented in that section.

Specifically, Table 1 reports, for PN1 and PN2, the average accuracy (computed over all node voltages and currents) and the prediction time (i.e., the time needed to predict voltage and current given the power) for all considered methods: ML and PIML with In, In+Pre, and In+Pre+Post processing (see Sect. 4). The accuracy was measured both in interpolation and extrapolation, using the MAE and the Mean Absolute Percentage Error (MAPE), along with the percentage improvement compared to the previously best method (Imp↰). Moreover, Fig. 2 presents, for PN1 and PN2, the probability density function of the absolute error for all considered methods (ML and PIML with In, In+Pre, and In+Pre+Post processing), under both interpolation and extrapolation.

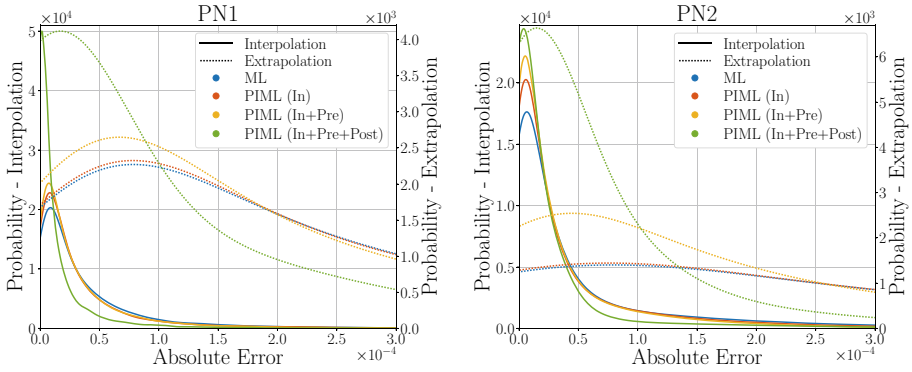


**Fig. 2.** Error distribution, in interpolation and extrapolation, for ML and PIML with In, In+Pre, and In+Pre+Post processing on PN1 and PN2.

From Table 1 and Fig. 2, we can draw several observations. While PIML consistently outperforms ML, the greatest performance gains are achieved through the newly proposed post-processing technique. In both interpolation and extrapolation settings, the improvements are evident. With a reasonably small amount of data, the most advanced PIML approach achieves errors that are practically negligible. Finally, note the significant savings in computational time compared to INT (up to five orders of magnitude). Although the most advanced PIML technique is slightly slower, due to the need to solve an optimization problem, its computational requirements remain entirely practical.

## 6 Conclusions

This work demonstrates the effectiveness of embedding domain knowledge at all stages of an ML pipeline to produce a robust and efficient power flow solution for increasingly complex power grids. By leveraging physics-informed insights in pre-processing, integrating them during training, and employing an innovative post-processing step that decouples mesh and radial network components, the proposed framework achieves superior accuracy and computational performance compared to conventional INT, ML, and PIML approaches. Notably, the use of post-processing partitioning optimally combines data-driven methods with fast numerical solvers for radial network segments, reducing computational overhead while preserving physical fidelity. These results highlight the promise of a more holistic PIML approach that capitalizes on deep learning and domain expertise, offering a powerful tool for real-world grid operations and paving the way for future refinements, such as applications in large-scale and dynamic scenarios.

**Acknowledgments.** This work is partially supported by (i) project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU, (ii) project FAIR (PE00000013) under the NRRP MUR program funded by the EU - NGEU, and (iii) project NEST (PE000021) under the NRRP MUR program funded by the EU - NGEU.

## References


1. Akter, M., Nazaripouya, H.: A review of data-driven methods for power flow analysis. In: North American Power Symposium (2023)
2. Baghaee, H.R., Mirsalim, M., Gharehpetian, G.B., Talebi, H.A.: Three-phase AC/DC power-flow for balanced/unbalanced microgrids including wind/solar, droop-controlled and electronically-coupled distributed energy resources using radial basis function neural networks. IET Power Electron. **10**, 313–328 (2017)
3. Chen, H., Chen, J., Shi, D., Duan, X.: Power flow study and voltage stability analysis for distribution systems with distributed generation. In: IEEE Power Engineering Society General Meeting (2006)
4. Das, D., Kothari, D., Kalam, A.: Simple and efficient method for load flow solution of radial distribution networks. Int. J. Electr. Power Energy Syst. **17**, 335–346 (1995)
5. Donon, B., Clément, R., Donnot, B., Marot, A., Guyon, I., Schoenauer, M.: Neural networks for power flow: graph neural solver. Electr. Power Syst. Res. **189**, 106547 (2020)
6. Ferro, G., Robba, M., D'Achiardi, D., Haider, R., Annaswamy, A.M.: A distributed approach to the optimal power flow problem for unbalanced and mesh networks. IFAC-PapersOnLine **53**(2), 13287–13292 (2020)
7. Gea-Bermúdez, J., Pade, L., Koivisto, M.J., Ravn, H.: Optimal generation and transmission development of the north sea region: impact of grid architecture and planning horizon. Energy **191**, 116512 (2020)
8. Gong, X., Wang, X., Cao, B.: On data-driven modeling and control in modern power grids stability: survey and perspective. Appl. Energy **350**, 121740 (2023)
9. Hu, X., Hu, H., Verma, S., Zhang, Z.: Physics-guided deep neural networks for power flow analysis. IEEE Trans. Power Syst. **36**, 2082–2092 (2020)
10. Huang, B., Wang, J.: Applications of physics-informed neural networks in power systems-a review. IEEE Trans. Power Syst. (2022)
11. Iwamoto, S., Tamura, Y.: A load flow calculation method for ill-conditioned power systems. IEEE Trans. Power Appar. Syst. **4**, 1736–1743 (1981)
12. Jeddi, A.B., Shafieezadeh, A.: A physics-informed graph attention-based approach for power flow analysis. In: IEEE International Conference on Machine Learning and Applications (2021)
13. de Jongh, S., Gielnik, F., Mueller, F., Schmit, L., Suriyah, M., Leibfried, T.: Physics-informed geometric deep learning for inference tasks in power systems. Electr. Power Syst. Res. **211**, 108362 (2022)
14. Karimi, M., Shahriari, A., Aghamohammadi, M.R., Marzooghi, H., Terzija, V.: Application of newton-based load flow methods for determining steady-state condition of well and ill-conditioned power systems: A review. Int. J. Electr. Power Energy Syst. **113**, 298–309 (2019)
15. Karniadakis, G., Kevrekidis, I., Lu, L., Perdikaris, P., Wang, S., Yang, L.: Physics-informed machine learning. Nat. Rev. Phys. **3**(6), 422–440 (2021)
16. Kersting, W.H.: Distribution System Modeling and Analysis. CRC press, Boca Raton (2017)
17. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
18. Lavaei, J., Low, S.H.: Zero duality gap in optimal power flow problem. IEEE Trans. Power Syst. **27**(1), 92–107 (2011)

19. Mahmud, N., Zahedi, A.: Review of control strategies for voltage regulation of the smart distribution network with high penetration of renewable distributed generation. Renew. Sustain. Energy Rev. **64**, 582–595 (2016)
20. Mendoza, J.E., Morales, D.A., Lopez, R.A., Lopez, E.A., Vannier, J., Coello, C.A.C.: Multiobjective location of automatic voltage regulators in a radial distribution network using a micro genetic algorithm. IEEE Trans. Power Syst. **1**, 404–412 (2007)
21. Molzahn, D.K., Dörfler, F., Sandberg, H., Low, S.H., Chakrabarti, S., Baldick, R., Lavaei, J.: A survey of distributed optimization and control algorithms for electric power systems. IEEE Trans. Smart Grid **8**(6), 2941–2962 (2017)
22. Müller, H.H., Rider, M.J., Castro, C.A.: Artificial neural networks for load flow and external equivalents studies. Electr. Power Syst. Res. **80**, 1033–1041 (2010)
23. Oneto, L.: Model Selection and Error Estimation in a Nutshell. MOST, vol. 15. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-24359-3
24. Parodi, G., et al.: Physics informed data driven techniques for power flow analysis. In: IEEE Symposium Series on Computational Intelligence (2023)
25. Potter, A., Haider, R., Ferro, G., Robba, M., Annaswamy, A.M.: A reactive power market for the future grid. Adv. Appl. Energy **9**, 100114 (2023)
26. Primadianto, A., Lu, C.N.: A review on distribution system state estimation. IEEE Trans. Power Syst. **32**(5), 3875–3883 (2016)
27. Raissi, M., Perdikaris, P., Karniadakis, G.E.: Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J. Comput. Phys. **378**, 686–707 (2019)
28. Shalev-Shwartz, S., Ben-David, S.: Understanding machine learning: from theory to algorithms (2014)
29. Stott, B.: Review of load-flow calculation methods. Proc. IEEE **62**(7), 916–929 (1974)
30. Sun, P., Wu, R., Wang, H., Li, G., Khalid, M., Konstantinou, G.: Physics-informed fully convolutional network-based power flow analysis for multi-terminal mvdc distribution systems. IEEE Trans. Power Syst. (2024)
31. Zampini, S., Parodi, G., Oneto, L., Coraddu, A., Anguita, D.: A review on full-, zero-, and partial-knowledge based predictive models for industrial applications. Inf. Fusion **119**, 102996 (2025)
32. Zimmerman, R.D., Murillo-Sánchez, C.E., Thomas, R.J.: MATPOWER: steady-state operations, planning, and analysis tools for power systems research and education. IEEE Trans. Power Syst. **26**(1), 12–19 (2010)