2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Physics Informed Data Driven Techniques for Power Flow Analysis

Guido Parodi, Luca Oneto, Giulio Ferro, Stefano Zampini, Michela Robba, Davide Anguita
University of Genoa, Via Opera Pia 11a, 16145, Genova, Italy
name.surname@unige.it

Andrea Coraddu
Delft University of Technology
Mekelweg 5, 2628 CC, Delft, The Netherlands
a.coraddu@tudelft.nl

***Abstract*—The last decade has seen significant changes in the power grid complexity due to the increased integration of multiple heterogeneous distributed energy resources. Accurate and fast power flow analysis tools have then become essential to guarantee grid stability, reliable operation, strategic planning, and market strategies. State-of-the-art approaches to power flow analysis are based on iterative numerical techniques which exhibit high accuracy but slow-, or even no-, convergence. For this reason, researchers have investigated the use of data-driven techniques that, while exhibiting lower accuracy with respect to iterative numerical ones, have the advantage of being extremely fast. To address the lack of accuracy, physics-informed data-driven techniques, i.e., techniques that leverage both the data and domain knowledge to generate simultaneously fast and accurate models, have been proposed. Nevertheless, these works exhibit two main limitations: i) they do not fully leverage the physical knowledge, and ii) they do not fairly compare the different approaches. In this paper, we propose a novel physics informed data-driven model able to address both limitations by fully leveraging the physical knowledge into the data-driven, i.e., constraining the model and augmenting the available data, and proposing a framework able to fairly compare the different approaches proving the actual effectiveness of the proposal. Results on the IEEE 57 realistic power network will support the proposal.**

***Index Terms*—Physics Informed, Machine Learning, Power Systems, Power Flow Analysis**

I. Introduction

Over the last decade, the power grid has been transformed due to the increasing presence of distributed energy resources (DERs). DERs include renewable energy resources (RERs), distributed generation, storage systems, microgrids, and flexible demand. These DERs introduce a host of new challenges for grid planners and operators, particularly in the presence of unpredictable and intermittent RERs [1]. As a result, there is an increasing need to make fast, accurate, and reliable decisions for an efficient, affordable, and resilient grid [2].

In this context, the power flow analysis is essential in understanding and managing the dynamic interactions between RER, the power grid, and other conventional generation units [3]. Power flow analysis involves determining the voltage magnitude and phase angles at different nodes in an electrical power system, given the system topology, generation, and load conditions [4]. It aids in ensuring grid stability, reliable operation, efficient planning, and effective market operations in the context of RER integration [5].

The power flow problem can be mathematically formulated as a set of nonlinear equations. The equations describe the power flow in various branches and buses of the power system, considering the power injections and line impedances. The goal is to find a solution that satisfies the power balance and voltage magnitude constraints for all nodes in the system [6]. Solving the power flow problem involves iterative numerical techniques (INT). The most common INT used are the Newton-Rhapson method, the Gauss-Seidel method, and the Fast Decoupled Load Flow [7]. However, these methods have some drawbacks related to the slow convergence for large-scale networks and specific topologies, furthermore, for ill-conditioned systems, these algorithms are not able to retrieve a feasible solution [8].

To address the limitations of INT researchers have investigated the use of Data-Driven Techniques (DDT) which basically transform the power flow analysis problem into a classical machine learning multi-output regression problem [9]–[11]. In particular, DDT leverages historical data (collected in-field or produced with simulators) to learn a function able to estimate the desired quantities without the need for INT [12]. The learning phase of this functional is both computational and data demanding but once performed (usually offline) its use is extremely fast. DDT, while being extremely fast, exhibits lower accuracy with respect to INT [9]–[11].

In order to address the lower accuracy of DDT with respect to INT, researchers have proposed to use Physics Informed DDT (PIDDT) [13]–[16], i.e., techniques that leverage both the data and domain knowledge to generate models that are accurate as the INT and fast as the DDT [17]. The main idea behind the current PIDDT is to constrain the functional learned by the DDT to satisfy some physical principle (e.g., in our case the predicted voltage and currents need to follow the power flow equations [15]).

The limitation of current PIDDT is twofold. The first one i) is that current PIDDT do not fully leverage the physical knowledge. The second one ii) is that they do not fairly compare the three approaches (INT, DDT, and PIDDT) providing only the accuracy of PIDDT or leveraging different architectures in plain DDT and PIDDT.

To overcome these limitations, in this work, we propose a novel PIDDT with a twofold contribution. Firstly, our methodology integrates domain-specific knowledge into the data-driven model by constraining compliance with the power flow equations and by augmenting the available data. Secondly, we present a framework able to fairly compare the different strategies (INT, DDT, and PIDDT), proving the actual effectiveness of the proposal.

For this purpose, we will use data created from the IEEE

978-0-7381-4408-5/23/$31.00 ©2023 IEEE

57-bus case. This network consists of 57 buses, 7 generators, and 42 loads. The data baseline has been created from the MATPOWER repository [18].

The rest of the paper is organized as follows. Section II reports the current status of the literature related to our work. Section III formally describes our problem and the available data. Section IV presents the proposed methodology. Section V presents the results of applying the methodology described in Section IV to the data described in Section III. Section VI concludes the paper.

II. Related Works

In the context of DDT for power flow analysis several approaches have been proposed. In the context of shallow machine learning models, authors have leveraged Least Squares and Bayesian linear regression [9], Koopman operator method [10], and (non-)linear Support Vector Machines [11]. In the context of (deep) neural models, authors have leveraged Radial Basis Function Network [19], Graph Convolutional Network [20], and Stacked Denoising Auto-Encoders [21].

In the context of PIDDT for power flow analysis some approaches, less than the ones based on DDT, have been proposed [16]. Authors of [13] proposed a multilayer perceptron model that leverages the power flow branch equations to compute additional features to predict physically admissible solutions. Authors of [14] proposed an approach in which the physical information is embedded into the model by adding network equations as penalty terms inside the loss function used to train a neural network. Finally, the authors of [15] presented an encoder-decoder architecture which takes into account the structure of the Kirchhoff's laws and the system topology achieving good interpolation and extrapolation performance.

III. Problem Formalization and Available Data

In this section, we will first introduce the power flow problem[1] (Section III-A) and then we will describe the data generated and leveraged by the DDTs and the PIDDTs (Section III-C).

A. 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, i.e., the quantities of interest at node $n \in N$, 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 $n \in N$ [22]. It is worth noting that we adopt the active sign convention, as described by Kersting [23], where positive values are used to denote nodal injections.

[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.

B. Iterative Numerical Technique

As a first step, we will describe the INT. First, we will 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 standard π line model, with series impedance $Z_{i,k}^{\mathcal{E}} = R_{i,k}^{\mathcal{E}} + jX_{i,k}^{\mathcal{E}}$ and admittance $Y_{i,k}^{\mathcal{E}} = (Z^{\mathcal{E}})_{i,k}^{-1}$. The network branch admittance matrix Y^{br} is then defined as follows:

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

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 $Y_n^{\mathcal{N}} = G_n^{\mathcal{N}} + jB_n^{\mathcal{N}}$ where $G_n^{\mathcal{N}}$ and $B_i^{\mathcal{N}}$ are the shunt conductance and susceptance at node $n \in N$. By defining the shunt matrix as

$$Y_{i,k}^{sh} = \begin{cases} Y_i^{\mathcal{N}} & \text{if } i = k \\ 0 & \text{otherwise} \end{cases} , \quad (2)$$

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_n^g at a specific bus. The injection is $S_n^g = P_n^g + jQ_n^g$ where P_n^g and Q_n^g 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_n^d and Q_n^d, and the load is $S_n^d = P_n^d + jQ_n^d$.

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

$$I = YV, \quad (3)$$

$$S = V \odot I^H. \quad (4)$$

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

$$P_n + P_n^d + P_n^g = 0, \quad Q_n + Q_n^d + Q_n^g = 0, \quad (5)$$

with $n \in N$. Eq. (5) can be rewritten in terms of polar coordinates according to the possible formulation of the voltage phasor

$$V_n = V_n^R + jV_n^I, \quad V_n = |V_n|(\cos(\angle V_n) + j\sin(\angle V_n)), \quad (6)$$

with $n \in N$, obtaining the equations in terms of the real and imaginary parts of the voltage

$$P_n = \sum_{k\in N} G_{n,k}(V_n^R V_k^R + V_n^I V_k^I) + B_{n,k}(V_n^I V_k^R - V_n^R V_k^I),$$
$$Q_n = \sum_{k\in N} G_{n,k}(V_n^I V_k^R - V_n^R V_k^I) - B_{n,k}(V_n^R V_k^R + V_n^I V_k^I), \quad (7)$$

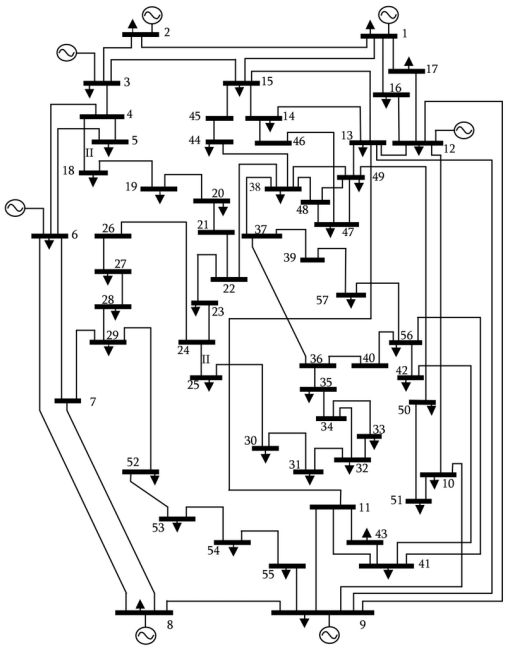


Fig. 1: IEEE 57-bus.

with $n \in N$, or in terms of magnitude and phase angle of the voltage

$$P_n = \sum_{k \in N} |V_n||V_k| \left((G_{n,k} + jB_{n,k}) e^{-j(\angle V_n - \angle V_k)} \right)^R,$$
$$Q_n = -\sum_{k \in N} |V_n||V_k| \left((G_{n,k} + jB_{n,k}) e^{-j(\angle V_n - \angle V_k)} \right)^I, \quad (8)$$

with $n \in N$, or in terms of real and imaginary parts of the voltage and the current

$$P_n = V_n^R I_n^R + V_n^I I_n^I, \quad Q_n = -V_n^R I_n^I + V_n^I I_n^R, \quad (9)$$

with $n \in N$. The non-linear Eqns. (7), (8), and (9) can be solved with different INT [7] but also free tools are available like MATPOWER [18].

C. Available Data

The network considered in this paper is the IEEE 57-bus case (Figure 1). The IEEE 57-bus has been chosen as it is small and easy to use, yet complex enough to use for the validation of our proposed method similarly to what has been done in [9], [10], [13], [15]. Regarding power generation, this test system has 7 distinct generating units, each situated at individual busbars.

The network is constituted of 57 bus, complemented by 7 generators, and 42 loads.

The dataset has been generated using MATPOWER's INT. In order to generate the data, we set a random amplitude noise ± 0.2 per unit, using a power basis of 100[MVA]. For each sample, we rely on the MATPOWER INT to retrieve voltages and currents. The dataset is composed of 10-thousand unique samples.

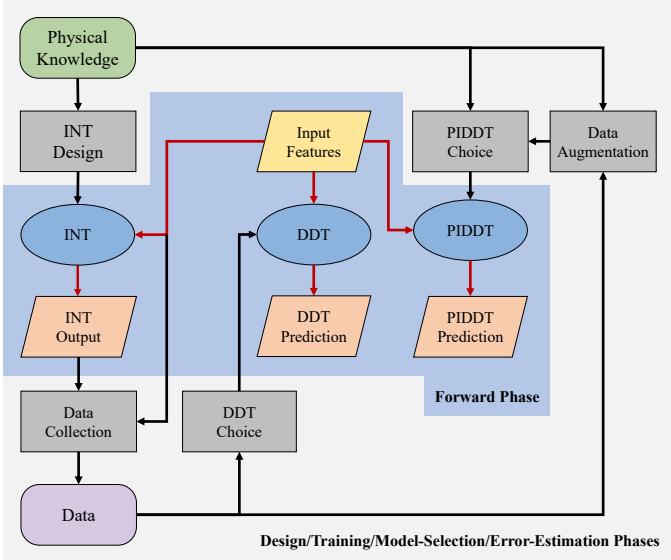


Fig. 2: A graphical abstract of Section IV.

IV. Methodology

In this section, we will provide an overall description of how we moved from INT to DDT, followed by an explanation of the shift to PIDDT, as referenced in Section IV-A. In particular, we will show how to construct a surrogate/digital-twin of the INT able to solve the power flow problem with the same INT's level of accuracy at a fraction of its computational costs. Then we will describe in detail the proposed DDT surrogate of the INT, carefully describing our choices and comparing them with the literature (Section IV-B). Finally, we will describe how to switch from the DDT to the PIDDT that fully embeds the physical knowledge into the DDT (Section IV-C). A graphical abstract of this section is reported in Figure 2.

A. Preliminaries

Creating a surrogate [24] of the INT-based solution of the power flow problem (Section III-A) can be mapped into a now-classical machine learning multi-output regression problem [12]. In particular, given an input space $\mathcal{X} = \mathbb{R}^{n_x}$ (in our case P_n and Q_n $\forall n \in \mathcal{N}$) and the associated output space $\mathcal{Y} = \mathbb{R}^{n_y}$ (in our case V_n^R, V_n^I, I_n^R, and I_n^I $\forall n \in \mathcal{N}$) where the association is made through the INT (Section III-B), namely a deterministic rule, the goal is to learn a model $f_\Theta : \mathcal{X} \to \mathcal{Y}$, characterized by its set of parameters Θ, through an algorithm $\mathscr{A}_\mathcal{H}$, characterized by its set of hyperparameters $\mathcal{H}$, that chooses f_Θ in order to well approximate the input output relation, in our case the INT. The quality of the approximation is measured according to a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. $\mathcal{H}$ takes into account many different aspects [12], [25], [26]: the functional form of f_Θ (e.g., linear, kernel, convolutions, pooling, and transformers), parameterization of the functional form (e.g., the kernel type, the kernel hyperparameters, the number of layers, and the type of layers), the possible explicit regularization (e.g., type of norm and amount of regularization and over-parameterization), and implicit regularization (e.g., type of optimizer, early stopping, and dropout).

In DDT, Θ and $\mathcal{H}$ are chosen (during the training and model selection phases respectively [12], [27]) purely based on a series of n_d observations of the input and corresponding output called dataset $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \cdots, (\boldsymbol{x}_{n_d}, \boldsymbol{y}_{n_d})\}$ with $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$. Usually the training phases consist in solving the following problem

$$\min_{\Theta} \frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \ell(f_\Theta(\boldsymbol{x}), \boldsymbol{y}) + R(\Theta), \tag{10}$$

namely, once chosen $\mathcal{H}$ a regularized empirical risk minimization procedure is employed where some hyperparameters appear explicitly (e.g., inside $R(\Theta)$) while others are hidden inside the functional form of f_Θ or in the $\min$. Once Problem (10) is solved the generalization performance of $\mathscr{A}_\mathcal{H}$, namely, its ability to find optimal f_Θ that works well on previously unseen data, is tuned during the model selection phase and estimated during the error estimation phase [27]. Researchers and practitioners commonly use resampling techniques since they work well in most situations. Resampling techniques are based on a simple idea: the original dataset $\mathcal{D}$ is resampled once or many (n_r) times, with or without replacement, to build three independent datasets called learning, validation, and test sets, respectively $\mathcal{L}^r$, $\mathcal{V}^r$, and $\mathcal{T}^r$, with $r \in \{1, \cdots, n_r\}$ such that $\mathcal{L}^r \cap \mathcal{V}^r = \oslash$, $\mathcal{L}^r \cap \mathcal{T}^r = \oslash$, $\mathcal{V}^r \cap \mathcal{T}^r = \oslash$, and $\mathcal{L}^r \cup \mathcal{V}^r \cup \mathcal{T}^r = \mathcal{D}$. Subsequently, to select the best combination of hyperparameters $\mathcal{H}^*$ in a set of possible ones $\mathcal{S} = \{\mathcal{H}_1, \mathcal{H}_2, \cdots\}$ for the algorithm $\mathscr{A}_\mathcal{H}$ or, in other words, to perform the model selection phase, the following procedure has to be applied

$$\min_{\mathcal{H}\in\mathcal{S}} \frac{1}{n_r} \sum_{r=1}^{n_r} \frac{1}{|\mathcal{V}^r|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{V}^r} \ell(\mathscr{A}_\mathcal{H}(\mathcal{L}^r)(\boldsymbol{x}), \boldsymbol{y}), \tag{11}$$

where $f_{\Theta^*} = \mathscr{A}_\mathcal{H}(\mathcal{L}^r)$ is a model built with the algorithm $\mathscr{A}_\mathcal{H}$ with its set of hyperparameters $\mathcal{H}$ and with the data $\mathcal{L}^r$. Since the data in $\mathcal{V}^r$ has not been seen during the training phase, the selected set of hyperparameters is the one that allows achieving a small error on a previously unseen set of data. Then, to evaluate the performance of the optimal model, i.e., the model f_{Θ^*} (the solution of Problem (10)) with the best hyperparameters configurations $\mathcal{H}^*$ (the solution of Problem (11)) the following quantity is computed

$$\frac{1}{n_r} \sum_{r=1}^{n_r} \frac{1}{|\mathcal{T}^r|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{T}^r} \ell(\mathscr{A}_{\mathcal{H}^*}(\mathcal{L}^r \cup \mathcal{V}^r)(\boldsymbol{x}), \boldsymbol{y}), \tag{12}$$

where $f_{\Theta^*} = \mathscr{A}_{\mathcal{H}^*}(\mathcal{L}^r \cup \mathcal{V}^r)$ is a model built with the algorithm $\mathscr{A}$ with the best set of hyperparameters $\mathcal{H}^*$ and with the data $\mathcal{L}^r \cup \mathcal{V}^r$. Since the data $\mathcal{T}^r$ has not been seen during the training phase nor the model selection phase the quantity of Eq. (12) is an unbiased estimator of the generalization ability of the final model [27].

In PIDDT everything is quite similar to what has been explained for DDT except for the fact that training and model selection phases are not made purely based on the data $\mathcal{D}$ but also based on the physical knowledge about how f_Θ, i.e., the model that we want to learn, should behave [17]. For example, in our case we know that the input/output relation induced by the INT should satisfy the Eqns. (3) and (4) (where Eq. (4) can be written as Eqns. (7), or (8), or (9)). In other cases the knowledge may be less precise, e.g., f_Θ should be monotonically increasing in some variable or its derivative should not be greater than a certain threshold. At the same time the physical knowledge may be leveraged also in augmenting $\mathcal{D}$. More formally, the physical knowledge can be encapsulated in $\mathscr{A}_\mathcal{H}$ in many different ways [16], [17]: by modifying functional form of f_Θ (e.g., that should satisfy some property), by including/excluding some parameterization of the functional form (e.g., some choice of the architecture may be not useful), as explicit regularization/constraint (e.g., transforming a constraint into a regularizer), and as implicit regularization (e.g., augmenting the data).

B. The proposed Data-Driven Technique

The no-free-lunch theorem [28] ensures us that, in order to find the best algorithm for a particular application, it is necessary to test multiple algorithms since no optimal a-propri choice can be made. Nevertheless, in our case, domain-specific constraints limit the set of possible choices. In fact, our purpose is to surrogate the INT with a DDT that can achieve high accuracy with limited computational requirements when it comes to computing $f_\Theta(\boldsymbol{x})$, namely in the forward phase. The computational requirement constraints immediately exclude some machine learning algorithms: kernel methods or ensemble methods that, in order to achieve high accuracy, need a totally different model for each of the outputs [12], resulting in a computationally expensive forward phase. Instead, neural networks, are the most suitable choice in this context since their structure is composed of a first non-linear block that learns a representation fed to a linear block that makes the actual predictions resulting in a model which is quite computationally efficient [25], [26]. For the specific needs of our application, deploying a deep neural network (e.g., graph neural network, convolution, or transformer) may not be the most effective or sensible choice. In fact, the topology of our power network is fixed, and a simple vector $\boldsymbol{x} \in \mathbb{R}^{n_x}$ containing P_n and Q_n $\forall n \in \mathcal{N}$ is already a good representation of the power network. So a single fully connected hidden-layer neural network should be a good choice: it allows to rely on an universal approximator [25], [26] limiting, at the same time, the possible number of parameters (and then computational requirements of the forward phase) and hyperparameters (and then the computational requirements of building the DDT). More formally, our choice for the network is the following one

$$f_{\Theta=\{W^o,W^h\}}(\boldsymbol{x}) = W^o \varphi(W^h \boldsymbol{x}), \tag{13}$$

where $W^o \in \mathbb{R}^{n_y \times n_h}$, $W^h \in \mathbb{R}^{n_h \times n_x}$ (for a total of $n_y n_h + n_h n_x$ parameters), φ is the activation function, and n_h is the number of hidden unit. Then, for $\mathscr{A}_\mathcal{H}$, i.e., Problem (10), we do not insert any regularization since data produced by the INT are not noisy, while for solving the minimization problem we opt for the ADAM algorithm [29], with a fixed batch size of 128, empowered by a scheduler that decreases the learning rate ρ by a factor ρ_f when training error does not decrease more than a threshold ρ_t for a given epoch number ρ_e. As a consequence the hyperparameters $\mathcal{H}$ of our algorithm $\mathscr{A}_\mathcal{H}$ are the following ones: the number of hidden units n_h, the activation φ, the initial learning rate ρ_0, the learning factor of the scheduler ρ_f, the threshold of the scheduler ρ_t, the epoch number of the scheduler ρ_e, and the total epochs n_e. Then in model selection and error estimation

			Hyperparameter	Ranges
			n_r	1
			$\lvert\mathcal{L}\rvert, \lvert\mathcal{V}\rvert, \lvert\mathcal{T}\rvert$	$70\%, 15\%, 15\%$
PIDDT$^\sigma$	PIDDT^0	DDT	n_h	$10^2, 10^{2.2}, \cdots, 10^4$
			φ	`relu`, `sigmoid`
			ρ_0	$10^{-6}, 10^{-5}, \cdots, 10^{-1}$
			ρ_f	$0.90, 0.95, 0.99$
			ρ_t	$10^{-7}, 10^{-6}, \cdots, 10^{-4}$
			ρ_e	$10, 15, 20$
			n_e	$1000, 2000, \cdots, 5000$
			λ_0	$10^{-4}, 10^{-6}, 10^{-8}, 10^{-10}$
			λ_c	$4, 5, 6, 7$
			λ_f	$10, 100$
			σ	$0.001, 0.01, 0.1, 1, 1.5, 2$

TABLE I: Hyperparameters ranges for DDT, PIDDT^0, and PIDDT$^\sigma$.

phase we set $n_r = 1$ using a sampler with no replacement, 70% of the data for learning 15% for validation, and 15% for test, and $\mathcal{S}$ as the Cartesian product of searching $n_h \in \{10^2, 10^{2.2}, \cdots, 10^4\}$, $\varphi \in \{\texttt{relu}, \texttt{sigmoid}\}$ where `relu` is the rectified linear unit and the `sigmoid` is the sigmoid function, $\rho_0 \in \{10^{-6}, 10^{-5}, \cdots, 10^{-1}\}$, $\rho_f \in \{0.90, 0.95, 0.99\}$, $\rho_t \in \{10^{-7}, 10^{-6}, \cdots, 10^{-4}\}$, $\rho_e \in \{10, 15, 20\}$, and $n_e \in \{1000, 2000, \cdots, 5000\}$.

A summary of the DDT hyperparameters ranges is reported in Table I.

C. The proposed Physics Informed Data-Driven Technique

For the PIDDT we inherit everything we described for the DDT, empowering it with the physical knowledge.

In the authors' opinion, this step is crucial since it allows us to compare the DDT with the PIDDT fairly. In fact, in most papers [13], [15], the DDT architecture (i.e., formulation of Problem (10), architectural choices, hyperparameter spaces, model selection phase, etc.) is different from the one of the PIDDT making the comparison unfair. In [14], instead, the architecture is the same, but the model selection phase is performed only for the DDT, fixing all the hyperparameters to the optimal choice for the DDT, and then the comparison with the PIDDT is performed with this hyperparameters configuration. This is again unfair since the optimal hyperparameters for the DDT may be different from the one of the PIDDT.

Our proposal to empower the DDT with the physical knowledge is to construct a custom regularizer $R(\Theta)$ to plug in Problem (10) that fully leverages Eqns. (3) and (4) (where Eq. (4) can be written as Eqns. (7), or (8), or (9)). In particular, we would like to underline three Facts

i Eqns. (3) and (4) tells us that the output (V and I in Eqns. (3) and (4) namely V_n^R, V_n^I, I_n^R, and I_n^I $\forall n \in \mathcal{N}$) of our model f_Θ based on its input (S namely P and Q that are P_n and Q_n $\forall n \in \mathcal{N}$) need to satisfy the constraint stated in the equations themselves, namely the following quantity

$$\|I - YV\| + \|S - V \odot I^H\|, \tag{14}$$

should be as small as possible (ideally it should tend to zero);

ii in order to collect $\mathcal{D}$ we need to run the INT, or in other words, we need to generate the data as explained in Section III-C. Note that $\forall(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}$ we basically have that $\boldsymbol{x} \equiv S \equiv P, Q$ and $\boldsymbol{y} \equiv V, I$;

iii in order to estimate the quantity Eq. (14) we do not need to run any INT, in fact we simply sample some feasible P and Q, compute V and I using f_Θ, and then we can compute the quantity Eq. (14).

Based on these Facts we can now formulate $R(\Theta)$ as follows

$$\begin{aligned} R(\Theta) = \lambda \textstyle\sum_{S\in\mathcal{D}} \big[&\|I(f_\Theta(S)) - YV(f_\Theta(S))\| \\ &+ \|S - V(f_\Theta(S)) \odot I^H(f_\Theta(S))\| \big], \end{aligned} \tag{15}$$

where λ is a hyperparameter which regulates how much we want to enforce the quantity Eq. (14) to be small and $I(f_\Theta(S))$ and $V(f_\Theta(S))$ are the I and V predicted by f_Θ when S is provided as input respectively. Note that the second term in the summation of Eq. (15) can be written in different ways considering Eqns. (7), or (8), or (9). For example if we exploit Eq. (9) we have

$$\begin{aligned} &\|S - V(f_\Theta(S)) \odot I^H(f_\Theta(S))\| \\ &= \sum_{n\in\mathcal{N}} \big[|P_n - V_n^R(f_\Theta(S)) I_n^R(f_\Theta(S)) - V_n^I(f_\Theta(S)) I_n^I(f_\Theta(S))| \\ &+ |Q_n + V_n^R(f_\Theta(S)) I_n^I(f_\Theta(S)) - V_n^I(f_\Theta(S)) I_n^R(f_\Theta(S))| \big]. \end{aligned} \tag{16}$$

Note that the regularizer of Eq. (15) is similar to the one that has been exploited in [14].

The primary limitation of this approach lies in its exclusive reliance on two of the previously identified Facts (Facts i and ii), thereby neglecting to effectively incorporate the third one (Fact iii). For this reason, in this work, we propose to modify regularizer of Eq. (15) as follows

$$\begin{aligned} R(\Theta) = \tfrac{\lambda}{2} \textstyle\sum_{S\in\mathcal{D}} \big[&\|I(f_\Theta(S)) - YV(f_\Theta(S))\| \\ &+ \|S - V(f_\Theta(S)) \odot I^H(f_\Theta(S))\| \big] \\ + \tfrac{\lambda}{2} \textstyle\sum_{\tilde{S}\in\{S+\mathtt{N}_S(\sigma): S\in\mathcal{D}\}} \Big[&\|I(f_\Theta(\tilde{S})) - YV(f_\Theta(\tilde{S}))\| \\ &+ \|\tilde{S} - V(f_\Theta(\tilde{S})) \odot I^H(f_\Theta(\tilde{S}))\| \Big], \end{aligned} \tag{17}$$

where $\mathtt{N}_S(\sigma)$ is a Gaussian noise of variance σ applied to all elements of S. Essentially, compared to Regularizer (15), Regularizer (17) makes more comprehensive use of Fact iii. This is because it enables a more precise estimation and exploration of the behavior of f_Θ with respect to the quantity defined in Eq. (14). This improved precision is achievable even when the cardinality of $\mathcal{D}$ is small, as the data can be augmented with a noise $\mathtt{N}_S(\sigma)$, eliminating the necessity to increase $\mathcal{D}$ and then run the INT multiple times. Moreover, when $\sigma = 0$, Regularizer (17) degenerates into Regularizer (15). For this reason we will refer as the classical PIDDT to the PIDDT with $\sigma = 0$ (PIDDT^0) and as the empowered PIDDT (PIDDT$^\sigma$) the PIDDT with σ considered as hyperparameter to tune.

At this stage, it is necessary to list the hyperparameters of the PIDDT which are the same as the ones of the DDT plus λ, and σ. Note that λ, theoretically, should be as large as possible so as to enforce the quantity of Eq. (14) to be zero. This cannot be done in practice since it would make it difficult to solve

the related optimization problem. For this reason, also for λ, we start from an initial value λ_0 and we set a scheduler that every ${}^{n_e}/_{\lambda_c}$ epochs, where λ_c is the number of changes of λ, increases λ of a factor of λ_f. Then in model selection and error estimation phase we leverage the same setting described for the DDT enriching $\mathcal{S}$ by making a Cartesian product with $\sigma \in \{0.001, 0.01, 0.1, 1, 1.5, 2\}$, $\lambda_0 \in \{10^{-4}, 10^{-6}, 10^{-8}, 10^{-10}\}$, $\lambda_c \in \{4, 5, 6, 7\}$, and $\lambda_f \in \{10, 100\}$.

A summary of the PIDDT hyperparameters ranges is reported in Table I.

V. Experimental Results

In this section we will compare the performance of DDT, PIDDT^0, and PIDDT$^\sigma$ (see Section IV) against the INT (see Section III-B) using the data described in Section III-C.

The performances, in terms of accuracy, will be measured in accordance with a quantitative metric, the Mean Absolute Percentage Error (MAPE), computed against the INT on the test set $\mathcal{T}$, which allows also to combine (average) over the different outputs (V_n^R, V_n^I, I_n^R, and I_n^I $\forall n \in \mathcal{N}$) of the model [30] and a qualitative metric, the scatter plot INT versus DDT, PIDDT^0, or PIDDT$^\sigma$ where all the outputs are normalized between $[0, 1]$ so to consider all output simultaneously [31]. For the sake of completeness, when space constraints make it possible, we will also report the accuracy result over the four groups of outputs (V^R, V^I, I^R, and I^I) using the MAPE and the Coefficient of Determination (R^2) [30].

The performances, in terms of computational requirements, will be measured by means of time to make a prediction of all the outputs (V_n^R, V_n^I, I_n^R, and I_n^I $\forall n \in \mathcal{N}$) given the inputs ($P_n$ and Q_n $\forall n \in \mathcal{N}$) since our surrogate will be leveraged to replace the INT so training time is unimportant while the forward phase is crucial.

The training, model selection (performance tuning), and error estimation (performance assessment) [27] have been performed following the procedure described in Section IV-A with the setting described in Table I. We performed all the experiments varying $n_d \in \{100, 1000, 1000\}$ and for $n_d \in \{100, 1000\}$ we repeated the experiment 30 times reporting mean and standard deviation of the result while, because of computational constraints, when $n_d = 10000$ experiment has been repeated just once.

All experiments have been run on the DelftBlue supercomputer at the Delft High-Performance Computing Center [32], which hosts 238 Compute nodes with a total of 476 Intel XEON E5-6248R 24C 3.0GHz CPUs and 192 GB of Memory per node and 10 GPU nodes with a total of 40 NVIDIA Tesla V100S 32GB GPUs and 256 GB of memory per node.

Table II reports, for the different values of n_d, the average MAPE over the different outputs of the DDT, the PIDDT^0, and the PIDDT$^\sigma$ against the INT together with the percentage of improvement of the PIDDT^0 over the DDT and the PIDDT$^\sigma$ over the PIDDT^0. From Table II it is possible to observe that, as expected from the theory, the more data (larger n_d) the better the models work. The PIDDT^0 is consistently more accurate than the DDT with peaks of improvements up to 50%. Also PIDDT$^\sigma$ is consistently more accurate than PIDDT^0 especially in the small sample (small n_d) regime with peaks of improvements up to 20%. Note also that the models start to become practical already when $n_d = 1000$ with errors below 1%.

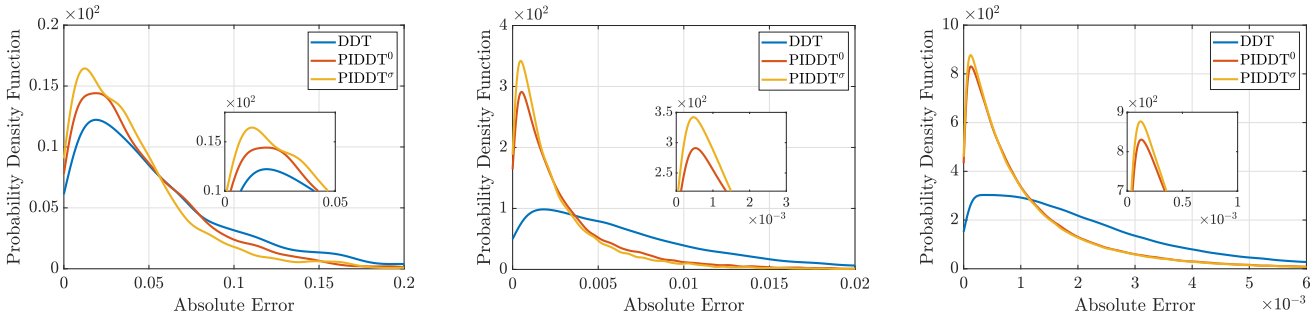
n_d	MAPE DDT	MAPE PIDDT^0	% Impr. DDT vs PIDDT^0	MAPE PIDDT$^\sigma$	% Impr. PIDDT^0 vs PIDDT$^\sigma$
100	5.68 ± 0.09	4.68 ± 0.13	18	3.84 ± 0.27	18
1000	0.77 ± 0.16	0.33 ± 0.01	57	0.28 ± 0.02	14
10000	0.23	0.12	48	0.11	2

TABLE II: Average MAPE over the different outputs of the DDT, the PIDDT^0, and the PIDDT$^\sigma$ against the INT together with the percentage of improvement of the PIDDT^0 over the DDT and the PIDDT$^\sigma$ over the PIDDT^0 for the different values of n_d.

Output	DDT	PIDDT^0	PIDDT$^\sigma$
MAPE			
V^R	0.2181	0.0910	0.0886
V^I	0.1443	0.0525	0.0499
I^R	0.2390	0.1500	0.1480
I^I	0.3259	0.1839	0.1824
R^2			
V^R	0.9821	0.9823	0.9823
V^I	0.9823	0.9824	0.9824
I^R	0.9996	0.9998	0.9998
I^I	0.9990	0.9996	0.9996

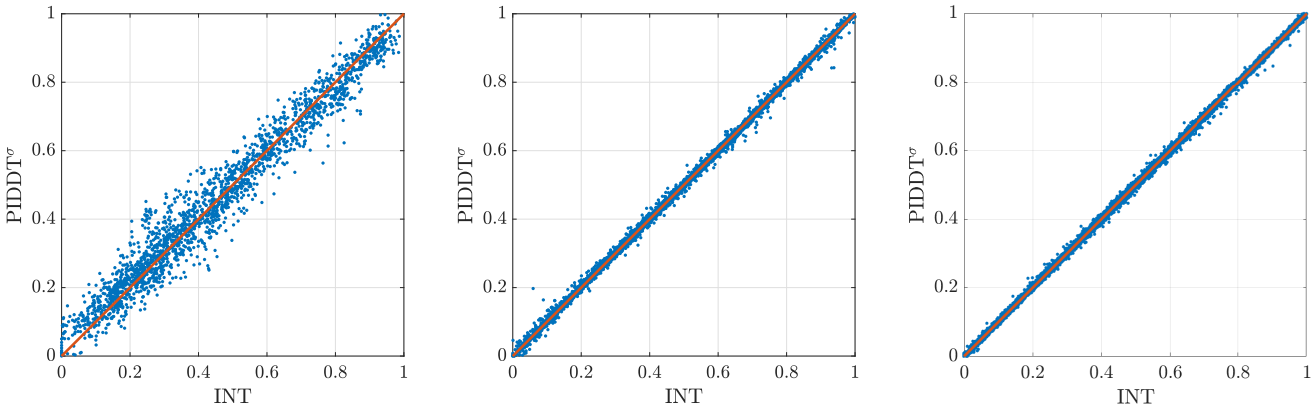
TABLE III: MAPE and R^2 of the DDT, the PIDDT^0, and the PIDDT$^\sigma$ against the INT, for $n_d = 10000$ and for each group of the outputs.

In order to get some more insight, with respect to the one derived from Table II, we reported a series of tables and graphs that shade some lights on other properties of the developed models. Table III reports, for $n_d = 10000$, namely the higher number of samples considered, and for each group of the outputs (V^R, V^I, I^R, and I^I) the MAPE and the R^2 of the DDT, the PIDDT^0, and the PIDDT$^\sigma$ against the INT. Table IV reports, for $n_d = 10000$, namely the higher number of samples considered, the optimal configuration of the hyperparameters for the DDT, the PIDDT^0, and the PIDDT$^\sigma$. Note that the blank cells mean that the particular hyperparameter does not apply to the particular model. Figure 3 reports, for the different values of n_d, the distribution of the absolute error of the DDT, the PIDDT^0, and the PIDDT$^\sigma$ against the INT. Figure 4 reports the scatter plot of the INT versus PIDDT$^\sigma$ (the best performing model), where all the outputs are normalized between $[0, 1]$ for the different values of n_d. From Tables III and IV and Figures 3 and 4 it is possible to derive a series of observations. From Table III, we can observe that all models work quite well on all outputs with small differences that can be mitigated, as future work, by balancing in a different way the errors on the different output during the training phase (see Eq. (10)). From Table IV, we can observe that some optimal configurations of the hyperparameters are actually the same across the different approaches (the DDT, the PIDDT^0, and the PIDDT$^\sigma$) reflecting the stability of the



(a) $n_d = 100$ (b) $n_d = 1000$ (c) $n_d = 10000$

Fig. 3: Distribution of the absolute error of the DDT, the PIDDT^0, and the PIDDT$^\sigma$ against the INT for the different values of n_d.



(a) $n_d = 100$ (b) $n_d = 1000$ (c) $n_d = 10000$

Fig. 4: Scatter plot of the INT versus PIDDT$^\sigma$ where all the outputs are normalized between $[0, 1]$ for the different values of n_d.

Hyperparameter	DDT	PIDDT^0	PIDDT$^\sigma$
n_r	1		
$\lvert\mathcal{L}\rvert, \lvert\mathcal{V}\rvert, \lvert\mathcal{T}\rvert$	70%, 15%, 15%		
n_h	1291	3162	3162
φ	`relu`	`relu`	`relu`
ρ_0	10^{-4}	10^{-4}	10^{-4}
ρ_f	0.90	0.95	0.95
ρ_t	10^{-4}	10^{-6}	10^{-6}
ρ_e	10	20	20
n_e	2000	5000	5000
λ_0		10^{-10}	10^{-10}
λ_c		7	7
λ_f		100	100
σ			0.01

TABLE IV: Optimal configuration of the hyperparameters for $n_d = 10000$ for the DDT, the PIDDT^0, and the PIDDT$^\sigma$.

results. Interesting to note that between the PIDDT^0 and the PIDDT$^\sigma$ the only optimal hyperparameter that changes is actually σ. Note also that the DDT requires much less weights, i.e., number of hidden neurons n_h, with respect to the PIDDT^0 and the PIDDT$^\sigma$: this reflects the fact that the PIDDT^0 and the PIDDT$^\sigma$ need also to embed the physical knowledge needing more representation power. Note also the PIDDT^0 and the PIDDT$^\sigma$ need more epochs to converge to the optimal solution with respect to the DDT further supporting our previous statement. From Figure 3, we can more clearly appreciate the improvements of the PIDDT^0 over the DDT and of the PIDDT$^\sigma$ over the PIDDT^0, the improvements and changes when increasing the number of samples n_d, and the fact that the distribution of the errors of the PIDDT^0 and the PIDDT$^\sigma$ is much more acceptable and plausible (light-tailed). Finally, from Figure 4, we can qualitatively observe the remarkable improvements of the PIDDT$^\sigma$ when increasing the number of samples n_d and the fact that already for $n_d = 1000$ the model becomes practical.

Finally Table V reports, for the different values of n_d, the time needed to make a prediction of all the outputs (V_n^R, V_n^I, I_n^R, and I_n^I $\forall n \in \mathcal{N}$) given the inputs ($P_n$ and Q_n $\forall n \in \mathcal{N}$) for the INT (which of course does not depend on n_d), the DDT, the PIDDT^0, and the PIDDT$^\sigma$. From Table V it is possible to observe that the INT is up to 5 order of magnitude slower than the DDT, the PIDDT^0, and the PIDDT$^\sigma$, then the latter are

INT	n_d	DDT	PIDDT^0	PIDDT$^\sigma$
15 s	100	0.06 ± 0.01 ms	0.07 ± 0.01 ms	0.19 ± 0.01 ms
	1000	0.09 ± 0.01 ms	0.07 ± 0.01 ms	0.08 ± 0.01 ms
	10000	0.09 ± 0.01 ms	0.09 ± 0.01 ms	0.11 ± 0.01 ms

TABLE V: Time needed to make a prediction of all the outputs (V_n^R, V_n^I, I_n^R, and I_n^I $\forall n \in \mathcal{N}$) given the inputs ($P_n$ and Q_n $\forall n \in \mathcal{N}$) varying n_d for the INT, the DDT, the PIDDT^0, and the PIDDT$^\sigma$.

much more practical and effective not having also the problem of convergence of the INT. Note that the differences between the DDT, the PIDDT^0, and the PIDDT$^\sigma$ are negligible in terms of time to make a prediction.

VI. Conclusions

The power grid complexity has significantly increased over the past decade, primarily due to the integration of diverse distributed energy resources. To ensure grid stability, reliable operation, strategic planning, and effective market strategies, the development of accurate and efficient power flow analysis tools has become crucial. While current state-of-the-art approaches rely on iterative numerical techniques that offer high accuracy, their convergence speed is often slow or even nonexistent. As a result, researchers have explored data-driven techniques as an alternative solution, despite their relatively lower accuracy compared to iterative methods. These data-driven techniques offer the advantage of exceptional speed. However, they have been limited by two main factors: a failure to fully exploit the existing physical knowledge and a lack of fair comparisons between different approaches. In this study, we propose a novel physics informed data-driven model that overcomes these limitations. By integrating domain knowledge into the data-driven model, constraining it and augmenting the available data, we were able to outperform current approaches especially in small sample scenarios. Furthermore, we introduce a comprehensive framework that enables a fair comparison of various approaches, thus demonstrating the true effectiveness of our proposal. Our experimental results on the IEEE 57 realistic power network provide strong support for our proposition. Of course this work is a first step toward the solution of the power flow problem and larger power networks need to be tested to support the quality of the proposal.

References

[1] N. Mahmud and A. Zahedi, "Review of control strategies for voltage regulation of the smart distribution network with high penetration of renewable distributed generation," *Renewable and Sustainable Energy Reviews*, vol. 64, pp. 582–595, 2016.

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

[3] H. Chen, J. Chen, D. Shi, and X. Duan, "Power flow study and voltage stability analysis for distribution systems with distributed generation," in *IEEE power engineering society general meeting*, 2006.

[4] B. Stott, "Review of load-flow calculation methods," *Proceedings of the IEEE*, vol. 62, no. 7, pp. 916–929, 1974.

[5] A. Potter, R. Haider, G. Ferro, M. Robba, and A. M. Annaswamy, "A reactive power market for the future grid," *Advances in Applied Energy*, vol. 9, p. 100114, 2023.

[6] G. Ferro, M. Robba, D. D'Achiardi, R. Haider, and A. M. Annaswamy, "A distributed approach to the optimal power flow problem for unbalanced and mesh networks," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 13 287–13 292, 2020.

[7] M. Karimi, A. Shahriari, M. R. Aghamohammadi, H. Marzooghi, and V. Terzija, "Application of newton-based load flow methods for determining steady-state condition of well and ill-conditioned power systems: A review," *International Journal of Electrical Power & Energy Systems*, vol. 113, pp. 298–309, 2019.

[8] S. Iwamoto and Y. Tamura, "A load flow calculation method for ill-conditioned power systems," *IEEE transactions on power apparatus and systems*, no. 4, pp. 1736–1743, 1981.

[9] Y. Liu, N. Zhang, Y. Wang, J. Yang, and C. Kang, "Data-driven power flow linearization: A regression approach," *IEEE Transactions on Smart Grid*, vol. 10, no. 3, pp. 2569–2580, 2018.

[10] L. Guo, Y. Zhang, X. Li, Z. Wang, Y. Liu, L. Bai, and C. Wang, "Data-driven power flow calculation method: A lifting dimension linear regression approach," *IEEE Transactions on Power Systems*, vol. 37, no. 3, pp. 1798–1808, 2021.

[11] J. Yu, Y. Weng, and R. Rajagopal, "Mapping rule estimation for power flow analysis in distribution grids," *arXiv preprint arXiv:1702.07948*, 2017.

[12] S. Shalev-Shwartz and S. Ben-David, *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[13] H. H. Müller, M. J. Rider, and C. A. Castro, "Artificial neural networks for load flow and external equivalents studies," *Electric power systems research*, vol. 80, no. 9, pp. 1033–1041, 2010.

[14] Y. Yang, Z. Yang, J. Yu, B. Zhang, Y. Zhang, and H. Yu, "Fast calculation of probabilistic power flow: A model-based deep learning approach," *IEEE Transactions on Smart Grid*, vol. 11, no. 3, pp. 2235–2244, 2019.

[15] X. Hu, H. Hu, S. Verma, and Z. Zhang, "Physics-guided deep neural networks for power flow analysis," *IEEE Transactions on Power Systems*, vol. 36, no. 3, pp. 2082–2092, 2020.

[16] B. Huang and J. Wang, "Applications of physics-informed neural networks in power systems-a review," *IEEE Transactions on Power Systems*, 2022.

[17] G. Karniadakis, I. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, vol. 3, no. 6, pp. 422–440, 2021.

[18] R. D. Zimmerman, C. E. Murillo-Sánchez, and R. J. Thomas, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on power systems*, vol. 26, no. 1, pp. 12–19, 2010.

[19] A. Karami and M. Mohammadi, "Radial basis function neural network for power system load-flow," *International Journal of Electrical Power & Energy Systems*, vol. 30, no. 1, pp. 60–66, 2008.

[20] V. Bolz, J. Rueß, and A. Zell, "Power flow approximation based on graph convolutional networks," in *IEEE international conference on machine learning and applications*, 2019.

[21] M. Xiang, J. Yu, Z. Yang, Y. Yang, H. Yu, and H. He, "Probabilistic power flow with topology changes based on deep neural network," *International Journal of Electrical Power & Energy Systems*, vol. 117, p. 105650, 2020.

[22] A. Primadianto and C. N. Lu, "A review on distribution system state estimation," *IEEE Transactions on Power Systems*, vol. 32, no. 5, pp. 3875–3883, 2016.

[23] W. H. Kersting, *Distribution system modeling and analysis*. CRC press, 2017.

[24] R. Alizadeh, J. K. Allen, and F. Mistree, "Managing computational complexity using surrogate models: a critical review," *Research in Engineering Design*, vol. 31, pp. 275–298, 2020.

[25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[26] C. C. Aggarwal, *Neural networks and deep learning*. Springer, 2018.

[27] L. Oneto, *Model Selection and Error Estimation in a Nutshell*. Springer, 2020.

[28] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural computation*, vol. 8, no. 7, pp. 1341–1390, 1996.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[30] M. Z. Naser and A. H. Alavi, "Error metrics and performance fitness indicators for artificial intelligence and machine learning in engineering and sciences," *Architecture, Structures and Construction*, pp. 1–19, 2021.

[31] K. L. Sainani, "The value of scatter plots," *PM&R*, vol. 8, no. 12, pp. 1213–1217, 2016.

[32] Delft High Performance Computing Centre (DHPC), "DelftBlue Supercomputer (Phase 1)," https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1, 2022.