Electric Power Systems Research 223 (2023) 109551

Contents lists available at ScienceDirect

Electric Power Systems Research

journal homepage: www.elsevier.com/locate/epsr

Gradient-enhanced physics-informed neural networks for power systems operational support

Mostafa Mohammadian [a,*], Kyri Baker [a], Ferdinando Fioretto [b]

[a] Civil, Environmental and Architectural Engineering, College of Engineering and Applied Science, University of Colorado Boulder, Boulder, 80309, CO, USA
[b] Electrical Engineering and Computer Science, Syracuse University, Syracuse, 13244, NY, USA

ARTICLE INFO

Keywords:
Deep learning
Power system dynamics
Physics-informed neural networks
Optimal power flow
Transfer learning

ABSTRACT

The application of deep learning methods to speed up the challenging power system problems has recently shown very encouraging results. However, power system dynamics are not snapshot, steady-state operations. These dynamics must be considered to ensure that the optimal solutions provided by these models adhere to practical constraints to avoid frequency fluctuations and grid instabilities. Unfortunately, dynamic system models based on ordinary or partial differential equations are frequently unsuitable for direct application in control or state estimates due to their high computational costs. To address these challenges, this paper introduces a machine learning method to approximate the behavior of power systems dynamics in near real-time. The proposed framework is based on gradient-enhanced physics-informed neural networks (gPINNs) and encodes the underlying physical laws governing power systems. A key characteristic of the proposed gPINN is its ability to train without the need of generating expensive training data. The paper illustrates the potential of the proposed approach in both forward and inverse problems in a single-machine infinite bus system and a three-bus power network for predicting rotor angles and frequency, and uncertain parameters such as inertia and damping to showcase its potential for a range of power systems applications. The model exhibited high accuracy in predicting the variables, achieving a range of 0.533–4.092 and an average L^2 relative error improvement of up to 13.30× compared to the PINN model. The computational performance of the proposed gPINN model was compared to a conventional solver, revealing a remarkable speed-up of 31 to 171 times faster in solving differential–algebraic systems of equations in power systems.

1. Introduction

Recently, electric power networks have incorporated increasing levels of renewable energy sources, further deregulated markets, and intricate communication and control systems have been implemented to increase stability and efficiency. Nonetheless, these advancements have resulted in a broader range of operating scenarios and potential risks that could pose a threat to the security of the power network [1,2]. On the other hand, these advancements provide new opportunities to utilize an abundance of measurements from the grid. Hence, with the dramatic growth of available data and computing resources, there has been a revolution in the successful application of Artificial Neural Networks (ANN), also commonly referred to as Deep Neural Networks (DNN) and Deep Learning (DL), in power systems [3].

Although some recent works have been focused on using machine learning to solve power systems' operational challenges at a fraction of the time required by traditional approaches, they have fundamental flaws; e.g., not considering dynamics, not being generalizable, and mostly being applied to steady state power flow. This is largely due to harnessing the benefit of historical data and moving most of the computational burden to an off-line setting [4–9]. In other words, power systems are dynamic (e.g., generator dynamics affect the stability of system operations). Therefore, their operations cannot be reliably controlled when considering steady-state behaviors alone.

In power systems, simulating the dynamic response of a power network requires solving a set of complex nonlinear differential–algebraic equations (DAEs) [10]. This task, however, is challenging as traditional explicit integration methods fail to produce accurate results [11]. Therefore, commercial solvers use numerically stable methods to integrate dynamic equations and iterative schemes to solve algebraic equations [12]. The cost and memory required to integrate DAEs are substantial and pose a hindrance to performing real-time dynamical assessments [13]. However, with the ongoing transformation of the power network, it will become increasingly crucial for electric utilities to perform real-time dynamical assessments, necessitating faster methods of integrating and simulating DAEs.

* Corresponding author.
E-mail address: momo6802@colorado.edu (M. Mohammadian).

https://doi.org/10.1016/j.epsr.2023.109551
Received 10 April 2023; Received in revised form 12 May 2023; Accepted 10 June 2023
Available online 19 June 2023

Nomenclature

$\boldsymbol{\theta}$	Network weights and biases
δ	Voltage angles behind the transient reactance (rad)
λ	System parameter
$\mathcal{B}$	Initial condition of an ODE
$\mathcal{N}$	Nonlinear differential operator
$\mathcal{N}_b$	Set of initial training points
$\mathcal{N}_f$	Set of residual training points
$\mathcal{N}_g$	Set of residual points associated with the derivative $\frac{\partial f}{\partial t_i}$
$\mathcal{N}_i$	Set of measurements points
ω	Angular frequency (rad/s)
B_∞	Standard infinite bus
d_g	Damping coefficient of generator
E_g	Voltage behind the transient reactance of generator (p.u.)
E_∞	Infinite bus voltage (p.u.)
f	Differential equation residual
m_g	Moment of inertia of generator
P_e	Electrical power output (p.u.)
P_m	Input shaft power (p.u.)
t	Time (s)
u	State of the dynamic system
w_f, w_b, w_i, w_g	Loss component weights
x_g	Synchronous generator reactance (p.u.)
x_l	Transmission line reactance (p.u.)

In recent years, scientific machine learning has introduced new innovative approaches aimed at understanding the differential equations that describe dynamic systems, thus providing a more efficient alternative to traditional and expensive numerical solvers. Despite the success of these methods, the current application of a DL-based framework for learning and simulating the solution trajectories of the dynamic behavior of a power network is limited to the single-machine infinite bus (SMIB) system. These methods lack robustness and generalization capabilities (e.g., different initial conditions or states of the system) [14]. Therefore, in this paper, a method that utilizes physics-informed neural networks to simulate power systems dynamics in multi-machine systems is developed while not requiring much training data and being more generalizable to the different initial conditions.

The remainder of this paper is organized as follows. Sections 2 and 3 are dedicated to the related works and goals of the paper, respectively. In Section 4, after introducing the physics-informed neural network's architecture, the extension to gPINN for the power system and theoretical intuitions for the enhanced performance over PINN models is presented. Section 5 describes the employed power system models including the single machine infinite bus (SMIB) system and three-bus power network used for the prediction task. In Section 6, the simulation results demonstrating the performance of the physics-informed neural networks and the gPINN models are presented. Finally, conclusions and future work are given in Section 7.

2. Related works

In physics and engineering, many crucial physical models are expressed as partial differential equations (PDE), such as Navier–Stokes equations [15] in fluid mechanics and Maxwell equations [16] in electromagnetic field theory. Conventional numerical methods used for solving PDEs can encounter challenges such as high computation costs, technical difficulties, and the curse of dimensionality. Recently, scientific machine learning has delivered new ground-breaking contributions, aiming at learning the differential equations describing dynamical systems and, hence providing us with an efficient alternative to traditional costly numerical solvers [17–19].

In 2018, Karniadakis and his team introduced the idea of physics-informed neural networks (PINNs) [17], which approximates PDEs by integrating their governing equations and their initial and boundary conditions into the neural network's loss function. These models heavily exploit automatic differentiation [20] – a key feature of deep learning frameworks such as Tensorflow [21] and PyTorch [22] – to compute partial derivatives with respect to quantities of interest. In contrast to traditional supervised learning methods, PINNs do not require labels, but training data can be generated *on the fly* and self-supervision is provided by virtue of approximating the known PDEs governing behavior and initial and boundary conditions. Additionally, and differently from traditional numerical methods used for solving PDEs, PINNs do not require discretizing the input space, thereby avoiding the curse of dimensionality [23], and provide significant benefits for solving forward and inverse problems of PDEs. Generally, they have the ability to solve systems of differential–algebraic equations in a fraction of the time required by traditional methods, to directly determine the value of state variables at any time instant t_1 (without the need to integrate from t_0 to t_1), and to solve higher-order differential equations without the need to introduce additional variables to solve a first-order system.

Recently, a small number of studies have explored the adoption of PINNs in power systems. One notable study by Misyris et al. [14] used a physics-informed continuous deep learning model to obtain the unknown parameters of a power network, demonstrating its potential with a single-machine infinite bus system. However, this PINN faced fundamental challenges: while it is able to predict with relatively good accuracy the voltage angle variables when the system is stable, the method fails to provide sufficient accuracy in unstable or oscillating cases, which are crucial for power system reliability assessments. This approach also suffers from fundamental efficiency problems, e.g., when it is required to predict multiple states (multiple ODEs). Finally, is also unclear if the framework learned the stiff nonlinear dynamical equations or a non-stiff ODE-based approximation of power network dynamics.

In another study [24], a DAE-PINN framework was developed to learn and simulate the solution trajectories of nonlinear differential–algebraic equations. This method effectively utilized the synergy between implicit Runge–Kutta time-stepping schemes and PINNs. However, it did not address the applicability to different system states or the ability to generalize to different initial conditions within a reasonable computation time.

This paper departs from previous contributions by addressing these issues and thoroughly validating the performance of a class of PINN networks suitable for solving power system problems.

3. Settings and goals — Departure from existing works

While progress has been made in physics-informed deep learning, these methods are not yet scalable enough to serve as accurate models for large-scale dynamic systems. To achieve this objective, the models must possess two critical capabilities: *(i)* accurately predicting solution trajectories for a diverse set of initial conditions and *(ii)* maintaining physical accuracy over extended time periods. Therefore, this paper, not only considers a single machine infinite bus case [14], but also solves stiff dynamical equations for entire networks with multiple generators, observing the evolution of multiple system states simultaneously. This ability to solve stiff dynamical equations for an entire network with multiple generators challenges previous models but is crucial in modern electrical power systems, which are becoming increasingly complex and interconnected.

To enhance the generalization of neural networks in power system applications, the use of Gradient-enhanced Physics-informed neural networks (gPINNs) is investigated [25]. Transfer learning is employed to reduce computational costs and enable the model to generalize to various initial conditions. This technique significantly lowers computation time, which is a current limitation of PINN models in power system applications. Residual-based adaptive refinement (RAR) and an improved fully-connected architecture for gPINN are introduced to enhance neural network performance. Additionally, the paper also proposes a variation of the gPINN formulation in order to integrate initial conditions effectively, even with a significant number of ODEs. This approach is applied to solve forward and inverse problems of ordinary differential equations, specifically for power system dynamics equations.

It is worth noting that the methodologies studied in this work are intended to serve as an *aid* to human dispatchers, rather than as a direct replacement for current controllers. By leveraging these models, operators can rapidly simulate a broad range of dynamic system conditions, enabling them to make well-informed decisions during operation and, ultimately, enhance security and decrease costs. Consequently, the results produced by our model may be utilized to implement preventive measures focused on operations, which dispatchers can utilize to steer clear of critical scenarios.

The main contributions of this paper center around demonstrating the efficacy of gPINN models in both finding solutions to the swing equation and a three-bus network and estimating unknown parameters. In more detail:

1. A model which uses deep learning to reliably calculate solutions to the swing equation is developed, as a result, can rapidly obtain values such as load angle and frequency in the swing equation and a three-bus power network example described by a set of stiff and nonlinear dynamical equations;
2. The model is capable of accurately estimating uncertain power system parameters from limited measurements, which becomes increasingly important as generators age and parameters naturally change;
3. Lastly, the proposed model can quickly generalize to different initial conditions through transfer learning, meaning that operators can use the model as an advisory tool to simulate a wide variety of possible system states.

4. gPINNs for power system application

4.1. Methods

In this section, an overview of physics-informed neural networks (PINNs) is provided, followed by the introduction of the method of gradient-enhanced PINNs. The aim is to enhance the accuracy and training efficiency of PINNs within the given context

The PINNs aim at approximating the solution of a system of one or more differential, possibly non-linear equations, by explicitly encoding the differential equation formulation in the neural network. For generality, let us consider the following parametrized and nonlinear ordinary differential equation (ODE):

$$\begin{aligned} &f(t) := \boldsymbol{u}(t)_t + \mathcal{N}[\boldsymbol{u}(t), \lambda] = 0, \qquad t \in [0, T] \\ &\mathcal{B}(\boldsymbol{u}(0)) = 0, \end{aligned} \tag{1}$$

where $\boldsymbol{u}(t)$ denotes the state of the dynamic system (the latent ODE solution), $\mathcal{N}[\cdot;\lambda]$ is a nonlinear differential operator connecting the state variables $\boldsymbol{u}$ with the system parameters λ and operator $\mathcal{B}$ denotes the initial condition of an ordinary differential equation. When λ is unknown in the context of approximating the solution of function (1), it becomes a problem of system identification. The goal is to find parameters λ that satisfy the expression in (1).

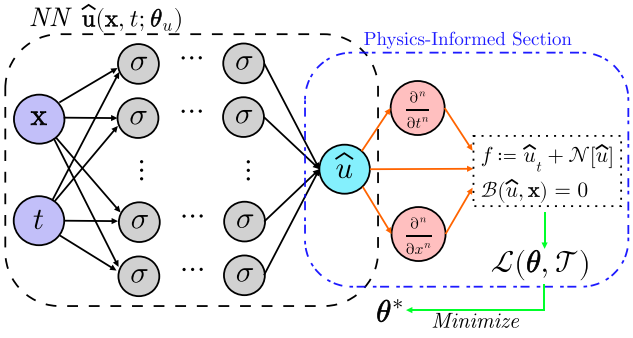

Fig. 1. Schematic of a typical PINN.

The physics-informed neural networks use automatic differentiation to embed the partial differential equation and its initial boundary conditions to the neural network loss function. The goal is to find the optimal weights of the model that minimize the defined function of loss. Mean Square Error (MSE) is commonly used to construct the loss function for Eq. (1), which is defined as follows [17]:

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{N}) = w_f \mathcal{L}_f(\boldsymbol{\theta}; \mathcal{N}_f) + w_b \mathcal{L}_b(\boldsymbol{\theta}; \mathcal{N}_b), \tag{2}$$

where

$$\mathcal{L}_f(\boldsymbol{\theta}; \mathcal{N}_f) = \frac{1}{|\mathcal{N}_f|} \sum_{t \in \mathcal{N}_f} |f(t)|^2$$

$$\mathcal{L}_b(\boldsymbol{\theta}; \mathcal{N}_b) = \frac{1}{|\mathcal{N}_b|} \sum_{t \in \mathcal{N}_b} |\mathcal{B}(\hat{\boldsymbol{u}})|^2,$$

here w_f and w_b are weights associated with the two loss components and $\mathcal{N}_f$ and $\mathcal{N}_b$ represent the number of ODE residual training points and the number of initial training points, respectively. Specifically, the loss $\mathcal{L}_f$ enforces the physics of the dynamical system by Eq. (1) at a finite set of collocation points and $\mathcal{L}_b$ corresponds to the initial conditions Fig. 1 shows the general architecture of a sample PINN. One key advantage of PINNs is that the same formulation can be utilized not only for forward problems but also for inverse PDE-based problems. When the parameter in Eq. (1) is unknown and additional measurements $\boldsymbol{u}$ are available on the set of points $\mathcal{N}_i$, it has been demonstrated in [26] that an extra data loss term is included.

$$\mathcal{L}_i(\boldsymbol{\theta}, \lambda; \mathcal{N}_i) = \frac{1}{|\mathcal{N}_i|} \sum_{t \in \mathcal{N}_i} |\hat{\boldsymbol{u}}(t) - \boldsymbol{u}(t)|^2. \tag{3}$$

To simultaneously learn the unknown parameters with the solution $\boldsymbol{u}$, then our new loss function is described as

$$\mathcal{L}(\boldsymbol{\theta}, \lambda; \mathcal{N}) = w_f \mathcal{L}_f(\boldsymbol{\theta}, \lambda; \mathcal{N}_f) + w_b \mathcal{L}_b(\boldsymbol{\theta}, \lambda; \mathcal{N}_b) + w_i \mathcal{L}_i(\boldsymbol{\theta}, \lambda; \mathcal{N}_i).$$

The approach in physics-informed neural networks is to enforce the differential equation residual, f, to be zero. Nonetheless, due to the nature of $f_\theta(t)$ being zero for any input t, it follows that the derivatives of f are also zero within the simulation domain. To address this limitation, the recent development of gradient-enhanced PINNs proposes enforcing the derivatives of the differential equation residual to be zero as well, as introduced by Yu et al. [25], i.e.,

$$\frac{\partial f}{\partial t} = 0, \qquad t \in [0, T] \tag{4}$$

Then the loss function of gPINN thus becomes:

$$\mathcal{L} = w_f \mathcal{L}_f + w_b \mathcal{L}_b + w_i \mathcal{L}_i + \sum_{i=1}^{D} w_{g_i} \mathcal{L}_{g_i}(\boldsymbol{\theta}; \mathcal{N}_{g_i}), \tag{5}$$

where the loss of the derivative $\mathcal{L}_{g_i}(\boldsymbol{\theta}; \mathcal{N}_{g_i})$ with respect to x_i is defined as

$$\mathcal{L}_{g_i}(\boldsymbol{\theta}; \mathcal{N}_{g_i}) = \frac{1}{|\mathcal{N}_{g_i}|} \sum_{t \in \mathcal{N}_{g_i}} \left| \frac{\partial f}{\partial t_i} \right|^2.$$

In this context, the set $\mathcal{N}_{g_i}$ refers to the residual points associated with the derivative $\frac{\partial f}{\partial t_i}$, which may differ from the set $\mathcal{N}_f$. Our simulation results demonstrate that by enforcing the gradient of the PDE residual function, the gPINN formulation offers enhanced accuracy for predicting solutions to $\boldsymbol{u}$, requiring fewer training points compared to other methods. In addition, gPINN significantly improves the accuracy of predicted solutions for $\frac{\partial \boldsymbol{u}}{\partial t_i}$, a crucial aspect of our study.

In large-scale dynamical systems with numerous differential equations to learn, it becomes beneficial to reduce the number of terms in the loss function to avoid vanishing gradient issues and enhance the satisfaction of initial constraints. To accomplish this, the loss term for initial conditions can be generalized as follows:

$$\mathcal{L}_b(\theta; \mathcal{N}_b) = \frac{1}{|\mathcal{N}_b|} \sum_{t \in \mathcal{N}_b} |\boldsymbol{u}(\boldsymbol{x}_j) - \hat{\boldsymbol{u}}(\boldsymbol{x}_j)|^2 \tag{6}$$

where the pair $(\boldsymbol{x}_j, \hat{\boldsymbol{u}})$ corresponds to the jth training example and $\boldsymbol{x}_j = (t, \boldsymbol{u}(0))_j$ is the whole input to the network (i.e., time and initial state). This dataset contains only the initial conditions of the modeled ODE. An example of a training data pair is $((0, 0.5), 0.5)$, which indicates that the initial state at time $t = 0$ is 0.5 and the desired output is also 0.5. Note that, with this approach, the training set includes an input value $u_j(0)$ equal to the desired output value $\hat{u}_j$. Thus, the neural network learns to reproduce the initial state $u_j(0)$ in its output value $u(x_j)$ at $t = 0$ in a supervised learning setting, as opposed to using automatic differentiation. This not only simplifies the training process but also reduces the computational burden.

4.2. Why gPINNs provides enhanced performance: Theoretical intuitions

In contrast to a standard Physics-Informed Neural Network, a gradient-enhanced PINN incorporates both the values and gradients of the partial differential equations being solved. This additional information provides insight into the underlying physics, allowing for a more precise and robust solution to stiff differential equations, often encountered in power systems dynamical studies.

Using gradient information in the loss function has several benefits over traditional PINN:

- First, it reduces the risk of overfitting and increases the network's convergence speed, improving overall performance.
- Second, gPINN is less sensitive to the choice of activation functions and network topology, making it more robust and reliable than standard PINN.
- Third, gPINN can handle more complex boundary and initial conditions, as well as multi-field PDEs with multiple variables and gradients, making it more versatile.
- Finally, gPINN can improve the accuracy of solutions in areas with sparse or unreliable data by aiding in the regularization of the network.

To comprehend this concept, the reference is made to the *Rademacher Complexity Bound theorem*, which presents a mathematical framework for bounding the expected difference between the empirical risk and the true risk of a function class. The Rademacher complexity bound provides a measure of the generalization performance of a function class, that is, how well the function class can generalize to new data beyond the training data. A smaller Rademacher complexity bound indicates a better generalization performance, as it suggests that the function class is less sensitive to random noise in the training data.

One way to reduce the Rademacher complexity of a function class is to introduce additional constraints or penalties that encourage simpler or smoother functions. For instance, this term can be included in the loss function to penalize large weights or biases, which can result in sparser or smoother functions. In gradient-enhanced PINNs, an additional gradient component is used to help enhance the generalization ability of the resulting PINNs.

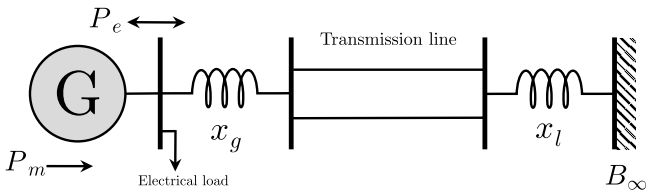

Fig. 2. Configuration of a single machine infinite bus system.

5. Physical model for power system dynamics

This section provides an overview of the two system models utilized to assess the performance of the proposed method.

5.1. Single machine infinite bus system

In three-phase, high-voltage power transmission systems, synchronous generators of the system accelerate or decelerate with respect to the synchronously rotating air gap magnetomotive force, to adapt to changing power transfer requirements that occur during system disturbances. In electrical power systems networks, the frequency varies constantly based on system dynamics. Modeling power system dynamics from oscillations and transients using time-synchronized measurements can provide real-time information, including angular displacements, voltage and current phasors, frequency changes, and rate of signal system decay from positive-sequence components that can help system operators or dispatchers in decision-making.

In general, the dynamics of a generator are defined by its internal voltage phasor. In the context of transient stability assessment, the internal voltage magnitude is usually considered to be constant due to its slow variation in comparison to its angle [27]. Therefore, power system dynamics are described by the swing equation in their simplest and most common form, neglecting transmission losses and bus voltage deviations. As such, for each generator k, the resulting swing equation that governs the dynamics of the synchronous generator can be described by [14]

$$m_{g_k} \ddot{\delta}_k + d_{g_k} \dot{\delta}_k + P_{e_k} - P_{m_k} = 0, \tag{7}$$

where $m_k > 0$ is the dimensionless moment of inertia of the generator, $d_k > 0$ represents primary frequency controller action on the governor (called damping coefficient), P_{m_k} is the input shaft power producing the mechanical torque acting on the rotor of the generator, P_{e_k} is the electrical power output in per unit, δ_k denotes the voltage angles behind the transient reactance, and $\dot{\delta}_k$ is the angular frequency of the kth generator, often also represented as ω_k.

The single machine infinite-bus (SMIB) system, as depicted in Fig. 2, where G is a synchronous generator and B_∞ denotes a standard infinite bus, has been widely used to perceive and analyze the fundamental dynamic phenomena occurring in power systems. The swing equation (7) for the SMIB system can be written as follows:

$$m_g \ddot{\delta} + d_g \dot{\delta} + E_g E_\infty B_{g\infty} \sin(\delta) - P_m = 0, \tag{8}$$

where δ is the phase difference between the voltage vector of the generator and the infinite bus, E_g is the voltage behind the transient reactance of the generator, E_∞ is the infinite bus voltage, and $B_{g\infty} := \frac{1}{x_g + x_l}$ is the combined susceptance in between (x_g and x_l are the synchronous generator and transmission line reactance, respectively). This study demonstrates the accurate estimation of rotor angle δ and angular frequency $\dot{\delta}$ (or ω) of the swing equation (8) at any given time t for a range of mechanical power P_m using gradient-enhanced physics-informed neural networks. Furthermore, the framework can effectively identify uncertain parameters, such as m_g and d_g.

At the first step for solving the forward problem, it is assumed that the system parameters for a given synchronous generator $\lambda := \{m_g, d_g\}$

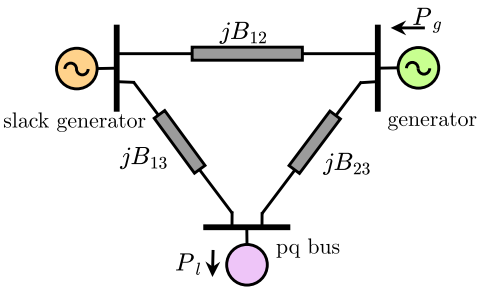

Fig. 3. Three-bus, two-generator power network.

are known priori and the voltages E_g and E_∞ are fixed and constant. As a result, the system input to our proposed gPINN (and PINN) model is defined as t (in our simulation domain). Despite conventional numerical solvers in simulating multi-physics problems, which necessitate the conversion of higher-order ordinary differential equations (ODEs) to first-order in order to solve them (by introducing additional variables), gPINNs can directly incorporate higher-order ODEs, as it will be shown in (10). Incorporating (8) to the neural network in Section 4.1, functions (9) and (10) are given (with the time index omitted for clarity) as:

$$u(t) := \delta(t), \tag{9}$$

$$f_{\boldsymbol{\theta}}(t) = m_g\,\ddot{\delta} + d_g\,\dot{\delta} + E_g E_\infty B_{g\infty}\sin(\delta) - P_m = 0, \tag{10}$$
$$P_m \in [P_{\min}, P_{\max}], t \in [0, T]$$

The interval $[0, T]$ can be specified based on the time horizon of interest for the dynamic simulation. The domain Ω of the input P_m is restricted to $[P_{\min}, P_{\max}]$ due to the capability of the generator. The neural network output is only $\delta(t)$, rotor angular displacement with respect to the synchronous reference axis. After the training phase, the angular frequency signal $\omega := \dot{\delta}$ is extracted as a function of the estimated angle δ using the automatic differentiation of the same neural network. As a result, the prediction error of the frequency ω depends on the prediction error of the angular position δ and the differential method.

In the second part, the system parameters $\lambda := \{m_g, d_g\}$ are also the problem of interest to be determined by the gPINN model as well as angular position and frequency. For system operators to avoid substantial frequency deviations and preserve frequency stability, information about power system factors such as system inertia is critical. Varying power-infeed from converter-based generation units introduces great uncertainty on system parameters such as inertia and damping and, therefore, has to be estimated (or predicted) at regular time intervals [28]. The problem of system identification and data-driven discovery of partial differential equations can be addressed with gradient-enhanced physics-informed neural networks. Therefore, m_g and d_g are defined as unknown parameters in Eq. (10). Here, the topology of the gradient-enhanced physics-informed neural network stays unchanged, with the exception that when minimizing (5) during neural network training, a subset of the system parameters are now handled as new variables.

5.2. Three-bus power network

This section contains a systematic study of a three-bus power network with the goal of illustrating the performance of our framework. The three-bus power network shown in Fig. 3 is considered, and it is described by the following set of nonlinear and stiff dynamical equations:

$$\dot{\omega}_1 = \frac{1}{M_1}(-d\,\omega_1 + f_1 + f_2) \tag{11a}$$

$$\dot{\omega}_2 = \frac{1}{M_2}(-d\,\omega_2 - f_1) \tag{11b}$$

$$\dot{\delta}_2 = \omega_2 - \omega_1 \tag{11c}$$

$$\dot{\delta}_3 = -(\omega_1 - \frac{f_2}{d_l}), \tag{11d}$$

where $(\omega_1, \omega_2, \delta_2, \delta_3)^{\mathrm{T}}$ are the dynamic states, and

$$f_1 = B_{12}V_1V_2\sin(\delta_2) + B_{23}V_2V_3\sin(\delta_2 - \delta_3) + P_g$$
$$f_2 = B_{13}V_1V_3\sin(\delta_3) + B_{23}V_2V_3\sin(\delta_3 - \delta_2) + P_l.$$

The power network's parameters are set to the following values: $M_1 = 0.6$, $M_2 = 0.8$, $d = 2$, $d_l = 0.5$, $V_1 = 1.02$, $V_2 = 1.05$, $V_3 = 1.02$, $B_{12}, B_{13}, B_{23} = 10$, $P_g = -2.0$, and $P_l = 3.0$.

6. Experiments

This section presents a series of tests conducted to validate the proposed methodology. The goal of our experiments is to show that our gradient-enhanced physics-informed neural network is indeed capable of approximating the analytical solution given in (10) and (11). The neural network models (PINNs and gPINNs) are constructed based on the given initial conditions. In this paper, the neural network approximation results are compared to the true solutions in order to test the models. The experimental design and findings of the equation utilizing the two models will be presented in the following sections.

6.1. Simulation setup and training

A fully-connected four-layer model with a hyperbolic tangent activation function is used for the approximator/surrogate network. The input layer consists of one neuron (the time coordinate of one collocation point), while the hidden layers comprise $[200, 150, 100, 50]$ neurons, respectively, and the output is the linear combination of output neurons. Although the weights and biases of the networks are initialized randomly at the start of the training, both of them start from the same initial condition to ensure a fair comparison. As a reminder, the output of the approximator/surrogate network is the approximate solution to our problem. The residual network is a graph encoding the swing equation and source term and provides the loss function or (5) to drive the approximator/surrogate network's optimization in PINN or gPINN, respectively. The residual points $|\mathcal{N}_f|$ are distributed randomly in the interior of the computational domain (time domain of $t \in [0, 20s]$) at which the swing equation should hold and the initial condition is incorporated with 50 values ($|\mathcal{N}_b|$). In addition to an initial training set, an extensive test data set is also required to evaluate the performance of the neural network. To create the true solutions to the swing equation, the *Scipy.integrate* package in Python is used as an ODE numerical solver. Here, the voltage magnitudes E_g and E_∞ are equal to 1 p.u. and $B_{g\infty} = 0.2$ p.u. The models were implemented and trained using the Pytorch package in Python 3.7 on a personal computer (Apple MacBook Pro with M1 chip), the Adam optimizer [29] with the default learning rate $lr = 10^{-3}$ and a maximum of 20000 epochs in all the cases barring the transfer learning case study. The L^2 relative error of the prediction of u and ω and the L^1 relative error of m_g and d_g is used to evaluate the performance of the estimating models (proposed model over the PINN model). In this study, weights $w_f = w_b = w_i = 1$ are chosen. Different values of w_g, including 1, 0.1, and 0.01, were used during the network training to evaluate the sensitivity of gPINN's performance to this parameter. By assessing the relative L^2 error between the predicted and the exact solution of $\delta(t)$ and $\omega(t)$, the proper value chosen for the w_g is 0.01 which achieved the lowest error.

In the first part of the study, system inertia and damping are assumed to be known, and it is assumed that the system is not in equilibrium. Active power input (P_m) and the initial condition ($[\delta(t_0), \omega(t_0)]^T$) is specific to each case study and will be defined later on. For different values of power input and initial conditions, the system may be stable,

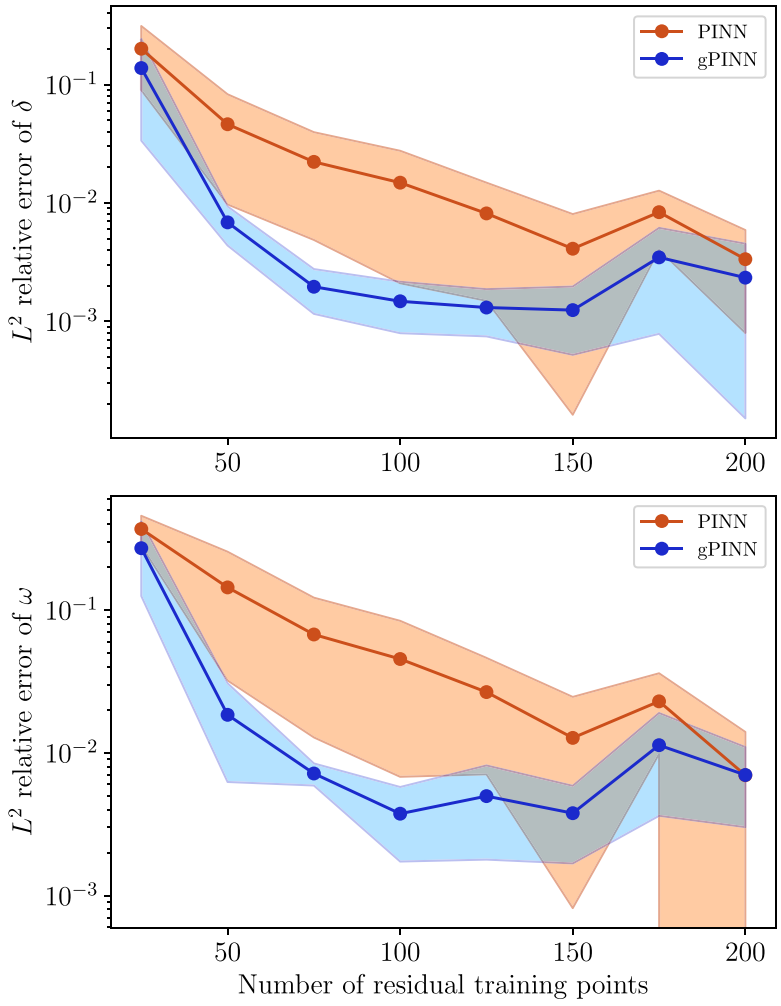

Fig. 4. L^2 relative error of δ and $\frac{\partial \delta}{\partial t} = \omega$ for the PINN and gPINN. The shaded regions represent one standard deviation across 10 random runs.

become unstable, or multiple oscillations may occur. Therefore, it is crucial to achieve high accuracy in each of these regimes. In the second part, inertia and damping are also unknown parameters. Given scattered observed data about angle measurements, our goal is to identify the parameters m_g and d_g of (10), as well as to obtain the trajectory of δ.

6.2. Forward ODE problem — Prediction accuracy in capturing power system dynamics

Our first experiment investigates how the size of the training dataset affects the training convergence of our proposed framework. Fig. 4 depicts the L^2 relative error of prediction $\delta(t)$ and $\omega(t)$ for PINN and gPINN when they are trained with different numbers of residual points. The mean and one standard deviation of 10 independent runs are represented by the line and shaded region. Since increasing the number of residual points in our simulations may lead to over-fitting to the training data, this observation can shed light on choosing the number of appropriate residual points for solving our specific problem that is the swing equation in (10). Note that these results are obtained for the case that the given generator is in the stable mode of operation. It is shown that using up to 150 training points results in smaller L^2 relative error for the prediction of $\delta(t)$ for both gPINN and PINN. Notably, gPINN outperforms PINN with an error approximately one order of magnitude smaller. Furthermore, the gPINN outperforms the PINN in terms of predicting the derivative $\frac{d\delta(t)}{dt} = \omega(t)$. The standard deviation in the gPINN has a tighter bound which represents the robustness of the proposed method in predicting the target variables. Finally, a baseline of 150 residual points was chosen for training the models in the remaining case studies.

Table 1 shows the L^2 relative error ($\times 10^2$) for the prediction of $\delta(t)$ and recovered $\omega(t)$ of 10 independent trials, where the performance of the gPINN algorithm over the PINN algorithm is demonstrated in each considered condition: stable, unstable, or oscillating. Note that for computing the frequency $\omega(t)$, automatic differentiation is employed to directly extract the frequency from the gPINN or PINN. Hence, the more accurate prediction of the $\delta(t)$ will result in a better estimation of the frequency of the system. The table reports the comparison of the best, average, and worst results that are obtained by these two models over the test set. It is observed that minimum, average, and maximum L^2 error for prediction of the $\delta(t)$ offered by the gPINN is lower than corresponding values obtained from the PINN, with the exception of the "stable" case study for the best load angle estimation. An even greater improvement by the gPINN can be seen in the L^2 relative error of $\omega(t)$. Further, the worst L^2 relative error obtained by the gPINN is lower than the average results of the PINN in almost all the cases. As these values indicate, contrary to the gPINN, the PINN struggles with predicting the $\delta(t)$ with high accuracy and subsequently recovering the frequency $\omega(t)$ when the system is facing an oscillating condition. Most of the time it fails to predict the target variables with an acceptable level of confidence for making an informed decision. The calculated standard deviation of the error for the gPINN is considerably less than the PINN

Table 1
Comparison of L^2 relative error ($\times 10^{-2}$) for prediction of δ and recovered ω of two models based on 10 trials.

Model	Test case	δ				ω			
		Min	Average	Max	Standard deviation	Min	Average	Max	Standard deviation
gPINN	Stable	0.054	0.533	1.467	0.506	0.217	0.705	1.278	0.441
	Unstable	0.123	0.481	1.655	0.481	0.147	0.737	2.356	0.708
	Oscillating	0.994	1.794	4.920	1.198	2.448	4.092	7.395	3.049
PINN	Stable	0.048	0.916	3.185	1.027	0.152	2.908	9.931	3.258
	Unstable	0.811	3.758	8.181	2.586	1.517	4.763	10.202	3.204
	Oscillating	3.304	22.946	38.792	16.536	7.653	54.415	94.681	39.243

Table 2
Comparison of L^2 relative error ($\times 10^{-2}$) for prediction of δ and ω of two models based on 10 trials.

Model	Test case	δ				ω			
		Min	Average	Max	Standard deviation	Min	Average	Max	Standard deviation
gPINN	Stable	0.062	0.536	1.288	0.463	0.244	0.620	1.167	0.403
	Unstable	0.120	0.440	1.649	0.501	0.166	0.769	2.375	0.814
	Oscillating	1.178	1.829	4.463	1.306	2.541	4.338	8.178	2.858
PINN	Stable	0.047	1.050	2.724	1.098	0.153	2.355	9.518	2.965
	Unstable	0.962	4.302	7.081	2.161	1.759	4.647	8.230	3.006
	Oscillating	2.848	18.892	42.497	14.311	6.687	63.828	98.526	41.55

(up to 5 and 10 times smaller in the oscillating case for predicting $\delta(t)$ and $\omega(t)$ respectively), which clearly indicates a small variation range for prediction values and robustness of the gPINN. The noted results prove the superiority of the proposed gPINN for solving the general swing equation problem as compared with the base PINN model in different conditions of the systems.

The training of the gPINN and PINN models was conducted over 10 independent trials, with respective average training times of 251.85 ± 27.48 s and 225.82 ± 20.97 s. The slightly increased training time for the gPINN models can be attributed to the additional computational burden involved in computing the gradient of the governing equation. After training, the computational speed of the models in solving the differential equation was evaluated. The solver package took an average of 9.11 ms$\pm$5.31 µs to solve the differential equations (8), while the gPINN model required only 294 µs$\pm$10.9 µs, resulting in a speed-up factor of 31. Importantly, the physics-informed neural network is capable of directly determining δ at any specified time step $\delta(t_1)$, bypassing the need for numerical integration from the boundary conditions at $t = t_0$ to $t = t_1$. This highlights the ability of gPINN/PINN models to accurately and efficiently predict solutions to higher-order differential equations, offering significant advantages over classical numerical integration methods.

To evaluate the performance of the gPINN/PINN models with a dedicated output for ω, we calculated the L^2 relative error ($\times 10^2$) for the prediction of $\delta(t)$ and $\omega(t)$ based on 10 independent trials. The results are presented in Table 2. It is important to note that the swing equation (Eq. (8)) can be rewritten as a system of two equations:

$$\begin{cases} \frac{d}{dt}\delta = \omega\ m_g, \dot{\omega} + d_g, \\ \omega + E_g E_\infty B_{g\infty} \sin(\delta) - P_m = 0. \end{cases}$$

By considering separate outputs for $\delta(t)$ and $\omega(t)$, the gPINN model still outperforms the PINN model in terms of overall performance. However, compared to the previous case, the results do not exhibit a significant improvement. Therefore, for the SMIB system, the neural network architecture with a single output for δ (and subsequently determining ω through numerical differentiation) is preferred due to its advantages in training time and predictive accuracy.

Fig. 5 depicts the swing equation's approximate solution with the actual trajectory of the load angle $\delta(t)$ (in the left side) and the corresponding recovered frequency signal $\omega(t)$ (in the right side) for the gPINN and PINN models on the test set. Fig. 5(a)–(c) shows the predicted load angle and frequency for the three possible states of the generator as mentioned in Table 1. In all these cases, the trajectory starts from the same initial condition $[\delta(t_0), \omega(t_0)]^T = [0.1, 0.1]^T$, and only the input mechanical power is different which puts the system in different condition. Moreover, the depicted approximate solutions have the L^2 relative error close to the average value reported in Table 1 to provide a pictorial overview of the ability of models in estimating the desired variables of interest. In the left figures, it is shown that the trajectory of the $\delta(t)$, with a L^2 relative error of $0.94 \cdot 10^{-2}$ and $0.23 \cdot 10^{-2}$ in stable, $5.89 \cdot 10^{-2}$ and $0.19 \cdot 10^{-2}$ in unstable, $25.35 \cdot 10^{-2}$ and $1.87 \cdot 10^{-2}$ in oscillation state for both PINN and gPINNs, respectively.

It can be found that the gradient-enhanced physics-informed neural network is able to predict the trajectory of the angle $\delta(t)$ with high accuracy and that the frequency signal $\omega(t)$ can be successfully recovered using automatic differentiation in all three examples. Interestingly, in the analysis of the accuracy of the prediction $\delta(t)$ or the recovered $\omega(t)$ in all three conditions mentioned, it is observed that the PINN tends to underestimate the peaks and troughs, particularly in the unstable or oscillating state (see Fig. 5(b) and (c)). Moreover, these predictions tend to worsen as the predictions are made further into the future. Overall, gPINN model outperforms the PINN in terms of both predicting $\delta(t)$ and obtaining corresponding $\omega(t)$, since gPINN utilizes the information of the gradient and thus has a much faster convergence rate than PINN. In other words, the gPINN also has computational benefits versus the PINN. Therefore, using only a handful of initial data, the gradient-enhanced physics-informed neural network can accurately capture the intricate nonlinear behavior of the swing equation.

6.3. The importance of transfer learning

Subsequently, the significance of employing transfer learning became evident as it accelerated training and enhanced the performance of our proposed model, even with a reduced number of epochs. The transfer learning method entails using pre-trained models as the starting point on specific tasks (e.g. solving a swing equation) given the vast computational and time resources required to develop neural network models on these problems and from the huge jumps in a skill that they provide on related problems. The initial network weights and biases, obtained from the fully trained network for predicting $\delta(t)$, can be utilized to initialize the gPINN/PINN network that is to be solved. In this way, the first gPINN/PINN passes on the encoding knowledge it has gained to the second gPINN/PINN.

To showcase the advantage of transfer learning in gPINN/PINN, a test case with a stable state and a source term is solved. The networks are initialized with the outcomes achieved during the training with Eq. (10) serving as the source term. In the absence of transfer learning,

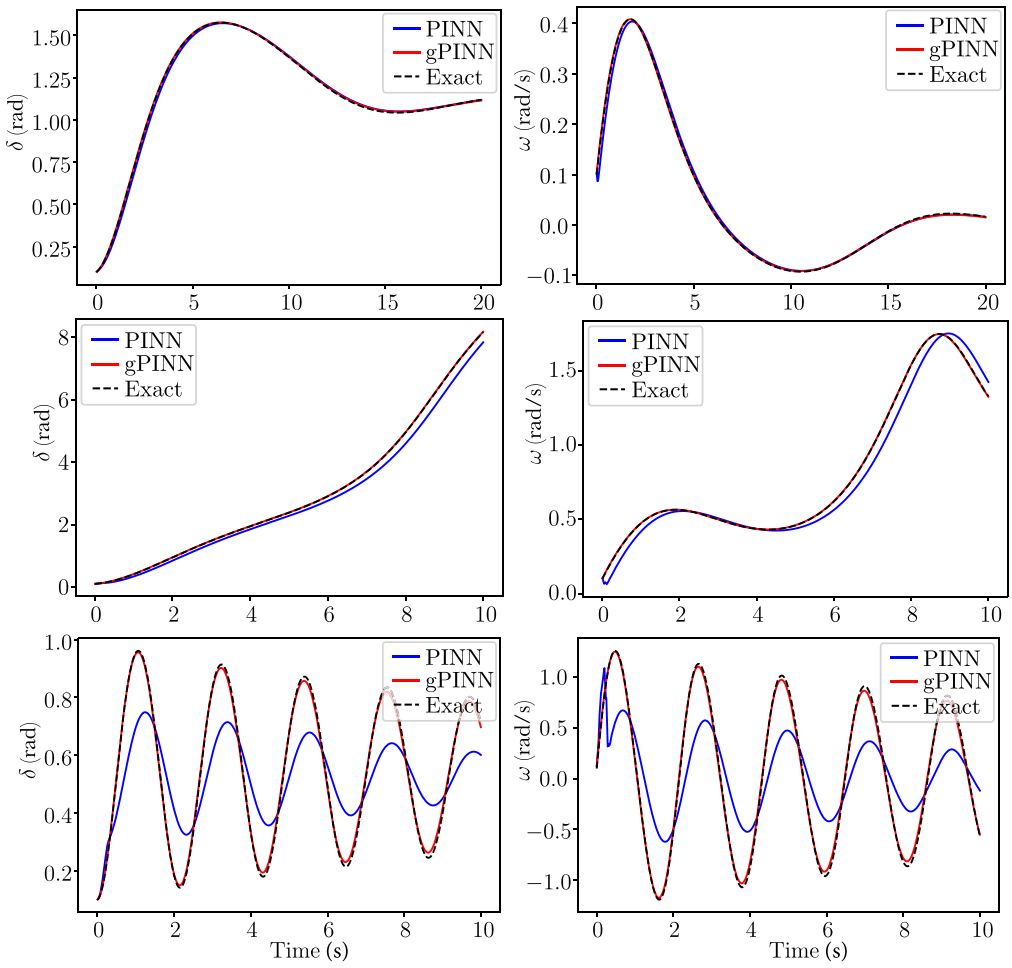

Fig. 5. Example of the predicted $\delta(t)$ (left) and corresponding $\omega(t)$ (right) in three cases. As expected, the prediction error increases as the distance into the future increases.

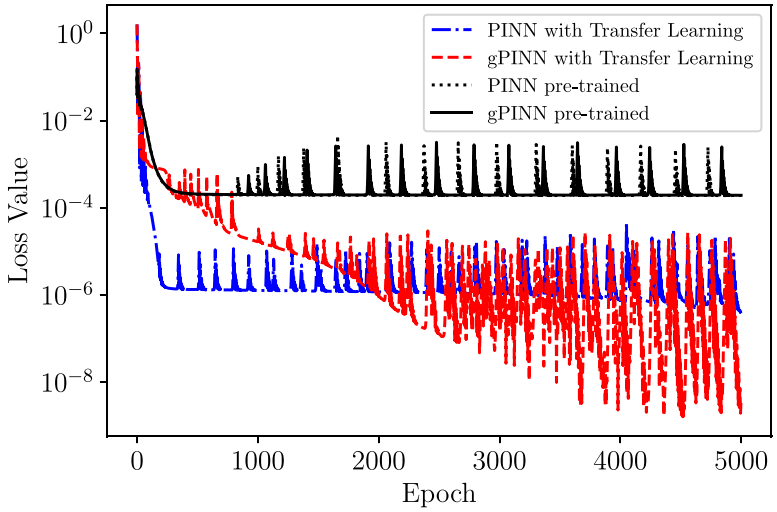

Fig. 6. Training error with transfer learning for the gPINN/PINN models and their pre-trained test cases.

the gPINN/PINN models were trained using the Adam optimizer for 20,000 epochs to achieve comparable performance to state-of-the-art studies in this field. But when using a transfer learning technique each model is only trained over 5000 epochs which therefore indicates quicker training. Fig. 6 depicts a comparison of the training error for the gPINN and PINN network initialized with their corresponding pre-trained models (in their first 5000 epochs), e.g., without transfer learning and with transfer learning. In this case, transfer learning usage allows gaining at least two orders of magnitude improvement in the training error in less than 1000 epochs. Further, it is found that utilization of transfer learning leads to an initial super-convergence to a relatively low training error. Applying this technique significantly reduces the average training time of gPINN/PINN to 64.98 ± 7.16 s and 62.43 ± 5.46 s, respectively, resulting in a notable speed-up of 3.8

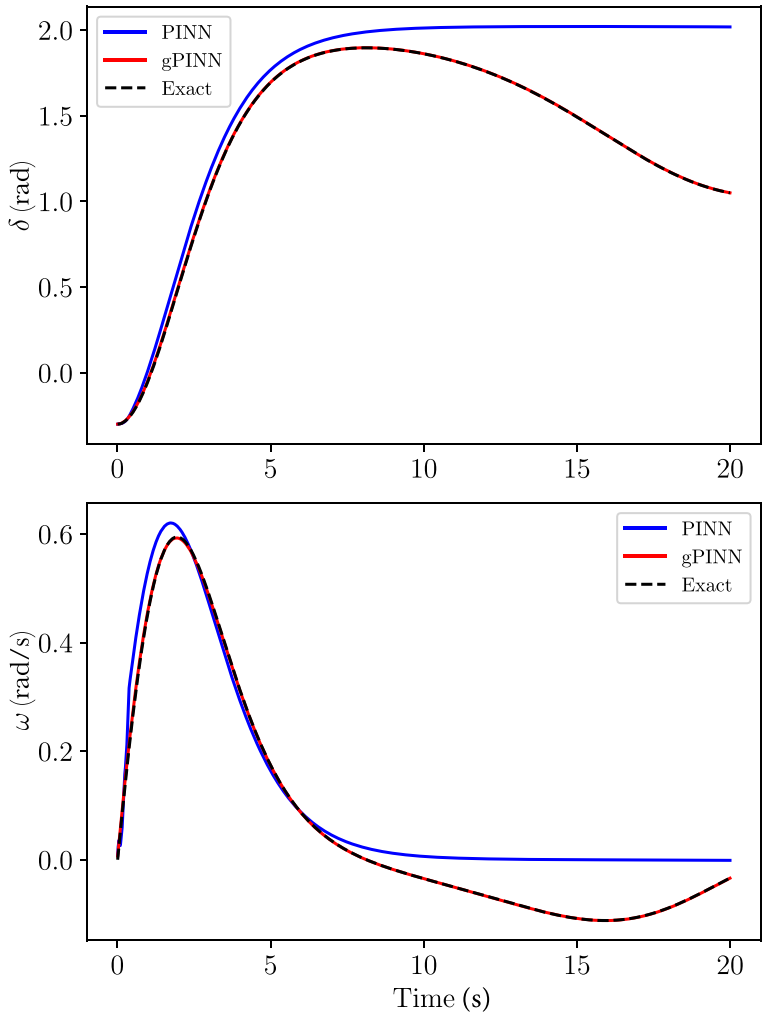

Fig. 7. The predicted δ and ω from PINN and gPINN after 5000 training epochs using transfer learning.

times compared to the original training process. It is important to note that the evaluation of a trained neural network remains unchanged, ensuring consistent performance during inference. As a result, transfer learning is a crucial operation for gPINN/PINN to compete with other scientific computing solvers.

The performance and convergence behavior of the proposed model suggests that the transfer learning method is a good match for the network learning this problem. Fig. 7 illustrates actual trajectories of the $\delta(t)$ and the recovered frequency signal $\omega(t)$, and the approximations learned by the PINN and gPINN on the test set. It is shown how much each model has the ability to tacking the true solution after just 5000 epochs. In this case, the relative L^2 between the exact and predicted solutions in $\delta(t)$ are $29.3 \cdot 10^{-2}$ and $0.17 \cdot 10^{-2}$ for PINN and gPINN models, respectively. These values for the frequency signal $\omega(t)$ are $27.7 \cdot 10^{-2}$ and $0.75 \cdot 10^{-2}$. This plot shows that the gPINN has an easy prediction task in finding the true solution compare to the PINN network, indicating that the proposed model may effectively capture such behavior, as indeed confirmed in the corresponding low gPINN prediction errors.

6.4. Inverse problem of ODE — Discovery of inertia and damping coefficients

Here, the performance in predicting system inertia and damping from observed trajectories is evaluated. Given scattered and potentially noisy data on the $\delta(t)$ component, our goal is to identify unknown parameters m_g and d_g, as well as to obtain a qualitatively accurate reconstruction of $\delta(t)$. For inferring m_g and d_g, the data measurements of the load angle $\delta(t)$ have been selected in only 5 time steps. These time steps are randomly chosen within the simulation horizon, demonstrating the capability of our method to learn from sparse and scattered training data The neural network architectures used here are the same as the cases for the forward problem. A visual comparison against the exact load angle and frequency solutions is presented in Fig. 8. Similar to the observation made in the forward swing equation problem, the gPINN exhibits superior performance compared to the PINN in this case. While the PINN failed to predict $\delta(t)$ and $\omega(t)$ near the peaks and end of the simulation, the gPINN has decent accuracy. Moreover, it is shown that during training, the predicted m_g and d_g in PINN did not converge to the true value (in the case of predicting d_g, the PINN performs better), while with the gPINN, the predicted m_g and d_g of a system is more accurate. Note that the predicted parameters will have an effect on the relative error of the predicted variables $\delta(t)$ and the recovered frequency signal $\omega(t)$. For a better illustration and to test the robustness of the models, Table 3 compares the performance of the gPINN and PINN algorithms in terms of the minimum, average, maximum, and standard deviation of L^1 relative error of the predicted parameters m_g and d_g based on 10 independent trials for the stable state of the system. It is found that the best L^1 relative error in predicting both parameters is offered by the gPINN algorithm ($0.000 \cdot 10^{-2}$ and $0.008 \cdot 10^{-2}$ for m_g and d_g, respectively). It is observed that the minimum, average, and maximum L^1 relative error offered by the gPINN algorithm is lower than the corresponding values obtained from

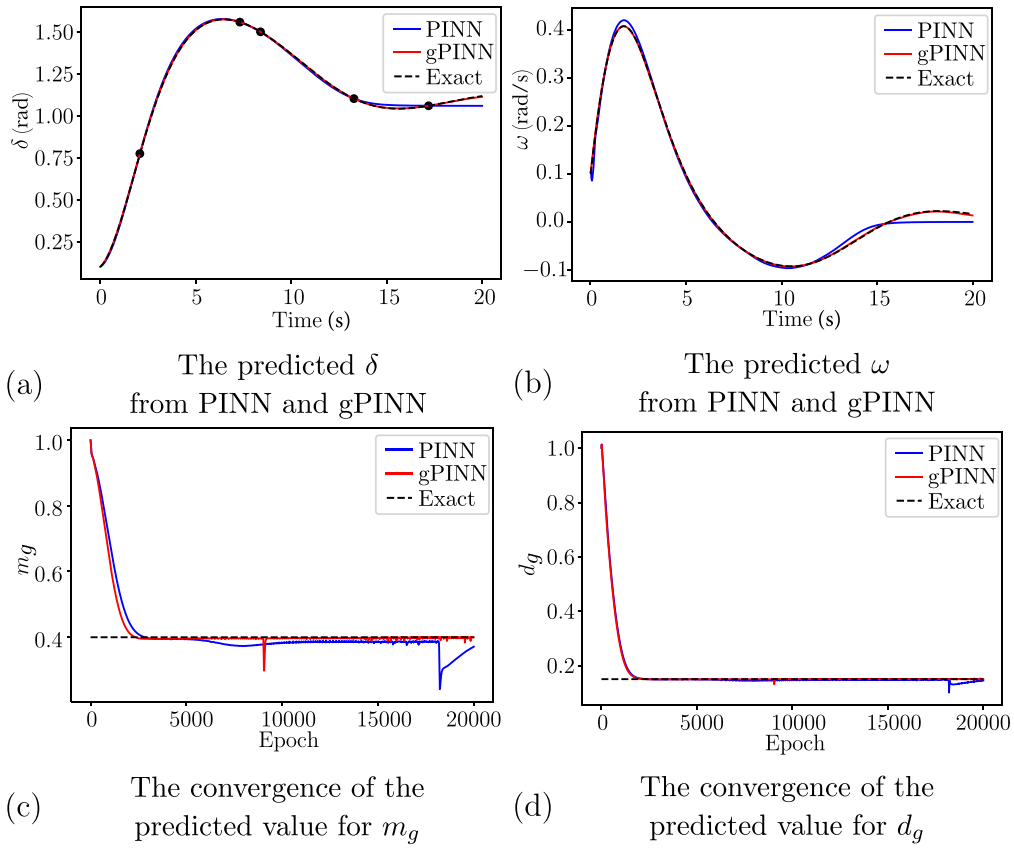

Fig. 8. Inferring both m_g and d_g throughout training. The black dots in (a) show the observed locations of δ.

Table 3
Comparison of L^1 relative error ($\times 10^{-2}$) for prediction of moment of inertia m_g and damping coefficient d_g of two models based on 10 trials.

Model	Parameter	Min	Average	Max	Standard deviation
gPINN	m_g	0.000	0.181	0.615	0.222
	d_g	0.008	0.195	0.456	0.167
PINN	m_g	0.061	1.432	7.150	2.124
	d_g	0.013	0.575	3.374	1.011

the PINN algorithm. This finding of inferring a continuous quantity of interest from auxiliary measurements by exploiting the underlying physics demonstrates promise in handling high-dimensional inverse problems.

6.5. Three-bus power network

The improved fully-connected architecture proposed in [30] was used to implement the neural network that estimates the state variable trajectories (i.e., dynamic equations). This architecture has a forward pass described as follows:

$$U = \phi(XW^1 + b^1), \quad V = \phi(XW^2 + b^2)$$
$$H^{(1)} = \phi(XW^{z,1} + b^{z,1})$$
$$Z^{(k)} = \phi(H^{(k)}W^{z,k} + b^{z,k}), k = 1, \ldots, d$$
$$H^{(k+1)} = (1 - Z^{(k)}) \odot U + Z^{(k)} \odot V, k = 1, \ldots, d$$
$$f_\theta(x) = H^{(d+1)}W + b,$$

In this context, the input tensor to the neural network is represented by X which is time, while d denotes the number of hidden layers in the network. The symbol $\odot$ represents the Hadamard or element-wise product, and ϕ is a point-wise hyperbolic tangent activation function used in this paper. The set of trainable parameters in this new network architecture is fundamentally identical to those in a typical fully-connected architecture with the addition of the weights and biases used by the two transformer networks. These parameters are initialized using the Glorot normal algorithm and can be represented by the following collection:

$$\theta = \{W^1, b^1, W^2, b^2, \{W^{z,l}, b^{z,l}\}_{l=1}^d, W, b\}$$

The results of our experiments indicate that this new architecture performs better than the traditional fully connected architecture. This is due to its ability to consider the multiplicative interactions between various inputs and improve the hidden-state representation using residual connections, as proposed in [30].

Fig. 9 compares the PINN and gPINN models for a simulated dynamical equations trajectory for the representative initial condition selected from the test dataset. There is excellent agreement observed between the simulated trajectory generated by the gPINN model and the true trajectory obtained through the integration of Eq. (11). The results clearly illustrate that our gPINN method significantly outperforms the other PINN model. In order to gain a better understanding of the long-time simulation accuracy of the gPINN/PINN framework, 10 initial conditions are randomly sampled from the test dataset. The frameworks are then trained on these conditions, and the average and standard deviation of the L^2 relative error for each state variable are computed. The L^2 relative errors for each state variable are shown in Table 4. Based on the results presented, it can be concluded that gPINN is capable of accurately simulating dynamical equations for a sample power network over lengthy time horizons. In this case, the average training time for gPINN/PINN is reported as 314.54 ± 27.10 s and 250.35 ± 20.95 s, respectively. The slight increase in training time compared to the SMIB case is attributed to the complexity of the models. In terms of

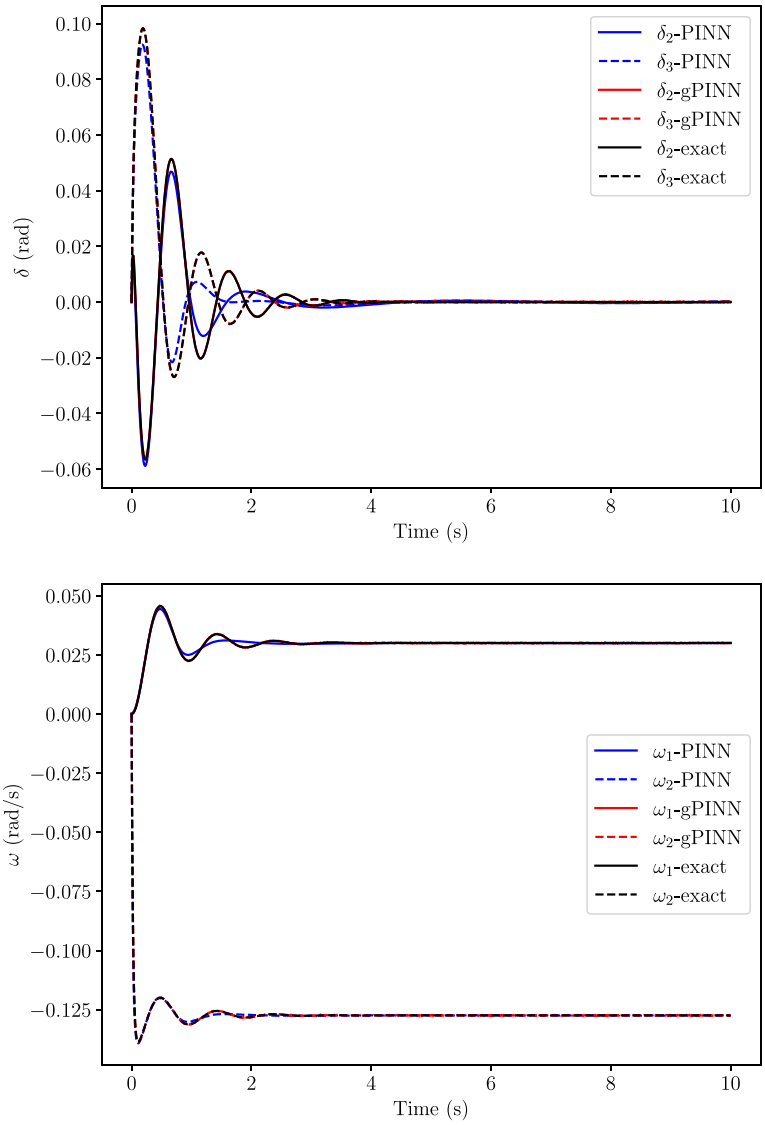

Fig. 9. Example of predicted variables for Eq. (11).

Table 4
Comparison of L^2 relative error ($\times 10^{-2}$) for prediction of variables of two models based on 10 trials.

Model		ω_1	ω_2	δ_2	δ_3
gPINN	Average	1.441	1.131	0.428	0.776
	St. Dev.	0.858	0.920	0.204	0.025
PINN	Average	3.932	3.174	1.373	1.111
	St. Dev.	1.339	1.412	0.512	0.057

inference, the solver package takes an average of 58.21 ms ± 6.37 μs to solve Eq. (11), while the gPINN only requires 325.93 μs ± 9.81 μs. As expected, for larger systems, the computational speed-up becomes even more significant, as solving large-scale differential equations can be computationally expensive. Meanwhile, the evaluation of a trained neural network remains computationally efficient even for large network sizes.

7. Discussion and conclusions

This paper presents a framework based on the gradient-enhanced physics-informed neural networks (gPINNs) for power system applications specifically in dynamics studies. The effectiveness of gPINN in determining the rotor angle and frequency for a single-machine infinite-bus system is demonstrated in the forward problem case. Our numerical findings from all of the cases show that the considered gPINN outperforms the considered PINN with the same number of training points. By incorporating the underlying swing equation, gradient-enhanced physics-informed neural networks achieve high accuracy (up to 13.30× improvement on the average L^2 of relative error compared to the PINN model) while requiring significantly less training data. Our approach significantly reduces the computational time required to solve differential–algebraic systems of equations in power systems. It achieves a speed-up ranging from 31 to 171 times faster compared to conventional numerical methods, showcasing its superior computational efficiency. Additionally, the application of transfer learning is shown to effectively optimize the model with a reduced number of training epochs (the training time is reduced to just over 1 min).

Moreover, one of the key advantages of gPINNs is that they can solve inverse PDE/ODE issues as readily as forward problems in terms of implementation. The results have demonstrated successful identification of uncertain system parameters, such as inertia and damping, with an L^1 relative error of approximately $0.181 - 0.195 \times 10^{-2}$ using a very limited set of input data. Our findings suggest that these methods have the potential to be successfully applied in larger systems, opening up a slew of new possibilities for power system security and optimization

while maintaining high computational speed and accuracy. As a future extension of this work, the focus will shift toward exploring physics-aware learning models for Optimal Power Flow (OPF). This approach is crucial to ensure that the optimal solutions obtained from machine learning models adhere to practical dynamical constraints.

CRediT authorship contribution statement

Mostafa Mohammadian: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing – original draft, Resources, Visualization. **Kyri Baker:** Conceptualization, Methodology, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Ferdinando Fioretto:** Conceptualization, Methodology, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

The research in this paper was supported by National Science Foundation awards 2041835, 2143706, 2242930, and 2242931. This work utilized the Alpine high performance computing resource at the University of Colorado Boulder. Alpine is jointly funded by the University of Colorado Boulder, the University of Colorado Anschutz, and Colorado State University.

References

[1] E. Zhao, Y. Han, X. Lin, E. Liu, P. Yang, A.S. Zalhaf, Harmonic characteristics and control strategies of grid-connected photovoltaic inverters under weak grid conditions, Int. J. Electr. Power Energy Syst. 142 (2022) 108280, http://dx.doi.org/10.1016/j.ijepes.2022.108280.

[2] E. Zhao, Y. Han, X. Lin, P. Yang, F. Blaabjerg, A.S. Zalhaf, Impedance characteristics investigation and oscillation stability analysis for two-stage PV inverter under weak grid condition, Electr. Power Syst. Res. 209 (2022) 108053, http://dx.doi.org/10.1016/j.epsr.2022.108053.

[3] M.H. Dinh, F. Fioretto, M. Mohammadian, K. Baker, Towards understanding the unreasonable effectiveness of learning AC-OPF solutions, 2021, arXiv:2111.11168.

[4] A. Zamzam, K. Baker, Learning optimal solutions for extremely fast AC optimal power flow, in: IEEE SmartGridComm, 2020.

[5] F. Fioretto, T.W. Mak, P. Van Hentenryck, Predicting AC optimal power flows: Combining deep learning and Lagrangian dual methods, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34, No. 01, 2020, pp. 630–637, http://dx.doi.org/10.1609/aaai.v34i01.5403.

[6] M. Mohammadian, F. Aminifar, N. Amjady, M. Shahidehpour, Data-driven classifier for extreme outage prediction based on Bayes decision theory, IEEE Trans. Power Syst. 36 (6) (2021) 4906–4914, http://dx.doi.org/10.1109/TPWRS.2021.3086031.

[7] D. Biagioni, P. Graf, X. Zhang, A.S. Zamzam, K. Baker, J. King, Learning-accelerated ADMM for distributed DC optimal power flow, IEEE Control Syst. Lett. 6 (2022) 1–6, http://dx.doi.org/10.1109/LCSYS.2020.3044839.

[8] M. Mohammadian, K. Baker, M.H. Dinh, F. Fioretto, Learning solutions for intertemporal power systems optimization with recurrent neural networks, in: 2022 17th International Conference on Probabilistic Methods Applied to Power Systems, PMAPS, 2022, pp. 1–6, http://dx.doi.org/10.1109/PMAPS53380.2022.9810638.

[9] Y. Han, Y. Feng, P. Yang, L. Xu, A.S. Zalhaf, An efficient algorithm for atomic decomposition of power quality disturbance signals using convolutional neural network, Electr. Power Syst. Res. 206 (2022) 107790, http://dx.doi.org/10.1016/j.epsr.2022.107790.

[10] P. Kundur, Power System Stability and Control, McGraw-Hill, 1994.

[11] A. Iserles, A First Course in the Numerical Analysis of Differential Equations, second ed., in: Cambridge Texts in Applied Mathematics, Cambridge University Press, 2008, http://dx.doi.org/10.1017/CBO9780511995569.

[12] B. Stott, Power system dynamic response calculations, Proc. IEEE 67 (2) (1979) 219–241, http://dx.doi.org/10.1109/PROC.1979.11233.

[13] R. Schainker, P. Miller, W. Dubbelday, P. Hirsch, G. Zhang, Real-time dynamic security assessment: fast simulation and modeling applied to emergency outage security of the electric grid, IEEE Power Energy Mag. 4 (2) (2006) 51–58, http://dx.doi.org/10.1109/MPAE.2006.1597996.

[14] G.S. Misyris, A. Venzke, S. Chatzivasileiadis, Physics-informed neural networks for power systems, in: IEEE Power and Energy Society General Meeting, 2020, pp. 1–5, http://dx.doi.org/10.48550/arxiv.1911.03737.

[15] R. Temam, A. Chorin, Navier Stokes equations: Theory and numerical analysis, J. Appl. Mech. 45 (2) (1984) 456, http://dx.doi.org/10.1115/1.3424338.

[16] A. Taflove, S. Hagness, Computational Electrodynamics: The Finite-Difference Time-Domain Method, 3rd ed., in: The Electrical Engineering Handbook, vol. 2062, 2005, http://dx.doi.org/10.1016/B978-012170960-0/50046-3.

[17] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.

[18] L. Lu, R. Pestourie, W. Yao, Z. Wang, F. Verdugo, S.G. Johnson, Physics-informed neural networks with hard constraints for inverse design, 2021, arXiv:2102.04626.

[19] A. Yazdani, L. Lu, M. Raissi, G.E. Karniadakis, Systems biology informed deep learning for inferring parameters and hidden dynamics, PLoS Comput. Biol. 16 (11) (2020) 1–19, http://dx.doi.org/10.1371/journal.pcbi.1007575.

[20] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. 18 (153) (2018) 1–43, URL: http://jmlr.org/papers/v18/17-468.html.

[21] M. Abadi, et al., TensorFlow: Large-scale machine learning on heterogeneous distributed systems, 2016, URL: http://arxiv.org/abs/1603.04467.

[22] A. Paszke, et al., Automatic differentiation in PyTorch, in: Proceedings of the NIPS 2017 Autodiff Workshop, 2017.

[23] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, Q. Liao, Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review, Int. J. Autom. Comput. 14 (2017) 1–17, http://dx.doi.org/10.1007/s11633-017-1054-2.

[24] C. Moya, G. Lin, DAE-PINN: A physics-informed neural network model for simulating differential-algebraic equations with application to power networks, 2021, arXiv:2109.04304.

[25] J. Yu, L. Lu, X. Meng, G.E. Karniadakis, Gradient-enhanced physics-informed neural networks for forward and inverse PDE problems, Comput. Methods Appl. Mech. Engrg. 393 (2022) 114823, http://dx.doi.org/10.1016/j.cma.2022.114823.

[26] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis, DeepXDE: A deep learning library for solving differential equations, SIAM Rev. 63 (1) (2021) 208–228, http://dx.doi.org/10.1137/19M1274067.

[27] T.L. Vu, K. Turitsyn, A framework for robust assessment of power grid stability and resiliency, IEEE Trans. Automat. Control 62 (3) (2017) 1165–1177, http://dx.doi.org/10.1109/tac.2016.2579743.

[28] D. Zografos, M. Ghandhari, Power system inertia estimation by approaching load power change after a disturbance, in: 2017 IEEE Power Energy Society General Meeting, 2017, pp. 1–5, http://dx.doi.org/10.1109/PESGM.2017.8273824.

[29] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May, 2015, [Online] Available at: https://arxiv.org/pdf/1907.02206.

[30] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient flow pathologies in physics-informed neural networks, SIAM J. Sci. Comput. 43 (5) (2021) A3055–A3081, http://dx.doi.org/10.1137/20M1318043.