Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 390 (2022) 114502
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems

Han Gao, Matthew J. Zahr, Jian-Xun Wang*

Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, IN, United States of America

Received 16 July 2021; received in revised form 3 November 2021; accepted 18 December 2021
Available online 11 January 2022

Abstract

Despite the great promise of the physics-informed neural networks (PINNs) in solving forward and inverse problems, several technical challenges are present as roadblocks for more complex and realistic applications. First, most existing PINNs are based on point-wise formulation with fully-connected networks to learn continuous functions, which suffer from poor scalability and hard boundary enforcement. Second, the infinite search space over-complicates the non-convex optimization for network training. Third, although the convolutional neural network (CNN)-based discrete learning can significantly improve training efficiency, CNNs struggle to handle irregular geometries with unstructured meshes. To properly address these challenges, we present a novel discrete PINN framework based on graph convolutional network (GCN) and variational structure of PDE to solve forward and inverse partial differential equations (PDEs) in a unified manner. The use of a piecewise polynomial basis can reduce the dimension of search space and facilitate training and convergence. Without the need of tuning penalty parameters in classic PINNs, the proposed method can strictly impose boundary conditions and assimilate sparse data in both forward and inverse settings. The flexibility of GCNs is leveraged for irregular geometries with unstructured meshes. The effectiveness and merit of the proposed method are demonstrated over a variety of forward and inverse computational mechanics problems governed by both linear and nonlinear PDEs.

Keywords: Partial differential equations; Inverse problem; Physics-informed machine learning; Graph convolutional neural networks; Mechanics

1. Introduction

Partial differential equations (PDEs) play an important role in engineering applications since most of the physics governing natural or man-made complex systems are described by PDEs. However, finding solutions to most PDEs is a challenging problem, which may involve sophisticated numerical techniques and can be time-consuming, particularly for scenarios where parameters or initial/boundary conditions are partially known. Most recently, physics-informed neural networks (PINNs) [1], as a new paradigm for solving both forward and inverse PDEs, have attracted increasing attention due to its great flexibility and simplicity compared to classic numerical methods. The general idea of PINNs is to approximate the PDE solutions with deep neural networks, whose loss

* Corresponding author.
E-mail address: jwang33@nd.edu (J.-X. Wang).

https://doi.org/10.1016/j.cma.2021.114502
0045-7825/

functions are formulated as a combination of PDE residuals and data mismatch. This unique loss formulation enables physics-informed training that leverages the information from both physics equations and sparse observation data.

Based on how to construct differential operators of PDE residuals using neural networks, PINNs can be classified into two categories: continuous and discrete. The continuous PINNs usually employ fully-connected (FC) neural networks to approximate the continuous solution function $f(\mathbf{x}, t)$ with respect to spatiotemporal coordinates $(\mathbf{x}, t)$ in a point-wise manner, where the spatial and temporal derivative terms are computed using automatic differentiation (AD) techniques [2]. The continuous PINNs are undergoing a renaissance since recent impressive contributions made by Raissi et al. [1] on development of the continuous FC-PINN for solving forward and inverse PDEs. Its merit and effectiveness has been demonstrated over a plethora of scientific applications in many areas [3–7]. For instance, in fluid applications, PINNs have been used for fast surrogate modeling of idealized vascular flow problems in a forward parametric setting without training labels [8]. Moreover, PINNs have also been formulated in an inverse modeling setting to extract unobservable information (e.g., blood flow velocity) from observable data (e.g., concentration data) in cardiovascular problems [9–12]. Jin et al. [13] applied FC-PINNs to solve Navier–Stokes equations, ranging from laminar to turbulent regimes, while Mao et al. [14] further showed their effectiveness on high-speed flow problems. Recently, NVIDIA developed a scalable implementation SimNet based on continuous PINNs and applied it to solve various multiphysics problems with massive GPU parallelization [15].

Despite the enormous success and rapid developments thanks to their great flexibility, the current continuous PINNs still have some limitations. First, they suffer from high training cost since the point-wise formulation requires huge amount of AD computations on vast collocation points in a high-dimensional spatiotemporal (and parameter) domain [16,17]. Second, it is challenging to formulate a strict enforcement of initial/boundary conditions (IC/BCs) for continuous PINNs, which has been demonstrated to be effective in finding correct unique PDE solutions, especially when labeled data is very scarce or absent [8]. Although a distance-based particular solution can be introduced to strictly impose IC/BCs on a few simple 2-D domains using either specifically designed algebraic expressions or low-capacity neural networks [8,18], it fails to show the effectiveness on complex geometries for real-world applications. To reduce training costs and enable efficient learning, discrete PINNs that leverage convolution operations and numerical discretizations have begun to spur interests due to their better efficiency and scalability [19,20]. Specifically, convolutional neural networks (CNN) are often used in discrete PINN to directly learn the entire spatiotemporal solution fields end to end and all the derivative terms of the physics-informed loss are calculated based on numerical discretization instead of point-wise AD. For instance, Zhu et al. [19] developed a physics-constrained convolutional an encoder–decoder to solve high-dimensional elliptic PDEs, and Geneva et al. [21] further extended this framework to dynamic hyperbolic PDEs with parametric initial conditions. Zhang et al. [22] presented a physics-guided CNN for seismic response modeling and also explored the similar idea with a Recurrent Neural Network (RNN) for metamodeling of nonlinear structures [23]. Wandel et al. [24] recently proposed a data-free fluid surrogate based on an autoregressive U-net in a parametric setting. In aforementioned works, the computational domains are regular and discretized by uniform grids, where PDE residuals are calculated by finite difference (FD) methods. This is because the FD-based CNNs are fundamentally rooted in structured Cartesian grids of rectangular domains. Besides FD-based PINN, finite volume (FV) discretization has also been utilized to construct the PDE-based loss function to solve steady fluid problems, which, however, is still restricted to rectangular domains due to intrinsic limitations of classic convolution operations [25]. To enable physics-informed CNNs to solve parametric PDEs on irregular domains with unstructured grids, Gao et al. [20] proposed a geometry-adaptive physics-informed CNN, PhyGeoNet, which embeds a pre-computed coordinate mapping into the classic CNN structure. Although the effectiveness of the PhyGeoNet has been demonstrated on simple irregular domains, it remains challenging for general complex geometries at large.

Motivated by existing challenges, we propose a novel discrete PINN framework to handle irregular domains with unstructured grids based on generalized convolution operations. Namely, the convolution operations are directly performed on unstructured mesh data, which can be seen as discrete non-euclidean manifolds, i.e., Graph. Moreover, the construction of PDE-informed graph convolutional network (GCN) structure is inspired by finite element (FE) method [26,27], which is another classic numerical discretization technique that possesses many advantages for physics-informed learning. First, thanks to a variational formulation (weak form) of PDE residuals, where Neumann boundary conditions can be naturally incorporated in the weak formulation of governing equations, the order of differential operators can be effectively reduced by integration by part and thus the learning complexity can be largely mitigated. Moreover, the massive amount of collocations points required by strong-form PINNs can be

对这里一直在提卷积，但似乎有无卷积对建立映射没有什么关系

① PINN 是对求解域整理点集，利用自动微分去使得 ANN 这个初始化的解函数逼近真实解

FC-PINN

② Discrete-PINN 并没有利用自动微分，而是结果使统数值方法对方程做离散，从而求得微分算子

对方程做离散.

其实计算机求 PDE 核心是微分算子怎么表示。传统计算机主要基于数值微分。在神经网络计算里，我们可以用自动微分精确计算某点导数.

这里列举了一些 weak form of PDEs 的优势：

① 微分算子的阶数会下降，从而可以极大降低神经网络学习的复杂性
并且 Neumann 边界条件会自动在残差中满足

② 之前用 strong form of PDES 我们要在 collection of points 上去逼近，现在我们只需在少量离散点上去逼近我们的解。训练难度会下降

③

如果是用 point-wise 形式的学习，我们很难施加 hard bc。那之前那篇 CMAME 文章怎么施加的？

replaced by a relatively small amount of quadrature points, which could potentially reduce considerable training cost. The variational (weak) formulation has been recently developed for continuous PINNs and notable superiority has been shown over strong-form PINNs [28–33]. In these variational continuous PINNs, a point-wise fully-connected neural network is usually built as the trial basis, combined with polynomial test functions, to formulate the variational forms in Petrov–Galerkin fashion. Due to the black-box nature of the deep neural networks, accurate quadrature rules are difficult to construction, which leads to additional error associated with variational crimes. Moreover, the essential BCs cannot be imposed in a hard manner due to the point-wise formulation. The FEM-Net proposed by Yao et al. [34] is a FE-based discrete PINN, where a FE-based convolution has been developed to build variational PDE residuals for CNNs. However, this method is under a linear assumption and still limited to rectangular domains due to the classic CNN backbone.

In this work, we proposed an innovative discrete PINN framework based on graph convolutional network and variational structure of PDE to solve forward and inverse PDEs in a unified manner. Specifically, the novel contributions are summarized as follows:

(a) We introduce the graph convolution operation into physics-informed learning to fully leverage the power of FE-based discretization for irregular domains with unstructured meshes. Unlike the state-of-art discrete PINNs based on classic CNNs, the proposed approach does not need rasterization as it can directly handle unstructured mesh with simplex/quadrilateral elements as traditional FE solver does.
(b) A set of finite-dimensional polynomial basis functions are used to reconstruct the full-field predictions based on the output nodal solution graph in a Galerkin formulation, and thus, the search space can be significantly reduced to facilitate training. Moreover, since both test/trial functions are based on standard polynomials, the variational integrals can be computed accurately using Gaussian quadrature.
(c) The proposed PINN is designed to exactly satisfy essential boundary conditions, avoiding penalty coefficient tuning in most PINNs with a soft BC enforcement.
(d) A new data assimilation scheme is proposed to strictly enforce observation data.

2. Methodology

2.1. Overview

Consider a physical system in a bounded domain ($\Omega \subset \mathbb{R}^d$) governed by a set of nonlinear, steady parameterized PDEs in the generic discretized form,

$$\boldsymbol{R}(\boldsymbol{U}(\boldsymbol{\mu}); \boldsymbol{\mu}) = 0, \tag{1}$$

where $\boldsymbol{\mu} \in \mathbb{R}^{N_{\mu}}$ is the PDE parameter vector, $\boldsymbol{U} : \mathbb{R}^{N_{\mu}} \to \mathbb{R}^{N_U}$ is the discrete parameter-dependent state vector implicitly defined as the solution of (1), and $\boldsymbol{R} : \mathbb{R}^{N_U} \times \mathbb{R}^{N_{\mu}} \to \mathbb{R}^{N_U}$ represents the discretized PDE operator. The set of PDEs are subjected to boundary conditions (BCs), which are defined on the boundary $\partial\Omega$ of the domain. In this work, we present an innovative physics-informed graph neural Galerkin network (PI-GGN) to establish a solution approach for such PDE-governed system in both forward and inverse settings. In the forward problem, we aim to obtain the solution $\boldsymbol{U}$ given known BCs and parameters $\boldsymbol{\mu}$; as for the inverse setting, the system is solved when BCs and parameters $\boldsymbol{\mu}$ are partially known, whereas sparse observations of the state are available. In the proposed framework, a GCN is devised to learn nodal solutions of the state on a set of unstructured grids. The PDE residuals in the physics-informed loss function are reconstructed based on the continuous Galerkin method. Essential BCs of the system are imposed in a hard manner and additional data can be assimilated to solve the forward and inverse problems simultaneously. Each component of the proposed method will be detailed in the following subsections.

2.2. Graph convolutional neural network for unstructured data

There has been growing interest in applying GCN for scientific machine learning problems because of its great flexibility in dealing with unstructured data. Excellent performance of the graph-based learning has been reported in modeling various computational mechanics problems through classic data-driven training [35–40]. In general, by defining convolution operations for non-Euclidean space, GCNs generalize CNN-type constructions to graph data. The capability of modeling dependencies between nodes of a graph is the key that enables GCNs to handle

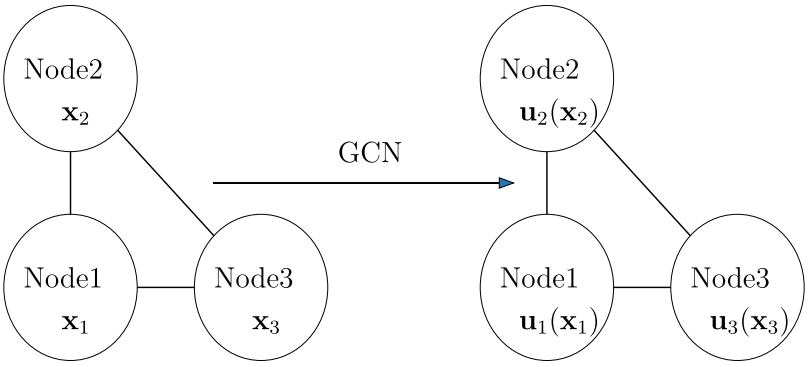


Fig. 1. An example of a GCN, where the input/output graph has 3 nodes & edges and the same adjacency matrix ($\mathcal{N}(1) = \{2, 3\}$, $\mathcal{N}(2) = \{1, 3\}$, $\mathcal{N}(3) = \{1, 2\}$). The input feature is the coordinate of each node ($\boldsymbol{f}_i^{(\text{in})} = x_i$), while the output feature is the nodal solution vector ($\boldsymbol{f}_i^{(\text{out})} = u_i(x_i)$).

unstructured mesh data with any arbitrary boundaries. As shown in Fig. 1, a graph consists of nodes and edges, where each node is defined by its feature vector $\boldsymbol{f}$ and the relation with other nodes is described by edges. The neighbor $\mathcal{N}(\cdot)$ of a node refers to a set of adjacent nodes that are connected to that node via edges. Therefore, a mesh with unstructured grids and corresponding nodal PDE solutions can be naturally described as graphs. Similar to CNN-based discrete PINN [20], a GCN is built to model the discretized solution fields $\boldsymbol{U}(\bar{\boldsymbol{\mu}}) \approx \hat{\boldsymbol{U}}(\boldsymbol{\Theta}^*)$, where $\boldsymbol{\Theta}^*$ are trained parameters of the GCN for graph convolutions for the parameter $\bar{\boldsymbol{\mu}}$.

Remark. In general, the input feature vector of GCN can be any spatially varying field discretized by the mesh due to the universal approximation capacity of deep neural network. In this work, the GCN takes an input graph that each node is associated with its spatial coordinates of the mesh, and then outputs the discretized solutions fields as an out graph, where each node contains the corresponding nodal solution vector.

Similar to CNNs, the output solution graph is obtained by applying multiple graph convolution operations on the input layer, sequentially updating nodal features via a message passing function, which can be written in a generic form,

$$\boldsymbol{f}_i^{(l)} = \gamma^{(l)}(\boldsymbol{f}_i^{(l-1)}, \square_{j\in\mathcal{N}(i)}^{(l)} \Psi^{(l)}(\boldsymbol{f}_i^{(l-1)}, \boldsymbol{f}_j^{(l-1)})), \tag{2}$$

where i denotes ith node, (l) denotes lth layer, γ, Ψ are differentiable non-linear functions, and $\square$ denotes a differentiable, permutation-invariant function (e.g., summation, mean, or maximum). The feature vectors are represented by $\boldsymbol{f}_i^{(l)} \in \mathbb{R}^{N_{\boldsymbol{f}^{(l)}}}$ and $\boldsymbol{f}_i^{(l-1)} \in \mathbb{R}^{N_{\boldsymbol{f}^{(l-1)}}}$, where $N_{\boldsymbol{f}^{(l-1)}}$ and $N_{\boldsymbol{f}^{(l)}}$ are feature dimensions in $(l-1)$th and lth layers, respectively. For implementation simplicity, all the nodal features are usually concatenated and flattened as a larger vector $\boldsymbol{X}$. The information of edge connection is stored in a sparse matrix $\boldsymbol{A}$, known as the adjacency matrix. In practical implementation, the edges are used to implicitly formulate the adjacency matrix. In this work, the GCN is constructed based on the Chebyshev spectral graph convolution operator [41], which is derived from the spectral convolution theorem [42], where Chebyshev polynomials are introduced to avoid expensive eigen-decomposition. Specifically, the message passing function of Chebyshev graph convolution can be written as,

$$\boldsymbol{X}^l = \text{ReLU}\left(\sum_{k=1}^{K} \boldsymbol{Z}^{(l-1,k)} \cdot \boldsymbol{\Theta}^{(l-1,k)} + \boldsymbol{b}^{l-1}\right), \tag{3}$$

where $\boldsymbol{\Theta}^{(l-1,k)}$ are trainable parameters for the kth basis in the $(l-1)$th layer, $\boldsymbol{b}^{(l-1)}$ is an additive trainable bias vector, and the kth basis $\boldsymbol{Z}^{(l-1,k)}$ is calculated recursively as follows,

$$\begin{aligned}
\boldsymbol{Z}^{(l-1,1)} &= \boldsymbol{X}^{(l-1)}, \\
\boldsymbol{Z}^{(l-1,2)} &= \hat{\boldsymbol{L}} \cdot \boldsymbol{X}^{(l-1)}, \\
\boldsymbol{Z}^{(l-1,k)} &= 2\hat{\boldsymbol{L}} \cdot \boldsymbol{Z}^{(l-1,k-1)} - \boldsymbol{Z}^{(l-1,k-2)},
\end{aligned} \tag{4}$$

and

$$\begin{aligned}\hat{\boldsymbol{L}} &= \boldsymbol{L} - \boldsymbol{I} \\ \boldsymbol{L} &= \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\boldsymbol{D}^{-\frac{1}{2}}\end{aligned} \tag{5}$$

where $\boldsymbol{I}$ is an identity matrix and $\boldsymbol{D}$ represents the degree matrix of the graph. The Rectified Linear Unit (ReLU) [43] is chosen as the nonlinear activation function and polynomial order K is set as 10 in this work.

2.3. Variational PDE-informed loss function

The loss function is built based on the PDE residuals (Eq. (1)), such that the conservation laws are utilized to inform/drive the GCN training. The generic PDE for steady-state scenarios can be re-written as,

$$\nabla \cdot F(u, \nabla u; \boldsymbol{\mu}) = S(u, \nabla u; \boldsymbol{\mu}) \quad \text{in } \Omega, \tag{6}$$

where $u : \Omega \to \mathbb{R}^{N_c}$ is the solution variable, $F : \mathbb{R}^{N_c} \to \mathbb{R}^{N_c \times d}$ is the flux function, $S : \mathbb{R}^{N_c} \to \mathbb{R}^{N_c}$ is the source term, and $\nabla := (\partial_{x_1}, \ldots, \partial_{x_d})$ denotes the gradient operator defined in the physical domain. Eq. (6) can represent a wide range of static PDEs such as Poisson equation, linear elasticity equations, and Navier–Stokes equations.

2.3.1. Weak formulation of PDE residuals

For continuous FC-PINNs, the derivative terms for constructing the PDE-informed loss function are obtained by AD in point-wise manner, and the FCNN as a continuous trial function searches an infinite-dimensional solution space. Therefore, the infinite search space over-complicates the non-convex optimization for the network training and a massive amount of collocation points are usually required. In this work, we use a piecewise polynomial basis to reduce the dimension of the search space and facilitate physics-informed training/convergence. Specifically, the conservation laws (Eq. (6)) are discretized based using a nodal continuous Galerkin method and the trial space $\mathcal{V}_h^p$ is constructed by continuous piecewise polynomial basis functions

$$\mathcal{V}_h^p = \left\{ v \in [\mathcal{H}^1(\Omega)]^{N_c} \;\middle|\; v|_K \in [\mathcal{P}_p(K)]^{N_c},\ \forall K \in \mathcal{E}_h \right\}, \tag{7}$$

where $\mathcal{H}^1(\Omega)$ represents Sobolev spaces where weak derivatives up to order one are square integrable, $\mathcal{P}_p(K)$ is the space of polynomial functions of degree up to p defined on the element K, and $\mathcal{E}_h$ is the finite element mesh. The test space is set to be the same as the trial space $\mathcal{V}_h^p$ and the solution $u_h \in \mathcal{V}_h^p$ satisfies the weak formulation of the PDEs for any test function $\omega_h \in \mathcal{V}_h^p$,

$$\int_{\partial\Omega} \omega_h \cdot F(u_h, \nabla u_h; \boldsymbol{\mu}) n \, dS - \int_{\Omega} \nabla \omega_h : F(u_h, \nabla u_h; \boldsymbol{\mu}) \, dV = \int_{\Omega} \omega_h \cdot S(u_h, \nabla u_h; \boldsymbol{\mu}) \, dV. \tag{8}$$

We introduce a basis $\boldsymbol{\Phi}(x) \in \mathbb{R}^{N_U \times N_c}$ for $\mathcal{V}_h^p$ to express the test variables as $\omega_h(x) = \boldsymbol{\Phi}(x)^T \tilde{\boldsymbol{W}}$, where $\tilde{\boldsymbol{W}} \in \mathbb{R}^{N_U}$ are the coefficients of the test variable in the basis, which leads to an equivalent version of the Galerkin form

$$\int_{\partial\Omega} \boldsymbol{\Phi} \cdot F\Big(u_h, \nabla u_h; \boldsymbol{\mu}\Big) n \, dS - \int_{\Omega} \nabla \boldsymbol{\Phi} : F\Big(u_h, \nabla u_h; \boldsymbol{\mu}\Big) \, dV - \int_{\Omega} \boldsymbol{\Phi} \cdot S\Big(u_h, \nabla u_h; \boldsymbol{\mu}\Big) \, dV = 0. \tag{9}$$

using arbitrariness of the test function coefficients. We convert this to residual form by introducing $\{(\beta_i^v, \tilde{x}_i^v)\}_{i=1}^{N_{qv}}$ and $\{(\beta_i^s, \tilde{x}_i^s)\}_{i=1}^{N_{qs}}$ as the quadrature weights and points for integrals over Ω and $\partial\Omega$, respectively, to define the residual as

$$\begin{aligned}\boldsymbol{R}(\tilde{\boldsymbol{U}}; \boldsymbol{\mu}) = &\sum_{i=1}^{N_{qs}} \beta_i^s \boldsymbol{\Phi}(\tilde{x}_i^s) \cdot F\Big(\tilde{u}_h(\tilde{x}_i^s; \tilde{\boldsymbol{U}}), \nabla \tilde{u}_h(\tilde{x}_i^s; \tilde{\boldsymbol{U}}); \boldsymbol{\mu}\Big) n - \\ &\sum_{i=1}^{N_{qv}} \beta_i^v \nabla \boldsymbol{\Phi}(\tilde{x}_i^v) : F\Big(\tilde{u}_h(\tilde{x}_i^v; \tilde{\boldsymbol{U}}), \nabla \tilde{u}_h(\tilde{x}_i^v; \tilde{\boldsymbol{U}}); \boldsymbol{\mu}\Big) - \\ &\sum_{i=1}^{N_{qv}} \beta_i^v \boldsymbol{\Phi}(\tilde{x}_i^v) \cdot S\Big(\tilde{u}_h(\tilde{x}_i^v; \tilde{\boldsymbol{U}}), \nabla \tilde{u}_h(\tilde{x}_i^v; \tilde{\boldsymbol{U}}); \boldsymbol{\mu}\Big),\end{aligned} \tag{10}$$

Table 1
Summary of notation.

Notations	Description	Treatment in PI-GGN	
$\boldsymbol{\mu}$	PDE parameter	Constant (if known)	Trainable (if unknown)
β_i^v, β_i^s	Quadrature weights	Constant tensors	
$\boldsymbol{\Phi}(\cdot)$	Basis function	Constant tensors	
F, S	Flux and source functions	Differentiable functions	
$\hat{\boldsymbol{U}}$	Nodal solution	Output graph of the GCN	
$\boldsymbol{\chi}$	Nodal coordinates	Input graph of the GCN	

where $\tilde{u}_h : \Omega \times \mathbb{R}^{N_U} \to \mathbb{R}^{N_c}$ is the continuous representation in $\mathcal{V}_h^p$ of the discrete state vector, i.e.,

$$\tilde{u}_h(x; \tilde{\boldsymbol{U}}) = \boldsymbol{\Phi}(x)^T \tilde{\boldsymbol{U}}. \tag{11}$$

The surface and volume quadrature coefficients (β^s and β^v) are stored as constant tensors and remain unchanged during the network training. The matrix of basis function $\boldsymbol{\Phi}$ is obtained on the limited amount of quadrature points and can be pre-computed as constant tensors ($\boldsymbol{\Phi}(\tilde{x}^v)$, $\boldsymbol{\Phi}(\tilde{x}^s)$, $\nabla\boldsymbol{\Phi}(\tilde{x}^v)$, $\nabla\boldsymbol{\Phi}(\tilde{x}^s)$). The variational formulation of the PDE residual (Eq. (10)) will be used to define the physics-informed loss function for the GCN. Namely, the nodal solution vector $\tilde{\boldsymbol{U}}$ will be learned by GCN as the output graph $\hat{\boldsymbol{U}}(\boldsymbol{\Theta})$, which takes the coordinates ($\boldsymbol{\chi}$) as the input graph. When the PDE parameters $\boldsymbol{\mu}$ are unknown, they can be treated as trainable parameters, being updated along with network parameters $\boldsymbol{\Theta}$. Both the flux and source functions (F, S) are differentiable functions, where gradient information can be propagated from the outputs to their inputs. Table 1 summarizes these notations.

2.3.2. Essential boundary conditions enforcement

We apply static condensation to (10) by restricting to the unconstrained degrees of freedom, e.g., degrees of freedom away from essential BCs, to yield

$$\boldsymbol{R}_u(\boldsymbol{U}_u(\boldsymbol{\mu}), \boldsymbol{U}_e; \boldsymbol{\mu}) = 0, \tag{12}$$

where $\boldsymbol{U}_e$ are the known value of the essential boundary conditions and $\boldsymbol{U}_u(\boldsymbol{\mu})$ are the indices of $\boldsymbol{U}(\boldsymbol{\mu})$ corresponding to the unconstrained degrees of freedom. In the neural network setting, we enforce the essential boundary conditions strongly by partitioning the degrees of freedom into unconstrained (unknown) and constrained (known) degrees of freedom as $\hat{\boldsymbol{U}}(\boldsymbol{\Theta}) = (\hat{\boldsymbol{U}}_u(\boldsymbol{\Theta})^T, \hat{\boldsymbol{U}}_c^T)^T$ and defining the constrained degrees of freedom using the known value of the essential BCs, i.e., $\hat{\boldsymbol{U}}_c = \boldsymbol{U}_e$, and the unconstrained degrees of freedom by minimizing the physics-informed loss function

$$\mathcal{L}_{\mathrm{f}}(\boldsymbol{\Theta}; \boldsymbol{\mu}) = \left\| \boldsymbol{R}_u \left(\hat{\boldsymbol{U}}_u(\boldsymbol{\Theta}), \boldsymbol{U}_e; \boldsymbol{\mu} \right) \right\|_2. \tag{13}$$

In this formulation, the essential boundary condition will be satisfied automatically by construction, which is in contrast to continuous FC-PINN that defines the FCNN as a point-wise solution function, posing challenges in hard boundary enforcement.

2.4. Unifying forward and inverse solutions

The GCN can be trained based on the physics-informed loss function defined in Eq. (13) by solving the following optimization problem without labels,

$$\boldsymbol{\Theta}^* = \arg\min_{\boldsymbol{\Theta}} \mathcal{L}_{\mathrm{f}}(\boldsymbol{\Theta}; \bar{\boldsymbol{\mu}}), \tag{14}$$

where $\boldsymbol{\Theta}^*$ denotes optimal network parameters and $\bar{\boldsymbol{\mu}}$ are the known PDE parameters; the GCN is then used to solve a forward PDE (*forward solution*). However, in many cases, some physical parameters such as material properties, inlet velocity, and Reynolds number, are not available, while sparse observation data (labels) $\boldsymbol{U}_o$ can be obtained, which can be assimilated to infer the unknown parameters (*inverse solution*). In previous PINN approaches,

the inverse problem can be solved by assimilating data $\boldsymbol{U}_o$ in a soft manner, where the physics-informed loss is augmented by a data loss component. Namely, the following optimization is formulated,

$$(\boldsymbol{\Theta}^*, \boldsymbol{\mu}^*) = \underset{\boldsymbol{\Theta},\boldsymbol{\mu}}{\arg\min}\ \mathcal{L}_{\mathrm{f}}(\boldsymbol{\Theta};\boldsymbol{\mu}) + \lambda \underbrace{\left\| \boldsymbol{\mathcal{F}}^{s2o}\left(\hat{\boldsymbol{U}}(\boldsymbol{\Theta})\right) - \boldsymbol{U}_o \right\|_2}_{\text{data loss: } \mathcal{L}^d}, \tag{15}$$

where $\boldsymbol{\mathcal{F}}^{s2o}$ represents the state-to-observable map and λ is the penalty parameter. Properly tuning the penalty weight λ is critical to the convergence, which is, however, challenging and often conducted empirically [16]. Here we introduce a novel approach to assimilate observation data and infer unknown parameters without the need of hyperparameter tuning. Specifically, the observation data are strictly imposed by constructing the GCN output as

$$\boldsymbol{\mathcal{F}}^{s2o}\left(\hat{\boldsymbol{U}}(\boldsymbol{\Theta})\right) = \boldsymbol{U}_o \tag{16}$$

Therefore, unknown parameters $\boldsymbol{\mu}$ and boundary conditions $\hat{\boldsymbol{U}}_u$ can be obtained along with the PDE solutions $\hat{\boldsymbol{U}}_u$ simultaneously by solving the following constrained optimization problem,

$$(\boldsymbol{\Theta}^*, \boldsymbol{\mu}^*) = \underset{\boldsymbol{\Theta},\boldsymbol{\mu}}{\arg\min}\ \mathcal{L}_{\mathrm{f}}(\boldsymbol{\Theta};\boldsymbol{\mu}), \quad \text{subject to:} \quad \boldsymbol{\mathcal{F}}^{s2o}\left(\hat{\boldsymbol{U}}(\boldsymbol{\Theta})\right) = \boldsymbol{U}_o. \tag{17}$$

Finally, the algorithms are summarized below.

Algorithm 1 Solve forward PDE-governed problems via GCN

Input: PDE parameter $\bar{\boldsymbol{\mu}}$, node coordinates $\boldsymbol{\chi}$ and adjacency matrix $\boldsymbol{A}$
Output: The solution $\hat{\boldsymbol{U}}$
1.Pre-compute the matrix basis function $\boldsymbol{\Phi}$ on the qudrature points to obtain $\boldsymbol{\Phi}(\tilde{x}^v)$, $\boldsymbol{\Phi}(\tilde{x}^s)$, $\nabla\boldsymbol{\Phi}(\tilde{x}^v)$, $\nabla\boldsymbol{\Phi}(\tilde{x}^s)$;
2.Formulate the residual function $\boldsymbol{R}(\tilde{\boldsymbol{U}};\boldsymbol{\mu})$ defined in (10);
3.Apply the static condensation to (10) to obtain (12);
4.Partition the degrees of freedom $\hat{\boldsymbol{U}}(\boldsymbol{\Theta}) = (\hat{\boldsymbol{U}}_u(\boldsymbol{\Theta})^T, \hat{\boldsymbol{U}}_c^T)^T$ and enforce the essential condition, $\hat{\boldsymbol{U}}_c = \boldsymbol{U}_e$, to formulate the physics-informed loss function (13);
5.Solve the optimization problem (14) to obtain $\hat{\boldsymbol{U}} = (\hat{\boldsymbol{U}}_u(\boldsymbol{\Theta}^*)^T, \boldsymbol{U}_e^T)^T$;

Algorithm 2 Solve inverse PDE-governed problems via GCN

Input: Sparse observation $\boldsymbol{U}_o$, node coordinates $\boldsymbol{\chi}$ and adjacency matrix $\boldsymbol{A}$
Output: The solution $\hat{\boldsymbol{U}}$ and PDE parameter $\boldsymbol{\mu}^*$
1.Pre-compute the matrix basis function $\boldsymbol{\Phi}$ on the qudrature points to obtain $\boldsymbol{\Phi}(\tilde{x}^v)$, $\boldsymbol{\Phi}(\tilde{x}^s)$, $\nabla\boldsymbol{\Phi}(\tilde{x}^v)$, $\nabla\boldsymbol{\Phi}(\tilde{x}^s)$;
2.Formulate the residual function $\boldsymbol{R}(\tilde{\boldsymbol{U}};\boldsymbol{\mu})$ defined in (10);
3.Apply the static condensation to (10) to obtain (12);
4.Partition the degrees of freedom $\hat{\boldsymbol{U}}(\boldsymbol{\Theta}) = (\hat{\boldsymbol{U}}_u(\boldsymbol{\Theta})^T, \hat{\boldsymbol{U}}_c^T)^T$, enforce the essential condition, $\hat{\boldsymbol{U}}_c = \boldsymbol{U}_e$, and strictly impose the observation data (16), to formulate the physics-informed loss function (13);
5.Solve the optimization problem (17) to obtain $\boldsymbol{\mu}^*$ and $\hat{\boldsymbol{U}} = (\hat{\boldsymbol{U}}_u(\boldsymbol{\Theta}^*)^T, \boldsymbol{U}_e^T)^T$;

3. Numerical experiments

We demonstrate the proposed physics-informed graph Galerkin neural network (PI-GGN) on a variety of computational mechanics problems in both forward and inverse settings. Specifically, Poisson equations, linear elasticity equations, and Navier–Stokes equations with known or unknown BCs/parameters are investigated here to demonstrate the effectiveness of the proposed method. Moreover, we also compare two different ways of assimilating sparse observation data and show the advantage of strictly enforcing data for the parameter/field inversion. For all cases, the GCN architecture remains the same, where the dimensions of node vector in hidden graph lays are fixed as [32, 64, 128, 256, 128, 64, 32]. The relative error metric e is defined as,

$$e = \frac{\|\hat{\boldsymbol{U}}(\boldsymbol{\Theta}^*) - \boldsymbol{U}(\bar{\boldsymbol{\mu}})\|_2}{\|\boldsymbol{U}(\bar{\boldsymbol{\mu}})\|_2}, \tag{18}$$

where $\boldsymbol{\Theta}^*$ is the optimal training parameters computed for the parameter configuration $\bar{\boldsymbol{\mu}}$.

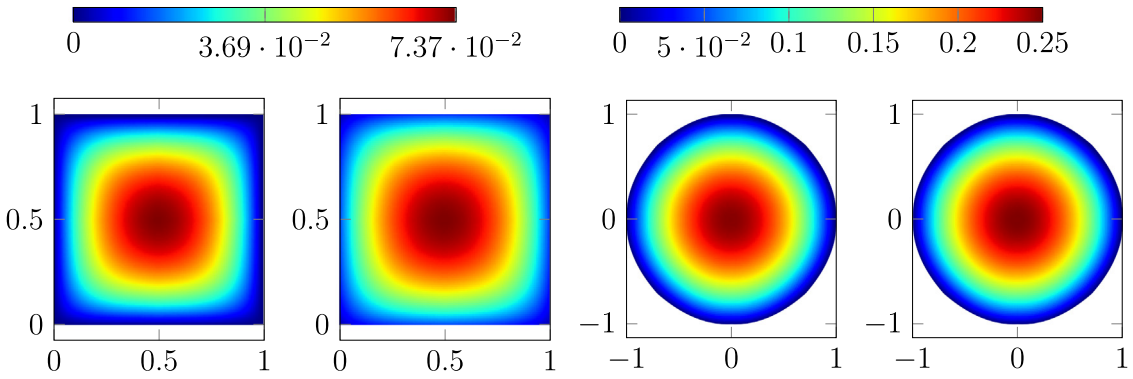


Fig. 2. PI-GGN forward solutions of the diffusion field u on the (a) square and (c) circular disks, compared against corresponding FEM or analytical solutions, where the relative prediction error of the PI-GGN is $e = 5 \times 10^{-3}$ on the square domain and $e = 5 \times 10^{-4}$ on the circular disk, respectively.

3.1. Poisson equation

We start from a 2-D homogeneous Poisson equation,

$$\begin{aligned} f + \Delta u &= 0 \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \end{aligned} \tag{19}$$

where u is the primary variable, f is the source term, and Δ denotes the Laplacian operator.

3.1.1. Forward solution of diffusion field

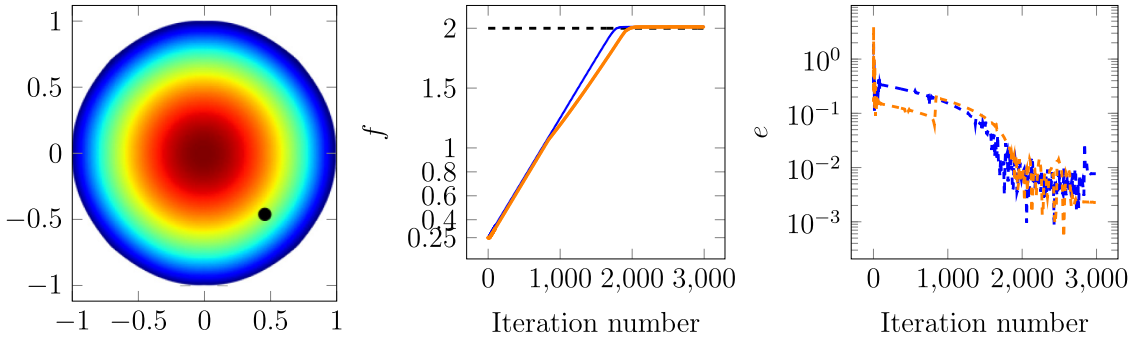
We first consider the forward problem, where the source term f is given ($f = 1$) over a unit square domain (Figs. 2a and 2b). Four quadrilateral elements are used to discretize the domain with 3rd order of polynomial basis for solution and domain transformation. As a result, the total nodal points of the graph is 49, which is much lower than the total number of collocation points for a typical point-wise FC-PINN. The contour of the PI-GGN prediction is in a good agreement with the FEM reference, and the relative error is $e = 0.5\%$, though slight under-estimation near the boundary is observed. In Fig. 2c, the same PDE is solved on a unit circular domain, where the analytical solution exists (Fig. 2d),

$$u(x, y) = \frac{1 - x^2 - y^2}{4}. \tag{20}$$

In PI-GGN, the number of elements remains the same, while the order of the polynomial basis is set as two and thus 25 nodal points are used to construct the graph. We can see the PI-GGN forward solution is almost identical to the analytical reference and the relative prediction error e is only 0.05%. This simple test case demonstrates that the graph-based discrete PINN can easily handle non-rectangular domains with unstructured meshes, which have posed challenges for standard FD-based CNN architectures, where special treatment such as rasterization or coordinate transformation is required [20], complicating the implementation and convergence.

3.1.2. Inverse solution of unknown source term

The real power of the PI-GGN is to solve the forward and inverse problems simultaneously by assimilating additional state observations. For example, when the source term is not given, the PI-GGN is able to assimilate sparse data to solve the diffusion field and meanwhile infer the unknown source term in a unified manner. Here we assume the constant source term $f = 2$ is unknown and observation of u is available only at one point as shown in Fig. 3a. We use two ways to assimilate the data and solve the inverse problem: one is to assimilate data by adding a data loss as a penalty term with (Eq. (15)) with the hyper-parameter chosen as $\lambda = 1000$, and the other is to assimilate data strictly based on Eq. (17). As shown in Fig. 3b, the inferred source terms from both approaches



(a) PI-GGN prediction (b) Value of inferred constant source (c) Error of predicted u field

Fig. 3. PI-GGN inverse solutions of the source term f by assimilating observed diffusion data (black dots) using (1) penalty method (—•—) and (2) hard enforcement (——), compared against the ground truth (- - -), where the error of field prediction by soft (- - -) and hard (- - -) data assimilation are presented.

converge to the ground truth and the forward solution of the u field is also obtained *simultaneously*. Overall, the prediction errors of the unknown source term and diffusion field are less than 1%.

3.2. Linear elasticity equations

Next, we consider problems governed by linear elasticity equations,

$$\begin{aligned} \nabla \cdot \sigma &= 0 \quad \text{in } \Omega, \\ \sigma \cdot n &= t \quad \text{on } \partial\Omega^N, \\ u &= u^D \quad \text{on } \partial\Omega^D, \end{aligned} \tag{21}$$

where $u : \Omega \to \mathbb{R}^d$ is the displacement vector, $\sigma : \Omega \to \mathbb{R}^{d\times d}$ is the stress tensor defined as $\sigma_{ij} = \lambda u_{kk}\delta_{ij} + \mu(u_{i,j} + u_{j,i})$, $n : \partial\Omega \to \mathbb{R}^d$ is the unit normal vector on the boundary, $t : \partial\Omega^N \to \mathbb{R}^d$ is the applied traction force, $u^D : \partial\Omega^D \to \mathbb{R}^d$ is the essential boundary condition, and λ and μ are the constant Lamé parameters. For each variable component u_i, a sub-GCN is constructed for the prediction.

3.2.1. Forward solution of displacement field

First, we solve the forward problem in a unit square domain. To discretize this domain, four quadrilateral elements are used, and the order of polynomial basis for solution and domain transformation is set as two, resulting in a 25-nodal graph. The Lamé parameters are set as $\lambda = 1$ and $\mu = 1$. The essential boundary condition $u = [0, 0]$ is prescribed on the left side ($x = 0$) and the natural boundary condition $t = [0.5, 0]$ is imposed on the right side. Fig. 4 shows that the PI-GGN forward solution of the displacement field agrees with the FEM reference very well.

Then we investigate an irregular domain, a rectangular with a notch, where same Lamé parameters are specified. The domain is discretized by 55 simplex elements with 1st order polynomial basis for the solution and domain transformation. The essential boundary conditions $u^D = [0, 0]$ are imposed at the left boundary side $x = -0.4$ and the natural boundary condition $t_1 = 0.5$ is prescribed at the right boundary side. As mentioned above, no special treatment is needed for PI-GGN to handle irregular geometry with simplex mesh. Fig. 5 shows that the forward solution by PI-GGN is very accurate compared to the FEM reference.

Lastly, we consider a 3-D domain. Specifically, the deformation of a 3-D hollow cylinder is solved by the PI-GGN. The essential boundary conditions $u^D = [0, 0, 0]$ are imposed at the left surface, the Neumann boundary conditions $t = -n$ are prescribed at the inner surface of the cylinder ($x^2 + y^2 = 1$), and $t = [0, 0, -0.25]$ are imposed at the right surface. The second order polynomial basis is used and the number of hexahedral element is 40 with 440 nodal points. The Lamé parameters are set as $\lambda = 0.73$ and $\mu = 0.376$. The forward solution of the displacement by PI-GGN agrees with the FEM reference reasonably well, though PI-GGN slightly over-predicts the displacement of the right end of the cylinder (Fig. 6).

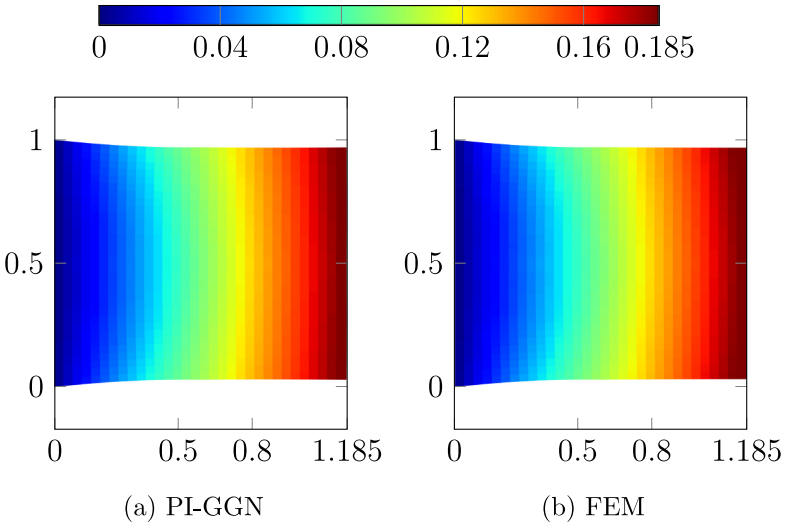


Fig. 4. PI-GGN forward solutions of the displacement field u, compared against corresponding FEM reference, where the relative prediction error of the PI-GGN is $e = 1 \times 10^{-2}$.

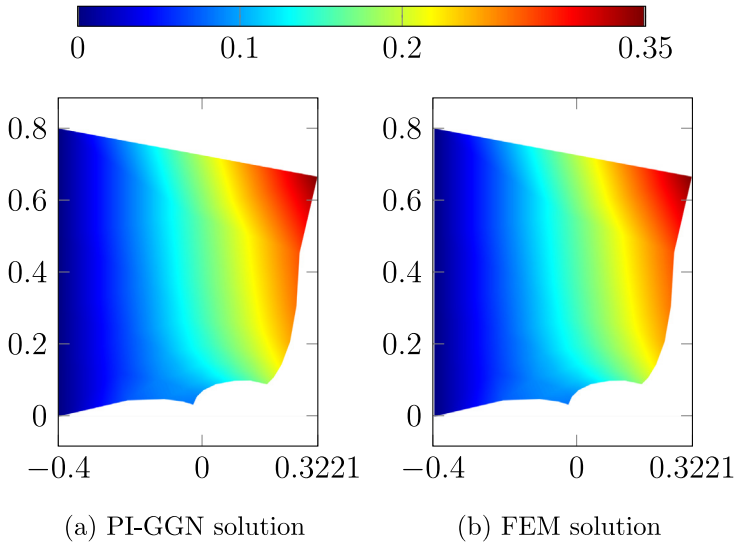
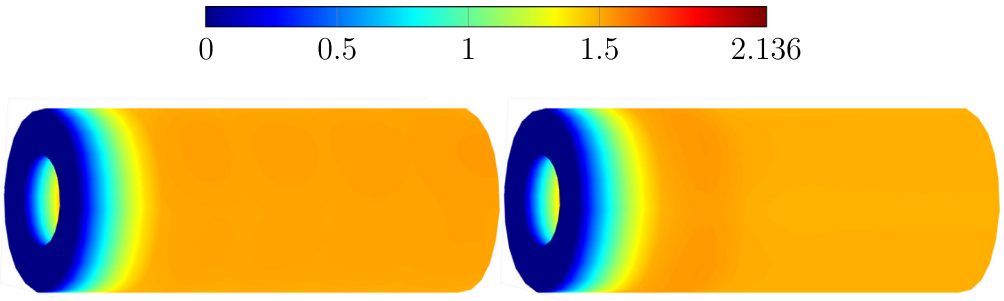


Fig. 5. PI-GGN forward solutions of the displacement field u, compared against corresponding FEM reference, where the relative prediction error of the PI-GGN is $e = 5 \times 10^{-3}$.

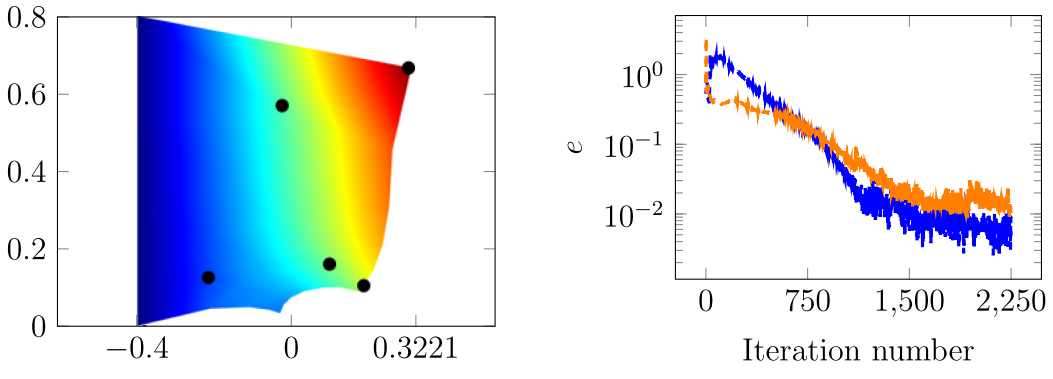
3.2.2. Inverse solution of unknown material properties

Next, we solve an inverse problem governed by the linear elasticity equations (Eq. (21)). The Lamé parameters (λ and μ) are assumed to be unknown, whose true values are set as $\lambda = \mu = 1$. The displacement field is observed at five randomly selected points shown in Fig. 7(a). The entire field of the displacement is obtained via PI-GGN and the Lamé parameters can be inferred accurately as well (Figs. 7c and 7d). The relative error of the PI-GGN predicted displacement field is 0.005 by assimilating data in a hard manner, which is slightly lower than that of using penalty method ($e = 0.01$), shown in Fig. 7b. The convergence speed and inference accuracy depend on the quantity and quality of sensor data, i.e., number and location of sensors. As shown in Fig. 8, the network requires more than 4000 iterations to converge to the true Lamé parameters if only two sensors are used, which is much slower than that with five sensors. Where to place the sensors also affects inference speed and accuracy, known as

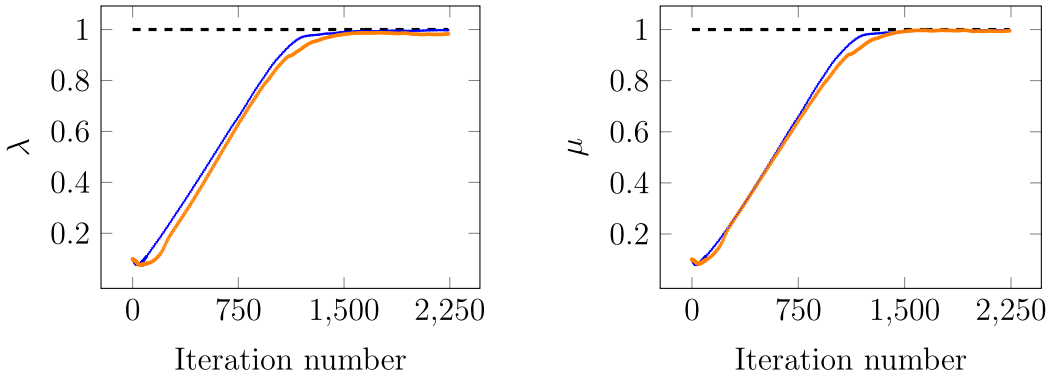


(a) PI-GNN (b) FEM

Fig. 6. PI-GGN forward solutions of the displacement field u, compared against corresponding FEM reference, where the relative prediction error of the PI-GGN is $e = 5 \times 10^{-2}$.



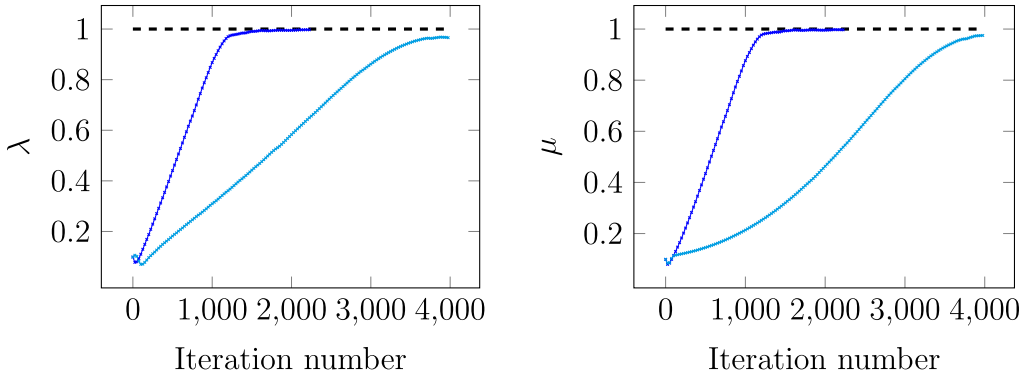
(a) PI-GGN prediction (b) Errors of predicted u field



(c) λ (d) μ

Fig. 7. PI-GGN inverse solutions of the Lamé parameters by assimilating observed displacement data (black dots) using (1) penalty method (—·—) and (2) hard enforcement approach (——), compared against the ground truth (- - -), where the error of field prediction by soft (- - -) and hard (- - -) data assimilation are presented.

design of experiments (DOE), making observation data more informative. However, the DOE study is out of the scope of this work.



(a) λ (b) μ

Fig. 8. PI-GGN inverse solutions of the Lamé parameters by assimilating observed data from different number of sensors: five sensors (—•—) and two sensors (—•—).

3.3. Naiver-Stokes equations

In the last test case, we study forward and inverse problems governed by the static incompressible Navier–Stokes (NS) equations, which is more challenging due to its strong nonlinearity. The steady NS equations model the viscous fluid flow with a constant density, which can be expressed as,

$$\begin{aligned} (v \cdot \nabla)v - \nu \Delta v + \nabla p = 0, \quad \nabla \cdot v = 0 \quad &\text{in } \Omega, \\ v = v^D \quad &\text{on } \partial\Omega^D, \\ \nu(n \cdot \nabla)v - pn = 0 \quad &\text{on } \partial\Omega^N, \end{aligned} \tag{22}$$

where $v : \Omega \to \mathbb{R}^d$ is the velocity vector, $p : \Omega \to \mathbb{R}$ is the pressure, ν is the viscosity of the fluid, and $n : \partial\Omega \to \mathbb{R}^d$ is the unit outward normal vector to the boundary. The solution variable vector is denoted by $u = [v_1, v_2, p]$. The viscosity is set as $\nu = 0.01$. For stability reasons, a mixed element approximation is adopted [44]. A separate sub-net is constructed for prediction of each of the solution variables v_1, v_2 and p.

3.3.1. Forward solution of velocity and pressure fields

First, we test the proposed approach on a classic flow problem, lid-driven cavity flow, defined on a square domain. The lid is placed on the top edge and moves rightward ($v_1 = 1, v_2 = 0$). The remaining three edges are set as no-slip walls ($v_1 = v_2 = 0$). The domain is discretized by 100 quadrilateral elements. The numbers of collocation points for velocity and pressure fields are 441 and 121, respectively. The contours of forward solutions of velocity and pressure by PI-GGN are in a good agreement with the corresponding FEM reference, as shown in Fig. 9. The relative prediction errors are less than 1%. It is worth noting that over 10000 collocation points were used to achieve same level of accuracy for AD-based FC-PINN [16,17].

We also test the PI-GGN on solving the fluid flow in an idealized stenosis, where the inlet velocity is set as $v^D = [0, 1]$ at the bottom ($y = 0$) and no-traction boundary condition is prescribed at the outlet on the top ($y = 0$). The same finite element setting is used as the lid-driven cavity problem. Similarly, both the velocity and pressure fields can be accurately solved and the PI-GGN predictions agree with the FEM reference well (see Fig. 10).

3.3.2. Inverse solution of unknown inlet velocity field and unobserved pressure field

Lastly, we consider an inverse problem governed by the NS equations. In particular, the inlet velocity field is assumed unknown and will be inferred by assimilating sparse velocity observation data as shown in Fig. 11b. The true inlet has a parabolic profile as shown in Fig. 11e. The functional form of the profile is not predefined in solving the inverse problem. Namely, the dimension of the inversion is equal to the degrees of free of the inlet, which is more than 20. By assimilating velocity observation data at sparse locations, our proposed method can accurately infer the unknown inlet velocity profile and also recover the entire velocity and pressure fields very well. However,

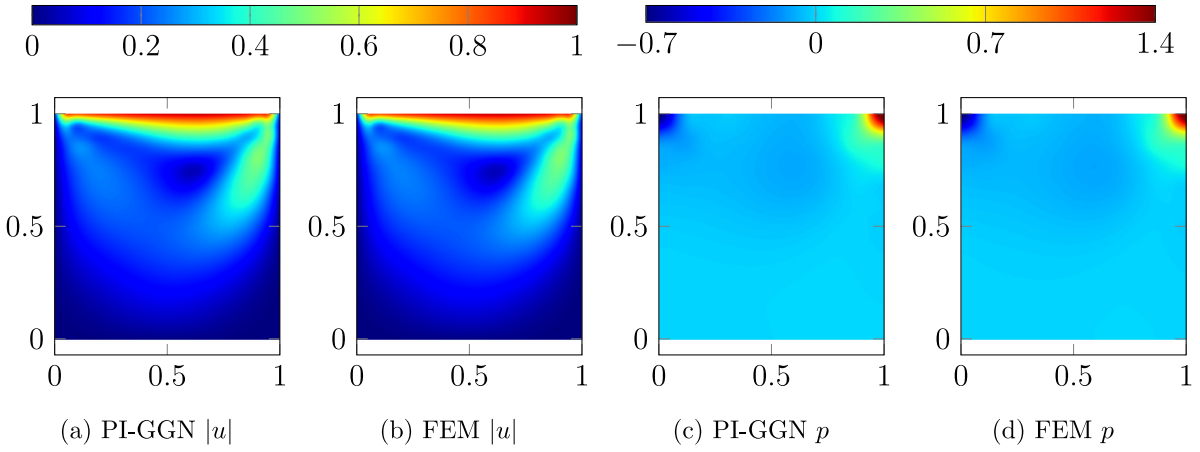


Fig. 9. PI-GGN forward solutions of the velocity magnitude and pressure fields, compared against corresponding FEM reference, where the relative errors are 8.7×10^{-3} for the velocity prediction and 1.95×10^{-2} for the pressure prediction.

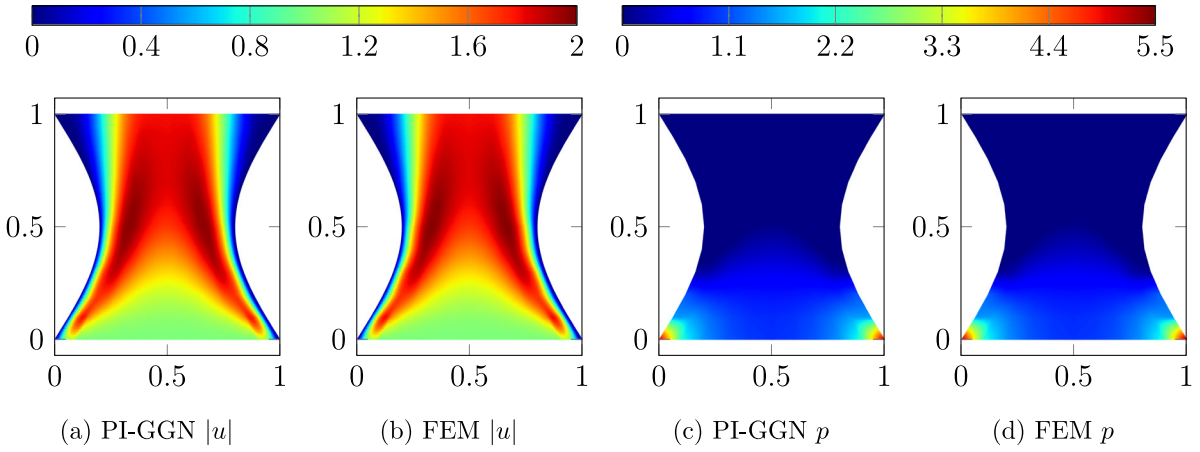


Fig. 10. PI-GGN forward solutions of the velocity magnitude and pressure fields, compared against corresponding FEM reference, where the relative errors are 4.4×10^{-3} for the velocity prediction and 1.8×10^{-2} for the pressure prediction.

it is observed that inferred inlet from the penalty-based data assimilation approach is not quite accurate, which notably deviates from the ground truth. Despite using same penalty coefficient as the previous cases, the inference performance significantly deteriorates. The proposed way of assimilating data strictly can avoid hyperparameter tuning and have better robustness. The problem will become more ill-posed to infer the space-dependent variables of the entire domain (spatial field) given limited observation data. But this can be addressed by projecting the spatial field onto reduced basis such as radial functions or Karhunen–Loéve modes to reduce the dimension and make the problem less ill-posed.

4. Discussion

4.1. Comparison between GCN and CNN

It is known that the classic CNN was originally developed for image recognition and processing, where the convolution filters were designed for images or image-like data, i.e., rectangular domain with uniform meshes. Therefore, CNNs are not directly applicable to cases studied in this work. To enable the use of the classic CNNs, 'rasterization' is required to preprocess nonuniform mesh data with an irregular domain [45–47]. However, the rasterization will also introduce issues and make the PDE-constrained learning challenging or even intractable. We

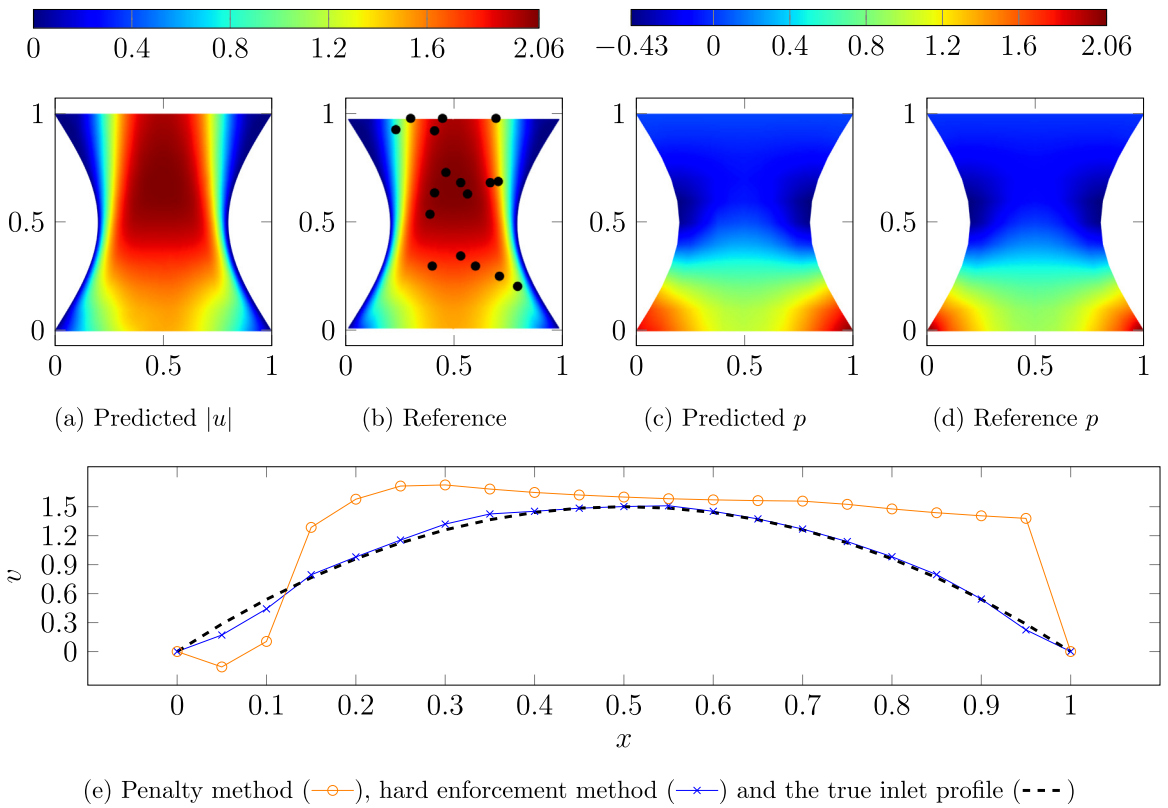


Fig. 11. PI-GGN inverse solutions of the inlet velocity field by assimilating observed velocity data at 19 randomly selected points. The relative error of the Inferred inlet profile is $e = 0.4$ by the soft penalty method while $e = 0.04$ by hard enforcement approach.

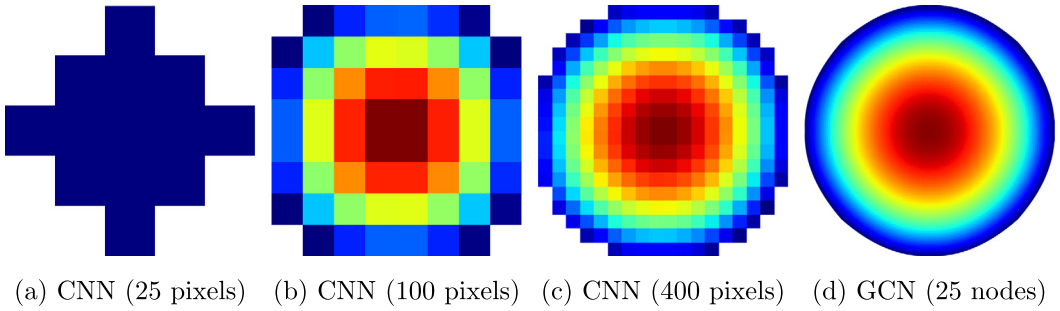


Fig. 12. The solution contours from CNN and GCN-based framework.

use the unit circular disc case to demonstrate the merit of the proposed GCN-based framework over CNN ones. Fig. 12 compares the solution contours of CNN at different resolutions with that of GCN. Even for a simple circular shape, massive nodes are needed to roughly resolve the field (see Fig. 12c). With the same number of nodes (25) used in unstructured mesh, pixel-based representation does not work. Due to the lack of smoothly resolved field and strictly-imposed boundary, the CNN-based method does not perform well. Even with $\times 8$ resolution (400 pixels), the solution of CNN is less accurate than GCN (around two-magnitude less accurate). Moreover, compared to finite-volume (FV)-based iterative solver, the GCN also shows advantages due to much fewer iterations needed. Fig. 13 further shows the merit of the proposed GCN-based framework in terms of training efficiency and convergence rate.

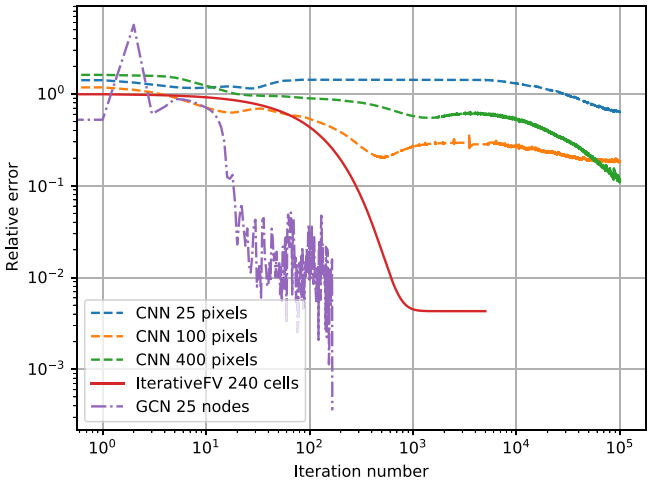


Fig. 13. Comparison of training curves of CNN-, GCN-, and FV-based methods.

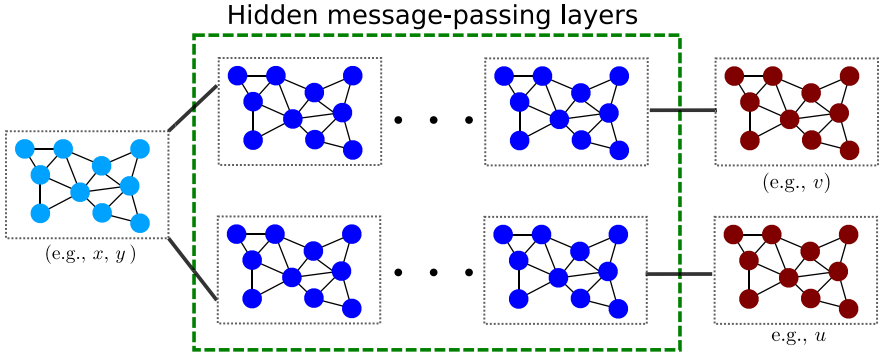


Fig. A.14. The GCN architecture directly operating unstructured meshes. The solution field of each physical variable is described by a separate subnet.

4.2. Future perspectives of the current framework

The most important contribution of this work lies in the algorithmic development that allows the direct use of GCN to solve forward and inverse PDEs on an unstructured mesh. The proposed framework is generic and can be coupled with various state-of-the-art GCNs [48–52] (available in torch geometric https://pytorch-geometric.readthedocs.io/en/latest/, which has lots of off-the-shelf examples). Although the current work is focused on steady-state PDEs, the proposed GCN-based PDE learning approach can also be extended to solve spatiotemporal PDEs with further developments. For example, the time derivatives in the physics-informed loss function can be formulated based on the forward Euler scheme, and graph convolution-based recurrent network or transformer can be constructed to capture the temporal dependence [38]. Moreover, the proposed framework can also be adapted for super-resolution (SR) tasks, where the input is a noisy low-resolution field and the output is super-resolved based on PDE-informed training. The proposed framework has the potential to broaden the application of SR to irregular geometries with unstructured meshes, since most existing SR works still heavily rely on CNNs [53,54]. Also, the GCN can be formulated as a Bayesian network via variational inference to deal with data noise and quantify the uncertainty [55].

5. Conclusion

In this paper, a novel discrete PINN framework is proposed for solving both forward and inverse problems governed by PDEs in a unified manner. Built upon the combination of graph convolutional networks (GCNs)

Table A.2
Training details of GCN.

Batch size	Optimizer	Learning rate	Activation	# Hidden layer	Device
1	Adam [56]	10^{-4}	ReLU	7	RTX2080
Poisson	Unit disc	Square	Inverse		
# Epoch	350	500	3000		
Linear elasticity	Square	Rectangular with notch		3D Cylinder	Inverse
# Epoch	4500	10000		18000	2250
Navier–Stokes	Cavity	Stenosis	Inverse		
# Epoch	33000	10000	10000		

and Galerkin variational formulation of physics-informed loss functions, the proposed PINN can naturally handle irregular domains with unstructured meshes, where the training is performed in an efficient way due to the reduced search space by polynomials. Thanks to the hard enforcement of boundary conditions and sparse observation data, the proposed method does not require tuning penalty parameters and has better robustness. The numerical results from several forward and inverse problems governed by linear and nonlinear PDEs have shown the effectiveness of the proposed method. Furthermore, the authors believe this work contributes to facilitating the healthy combination of scientific deep learning and classic numerical techniques rather than isolating them against each other.

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.

Acknowledgments

The authors would like to acknowledge the funds from National Science Foundation, United States of America under award numbers CMMI-1934300 and OAC-2047127 (JXW, HG), the Air Force Office of Scientific Research (AFOSR), United States of America under award number FA9550-20-1-0236 (MZ), and startup funds from the College of Engineering at University of Notre Dame, United States of America in supporting this study. The authors finally thank the anonymous reviewers for their insightful comments and suggestions to improve the quality of this paper.

Appendix A. Implementation details

Throughout this work, we used a standard feed-forward graph convolutional neural networks (GCN) architecture as shown in Fig. A.14. The training details are summarized in Table A.2. The implementation is based on PyTorch Geometric (PyG) package, which is available at https://pytorch-geometric.readthedocs.io/en/latest/. For dataset information and other details, please refer to the code repository, which is available at https://github.com/Jianxun-Wang/graphGalerkin upon publication.

References

[1] M. Raissi, P. Perdikaris, G.E. 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.
[2] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, Automatic differentiation in machine learning: a survey, J. Mach. Learn. Res. 18 (2018).
[3] Y. Chen, L. Lu, G.E. Karniadakis, L. Dal Negro, Physics-informed neural networks for inverse problems in nano-optics and metamaterials, Opt. Express 28 (8) (2020) 11618–11633.
[4] 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.
[5] C. Rao, H. Sun, Y. Liu, Physics-informed deep learning for computational elastodynamics without labeled data, J. Eng. Mech. 147 (8) (2021) 04021043.
[6] Z. Chen, Y. Liu, H. Sun, Deep learning of physical laws from scarce data, 2020, arXiv preprint arXiv:2005.03448.

[7] F. Sahli Costabal, Y. Yang, P. Perdikaris, D.E. Hurtado, E. Kuhl, Physics-informed neural networks for cardiac activation mapping, Front. Phys. 8 (2020) 42.
[8] L. Sun, H. Gao, S. Pan, J.-X. Wang, Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data, Comput. Methods Appl. Mech. Engrg. 361 (2020) 112732.
[9] M. Raissi, A. Yazdani, G.E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, Science 367 (6481) (2020) 1026–1030.
[10] G. Kissas, Y. Yang, E. Hwuang, W.R. Witschey, J.A. Detre, P. Perdikaris, Machine learning in cardiovascular flows modeling: Predicting arterial blood pressure from non-invasive 4D flow MRI data using physics-informed neural networks, Comput. Methods Appl. Mech. Engrg. 358 (2020) 112623.
[11] S. Cai, H. Li, F. Zheng, F. Kong, M. Dao, G.E. Karniadakis, S. Suresh, Artificial intelligence velocimetry and microaneurysm-on-a-chip for three-dimensional analysis of blood flow in physiology and disease, Proc. Natl. Acad. Sci. 118 (13) (2021).
[12] A. Arzani, J.-X. Wang, R.M. D'Souza, Uncovering near-wall blood flow from sparse data with physics-informed neural networks, Phys. Fluids (2021).
[13] X. Jin, S. Cai, H. Li, G.E. Karniadakis, NSFnets (Navier–Stokes flow nets): Physics-informed neural networks for the incompressible Navier–Stokes equations, J. Comput. Phys. 426 (2021) 109951.
[14] Z. Mao, A.D. Jagtap, G.E. Karniadakis, Physics-informed neural networks for high-speed flows, Comput. Methods Appl. Mech. Engrg. 360 (2020) 112789.
[15] O. Hennigh, S. Narasimhan, M.A. Nabian, A. Subramaniam, K. Tangsali, M. Rietmann, J.d.A. Ferrandis, W. Byeon, Z. Fang, S. Choudhry, NVIDIA SimNetˆ{TM}: An AI-accelerated multi-physics simulation framework, 2020, arXiv preprint arXiv:2012.07938.
[16] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient pathologies in physics-informed neural networks, 2020, arXiv preprint arXiv:2001.04536.
[17] A.D. Jagtap, E. Kharazmi, G.E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems, Comput. Methods Appl. Mech. Engrg. 365 (2020) 113028.
[18] J. Berg, K. Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries, Neurocomputing 317 (2018) 28–41.
[19] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, P. Perdikaris, Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data, J. Comput. Phys. 394 (2019) 56–81.
[20] H. Gao, L. Sun, J.-X. Wang, PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain, J. Comput. Phys. (2020) 110079.
[21] N. Geneva, N. Zabaras, Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks, J. Comput. Phys. 403 (2020) 109056.
[22] R. Zhang, Y. Liu, H. Sun, Physics-guided convolutional neural network (PhyCNN) for data-driven seismic response modeling, Eng. Struct. 215 (2020) 110704.
[23] R. Zhang, Y. Liu, H. Sun, Physics-informed multi-LSTM networks for metamodeling of nonlinear structures, Comput. Methods Appl. Mech. Engrg. 369 (2020) 113226.
[24] N. Wandel, M. Weinmann, R. Klein, Teaching the incompressible Navier–Stokes equations to fast neural surrogate models in three dimensions, Phys. Fluids 33 (4) (2021) 047117.
[25] R. Ranade, C. Hill, J. Pathak, Discretizationnet: A machine-learning based solver for Navier–Stokes equations using finite volume discretization, Comput. Methods Appl. Mech. Engrg. 378 (2021) 113722.
[26] T.J. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Courier Corporation, 2012.
[27] C.A. Duarte, J.T. Oden, H-p clouds—An h-p meshless method, Numer. Methods Partial Differential Equations: Int. J. 12 (6) (1996) 673–705.
[28] E. Weinan, B. Yu, The deep ritz method: A deep learning-based numerical algorithm for solving variational problems, Commun. Math. Stat. 6 (1) (2018) 1–12.
[29] E. Samaniego, C. Anitescu, S. Goswami, V.M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, T. Rabczuk, An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications, Comput. Methods Appl. Mech. Engrg. 362 (2020) 112790.
[30] Y. Zang, G. Bao, X. Ye, H. Zhou, Weak adversarial networks for high-dimensional partial differential equations, J. Comput. Phys. 411 (2020) 109409.
[31] E. Kharazmi, Z. Zhang, G.E. Karniadakis, Variational physics-informed neural networks for solving partial differential equations, 2019, arXiv preprint arXiv:1912.00873.
[32] E. Kharazmi, Z. Zhang, G.E. Karniadakis, hp-VPINNs: Variational physics-informed neural networks with domain decomposition, Comput. Methods Appl. Mech. Engrg. 374 (2021) 113547.
[33] R. Khodayi-Mehr, M. Zavlanos, Varnet: Variational neural networks for the solution of partial differential equations, in: Learning for Dynamics and Control, PMLR, 2020, pp. 298–307.
[34] H. Yao, Y. Gao, Y. Liu, FEA-net: A physics-guided data-driven model for efficient mechanical response prediction, Comput. Methods Appl. Mech. Engrg. 363 (2020) 112892.
[35] A. Sanchez-Gonzalez, N. Heess, J.T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, P. Battaglia, Graph networks as learnable physics engines for inference and control, in: International Conference on Machine Learning, PMLR, 2018, pp. 4470–4479.
[36] P.W. Battaglia, R. Pascanu, M. Lai, D. Rezende, K. Kavukcuoglu, Interaction networks for learning about objects, relations and physics, 2016, arXiv preprint arXiv:1612.00222.
[37] R. Maulik, P. Balaprakash, Site-specific graph neural network for predicting protonation energy of oxygenate molecules, 2019, arXiv preprint arXiv:2001.03136.

[38] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, P.W. Battaglia, Learning mesh-based simulation with graph networks, 2020, arXiv preprint arXiv:2010.03409.
[39] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, P. Battaglia, Learning to simulate complex physics with graph networks, in: International Conference on Machine Learning, PMLR, 2020, pp. 8459–8468.
[40] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Multipole graph neural operator for parametric partial differential equations, 2020, arXiv preprint arXiv:2006.09535.
[41] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, Adv. Neural Inf. Process. Syst. 29 (2016) 3844–3852.
[42] E. BrianDavies, G. Gladwell, J. Leydold, P. Stadler, Discrete nodal domain theorems, Linear Algebra Appl. 336 (1–3) (2001) 51–60.
[43] V. Nair, G.E. Hinton, Rectified linear units improve restricted boltzmann machines, in: ICML, 2010.
[44] P. Letallec, A mixed finite element approximation of the Navier–Stokes equations, Numer. Math. 35 (4) (1980) 381–404.
[45] J. Xu, K. Duraisamy, Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics, Comput. Methods Appl. Mech. Engrg. 372 (2020) 113379.
[46] S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, S. Kaushik, Prediction of aerodynamic flow fields using convolutional neural networks, Comput. Mech. 64 (2) (2019) 525–545.
[47] N. Geneva, N. Zabaras, Transformers for modeling physical systems, 2020, arXiv preprint arXiv:2010.03957.
[48] D. Duvenaud, D. Maclaurin, J. Aguilera-Iparraguirre, R. Gómez-Bombarelli, T. Hirzel, A. Aspuru-Guzik, R.P. Adams, Convolutional networks on graphs for learning molecular fingerprints, 2015, arXiv preprint arXiv:1509.09292.
[49] M. Schlichtkrull, T.N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, M. Welling, Modeling relational data with graph convolutional networks, in: European Semantic Web Conference, Springer, 2018, pp. 593–607.
[50] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, K. Weinberger, Simplifying graph convolutional networks, in: International Conference on Machine Learning, PMLR, 2019, pp. 6861–6871.
[51] F.M. Bianchi, D. Grattarola, L. Livi, C. Alippi, Graph neural networks with convolutional arma filters, IEEE Trans. Pattern Anal. Mach. Intell. (2021).
[52] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, Y. Sun, Masked label prediction: Unified message passing model for semi-supervised classification, 2020, arXiv preprint arXiv:2009.03509.
[53] H. Gao, L. Sun, J.-X. Wang, Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels, Phys. Fluids 33 (7) (2021) 073603.
[54] L. Guo, S. Ye, J. Han, H. Zheng, H. Gao, D.Z. Chen, J.-X. Wang, C. Wang, SSR-VFD: Spatial super-resolution for vector field data analysis and visualization, in: 2020 IEEE Pacific Visualization Symposium (PacificVis), IEEE Computer Society, 2020, pp. 71–80.
[55] L. Sun, J.-X. Wang, Physics-constrained Bayesian neural network for fluid flow reconstruction with sparse and noisy data, Theoret. Appl. Mech. Lett. 10 (3) (2020) 161–169.
[56] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.