Journal of Computational Physics 428 (2021) 110079

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain

Han Gao, Luning Sun, Jian-Xun Wang *

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

ARTICLE INFO

Article history:
Received 22 April 2020
Received in revised form 8 October 2020
Accepted 10 December 2020
Available online 17 December 2020

Keywords:
Physics-informed neural networks
Label-free
Surrogate modeling
Physics-constrained deep learning
Partial differential equations
Navier-Stokes

ABSTRACT

Recently, the advent of deep learning has spurred interest in the development of physics-informed neural networks (PINN) for efficiently solving partial differential equations (PDEs), particularly in a parametric setting. Among all different classes of deep neural networks, the convolutional neural network (CNN) has attracted increasing attention in the scientific machine learning community, since the parameter-sharing feature in CNN enables efficient learning for problems with large-scale spatiotemporal fields. However, one of the biggest challenges is that CNN only can handle regular geometries with image-like format (i.e., rectangular domains with uniform grids). In this paper, we propose a novel physics-constrained CNN learning architecture, aiming to learn solutions of *parametric PDEs on irregular domains without any labeled data*. In order to leverage powerful classic CNN backbones, elliptic coordinate mapping is introduced to enable coordinate transforms between the irregular physical domain and regular reference domain. The proposed method has been assessed by solving a number of steady-state PDEs on irregular domains, including heat equations, Navier-Stokes equations, and Poisson equations with parameterized boundary conditions, varying geometries, and spatially-varying source fields. Moreover, the proposed method has also been compared against the state-of-the-art PINN with fully-connected neural network (FC-NN) formulation. The numerical results demonstrate the effectiveness of the proposed approach and exhibit notable superiority over the FC-NN based PINN in terms of efficiency and accuracy.

1. Introduction

Physical phenomena in science and engineering are usually modeled by partial differential equations (PDEs), whose states live in infinite-dimensional spaces. Due to the lack of analytical solutions in most cases, their finite-dimensional approximations are resorted to based on traditional numerical approaches, e.g., finite difference (FD), finite volume (FV), and finite element (FE) methods, which have been developed and advanced over the past several decades. Nonetheless, the traditional numerical solvers often require significant computational efforts particularly for complex systems with multiscale/multi-physics features and might not even be feasible in real-time or many-query applications, e.g., optimization, inverse problem, and uncertainty quantification (UQ), where a large number of repeated simulations are required. Solving PDE systems with

* Corresponding author at: Department of Aerospace and Mechanical Engineering, University of Notre Dame, Notre Dame, IN, United States of America.
E-mail address: jwang33@nd.edu (J.-X. Wang).

https://doi.org/10.1016/j.jcp.2020.110079
0021-9991/

an optimal balance between accuracy and efficiency still remains a long-standing challenge. Recently, deep neural networks (DNNs) have been demonstrated to be promising for solving PDEs or metamodeling of PDE-based systems [1–6]. The advantages of using DNNs to approximate solutions of PDEs can be summarized as follows: (i) DNNs are capable to express any strong nonlinear relationships, mathematically supported by universal approximation theorems [7,8]; (ii) forward evaluations of trained DNNs are extremely efficient, which is a desirable feature for real-time or many-query applications; (iii) DNN models are analytically differentiable and thus derivative information can be easily extracted via automatic differentiation for optimization and control problems.

1.1. Physics-informed fully-connected neural networks

Generally, training of a DNN model requires a vast amount of labeled data, which are often unavailable in many scientific machine learning (SciML) applications [9–12]. However, when the governing PDEs are known, their solutions can be learned in a physics-constrained fashion with less data [2,13,14] or even without any data [3,4,15]. Namely, physics-informed loss functions are constructed based on PDE residuals and the DNN is trained by minimizing the violation of physical laws. The idea of using neural networks to solve differential equations has been proposed decades ago [16–18] and recently experienced a resurgence of interest due to remarkable advances in deep learning [2,19]. Raissi et al. [2] developed physics-informed neural networks (PINN) using modern deep learning techniques, where the PDE residual is incorporated into the loss function of fully-connected neural networks (FC-NN) as a regularizer, enabling training in "small data" regimes. This weakly-supervised method has been applied to solve various PDEs with limited training data in many scientific problems, including subsurface flows [20], vortex-induced vibrations [21], turbulent flows [22,23], cardiovascular systems [14,24–26], metamaterial design [27–29], geostatistical modeling [30], and others [31–33]. This idea has also been extended to utilize multi-fidelity datasets [34] and solve system identification problems [35–37]. In order to handle sparse, noisy data and quantify aleatoric uncertainty arising from measurement noise, a Bayesian formulation of the physics-constrained learning was proposed by Sun and Wang [14] using variational inference (VI), and a more comprehensive study on Bayesian PINN was conducted by Yang et al. [38], where both the VI and Hamiltonian Monte Carlo formulations are compared. The total uncertainty quantification (UQ) in PINN was also investigated based on adversarial inference [39] and arbitrary polynomial chaos in conjunction with dropout [40]. Although a moderate amount of training data are needed in the aforementioned works, the requirement of data can be entirely avoided if initial/boundary conditions (IC/BC) are properly enforced. Namely, the PDEs with specified IC/BCs can be solved deterministically within a deep learning framework. The effectiveness of the data-free FC-NN based PDE solution algorithm has been demonstrated on a number of canonical PDEs [41,42] and stochastic PDEs [1,43,44]. To improve the learning performance, researchers have recently explored several different directions, e.g., strong/variational formulations of PDE residuals [45–48], distributed learning using domain decomposition [49–51], and convergence analysis in network training/optimizations [41,52,53].

1.2. Physics-informed convolutional neural networks

Although physics-informed FC-NN for deterministically solving PDEs has been extensively investigated, the solution predictions for parametric PDEs in parameterized spaces (e.g., IC/BCs, geometry, and equation parameters) are less explored. The latter is critical for developing efficient metamodels (i.e., surrogate models). Only a very few existing studies aimed to build surrogate models using physics-informed FC-NN. Nabian and Meidani [54] and Karumuri et al. [55] applied the physics-informed FC-NN for efficient uncertainty propagation in steady heat equations. Sun et al. [4] developed a physics-constrained DNN surrogate for fluid flows with varying geometries and fluid properties, which was the first attempt of using FC-NNs to learn the solutions of parametric Navier-Stokes equations without relying on any labeled data. Despite some success, the FC-NN formulation suffers from scalability issues. This is because the training cost of FC-NNs will significantly increase for complex problems especially with parametric variations since the PDE residuals need to be evaluated on massive amounts of collocation points in high-dimensional input spaces. To enable efficient learning of large-scale spatiotemporal solution fields, the convolutional neural network (CNN) structure has attracted increasing attention in the SciML community. Compared to FC-NNs, CNNs usually need orders of magnitude fewer parameters because of parameter sharing via filter-based convolution operations, which is thus well-positioned for large-scale and high-dimensional problems [3,56].

Most recently, researchers started to blend physics with CNN-based learning, such as imposing physical constraints into CNNs [57–63], development of physics-informed CNN for surrogate modeling [3,6,15,64,65], and discovering underlying governing equations from observed data within CNN-based architectures [66–69]. Kim et al. [57] presented a generative CNN model to synthesize fluid simulations with the strictly imposed divergence-free condition using stream functions, and a similar idea was investigated by Mohan et al. [61] for coarse-graining of 3D turbulent flows, where boundary conditions were also strictly enforced on a regular domain. The physical constraints can also be imposed in a soft manner as penalty terms by modifying loss functions, and this idea has been exploited to enable weakly-supervised learning (i.e., using less labeled data) [58] or physically-correct synthetic turbulence generation and turbulence enrichment [60,62]. In terms of surrogate modeling, Zhu et al. [3] devised a physics-constrained convolutional encoder-decoder structure with a conditional-based generative model to learn solutions of high-dimensional elliptic PDEs without using any labeled data, and this approach has recently been extended for learning dynamical PDEs with parameterized initial conditions [15]. In a similar vein, Joshi

et al. [65] generated solutions of parametric Burgers equations by CNNs using adversarial learning. As for equation discovery, Long et al. [66,67] utilized deep CNNs in combination with symbolic networks to discover PDEs from spatiotemporal data by interpreting the learned filters. Rackauckas et al. [69] developed universal differential equations (UDEs) that leverage CNN to discover unknown equations from data. Singh et al. [68] presented a low-weight interpretable convolutional encoder-decoder network to capture the invariant structure of observation data for various PDE systems.

1.3. Scope and contributions of present work

Despite showing great promise, all the existing studies on physics-informed CNNs are only able to deal with problems defined on *regular (rectangular) domains with uniform grids*, which largely limits their applications to general scientific problems, where geometries are often *complex and irregular*. The underlying reason is that classic CNNs and their convolution operations are originally designed for processing natural images, which are described as functions in Euclidean space, sampled on a uniform mesh. However, the coordinate frames for problems with irregular domains have non-Euclidean structures, where the shift invariance that justifies the use of classic convolutional filters is no longer valid. Particularly in physics-constrained learning, the derivative terms in PDE-informed loss are computed based on finite difference through convolution operations, which only works in image-like rectangular domains. To handle data with non-Euclidean structures, in the Artificial Intelligence (AI) community, there recently has been growing interest in geometric deep learning, which is an umbrella term for emerging techniques aiming to adapt CNN to non-Euclidean space. Graph theory, spectral transformation, and manifold embedding, etc., have been utilized to reformulate the non-Euclidean convolution operations. Nonetheless, many of these newly proposed geometric CNNs have difficulties generalizing across different topologies, and moreover, it is not clear how to construct PDE-based loss function and impose boundary conditions for physics-constrained learning.

In this work, we propose a *novel* physics-constrained deep learning method, named as the physics-informed geometry-adaptive convolutional neural network (PhyGeoNet), *aiming to learn solutions of parametric PDEs on irregular domains without using any labeled data* (no simulations data are needed for training). Specifically, we use elliptic coordinate transformations to adapt CNN-based learning for problems with irregular geometries. Unlike the graph/geodesic CNNs, where problem-specific convolutional filters have to be designed, our proposed method can directly leverage powerful state-of-the-art uniform Cartesian-grid-based CNN architectures. Moreover, PDEs on the reference domain will be used to construct physics-informed loss function via finite-difference convolution kernels and boundary conditions are strictly encoded into networks based on padding operations. The *novel contributions* of this paper are as follows: (a) we propose a physics-informed CNN architecture, enabling data-free learning for parametric PDEs with irregular geometries; (b) encode boundary conditions into the CNN architecture in a hard manner (i.e., BCs are strictly enforced [4,61] for irregular geometries); (c) demonstrate the effectiveness of the proposed method on parametric heat equations and Navier-Stokes equations; (d) compare the proposed method with physics-informed FC-NNs (i.e., PINN) in terms of accuracy and efficiency. Moreover, to the best of authors' knowledge, this is the first attempt of using CNN to learn parametric Navier–Stokes equations on complex geometries without relying on any labeled data for training. The rest of the paper is organized as follows. The framework of proposed PhyGeoNet is introduced in Section 2. Numerical results on heat equations and Navier-Stokes equations in both non-parametric and parametric settings are presented in Section 3. The performance comparison between PhyGeoNet and PINN is discussed in Section 4. Finally, Section 5 concludes the paper.

2. Methodology

Physics-constrained learning is formulated by constructing a PDE-based loss function, where PDE residuals with the neural network (NN)-approximated solutions are computed. In CNN architecture, the derivative terms are obtained by finite difference using spatially-invariant convolution operations, which fails on irregular domains with non-uniform grids. In this section, we first provide a mathematical overview of physics-constrained learning using classic CNN and discuss its limitations for complex geometries. Then, we introduce a novel learning framework of the geometry-adaptive CNN, PhyGeoNet, where elliptic coordinate transformation is encoded to handle non-uniform grids and irregular geometries. The mathematical concept, network architecture, and implementation details are discussed.

2.1. Physics-constrained learning with classic convolutional neural network

2.1.1. Learning for PDE solutions on uniform grids without labels

The goal of this work is to learn the steady-state solutions of PDEs in a parametric setting. Consider a system of parametric steady PDEs,

$$\begin{aligned} \mathcal{F}(\mathbf{u}, \nabla\mathbf{u}, \nabla^2\mathbf{u}, \cdots; \boldsymbol{\mu}) &= 0, \quad \text{in } \Omega_p, \\ \mathcal{B}(\mathbf{u}, \nabla\mathbf{u}, \nabla^2\mathbf{u}, \cdots; \boldsymbol{\mu}) &= 0, \quad \text{on } \partial\Omega_p, \end{aligned} \tag{1}$$

where $\mathcal{F}(\cdot)$ represents PDE operators, defined on the physical domain Ω_p and parameterized by $\boldsymbol{\mu}(\mathbf{x})$; The solution variables of the PDE system are denoted by $\mathbf{u} = \mathbf{u}(\mathbf{x})$, which is a scalar or vector function of spatial coordinates $\mathbf{x} \in \Omega_p$; ∇ is the gradient operator with respect to $\mathbf{x}$; $\mathcal{B}(\cdot)$ represents functions for boundary conditions (BCs), which are enforced on the

boundary $\partial\Omega_p$ of the physical domain. When a set of parameters $\boldsymbol{\mu}$ is given, the PDE system (Eq. (1)) can be solved numerically using FD, FV, or FE methods, which is usually time-consuming. However, such a process has to be performed each time $\boldsymbol{\mu}$ is changed, posing great challenges on UQ/optimization applications. To enable rapid forward propagation of coordinates/parameters $[\mathbf{x}, \boldsymbol{\mu}]$ to state variables $\mathbf{u}(\mathbf{x}; \boldsymbol{\mu})$, neural networks are introduced to learn the parameter-to-state map. The learning process can be done completely offline and the online inference for new parameters will be very efficient via the trained network. In most existing works, FC-NN architectures were employed to learn the mapping in a pointwise fashion, which, however, introduces considerable training burden for large-scale problems. Unlike FC-NNs, CNNs enable image-based end-to-end learning and can directly generate solution fields over the entire domain instead of pointwise outputs. Hence, they often have much less training costs and better predictive accuracy [3]. Specifically, the discrete solution field $\mathbf{u}(\boldsymbol{\chi}, \boldsymbol{\mu}(\boldsymbol{\chi}))$ on a *rectangular domain* can be approximated by a CNN model,

$$\mathbf{u}\Big(\boldsymbol{\chi}, \boldsymbol{\mu}(\boldsymbol{\chi})\Big) \approx \mathbf{u}^{cnn}\Big(\boldsymbol{\chi}, \boldsymbol{\mu}(\boldsymbol{\chi}); \Gamma\Big), \tag{2}$$

where $\boldsymbol{\chi} = \{\mathbf{x}_1, \cdots, \mathbf{x}_{n_g}\}$ represents a set of n_g fixed grid points uniformly spaced (like pixels/voxels of images); $\Gamma = \{\gamma^l\}_{l=1}^{n_l}$ is a bank of trainable filters for convolution operations. The input layer of the CNN model includes discrete spatial coordinate $\boldsymbol{\chi}$ and parameter fields $\boldsymbol{\mu}(\boldsymbol{\chi})$, which can be formulated as multiple image channels. To obtain the output solutions, multiple convolutional (conv) layers are acted on the input channels by applying the bank of filters Γ and pointwise nonlinearity $\phi(\cdot)$ (i.e., activation functions). For instance, the output (i.e., feature map) $g^l(\mathbf{x})$ of the lth conv layer can be expressed as,

$$g^l(\mathbf{x}) = \phi\Big((g^l \odot \gamma^l)(\mathbf{x})\Big), \ \mathbf{x} \in \boldsymbol{\chi}^l, \tag{3}$$

where $\odot$ denotes convolution operation,

$$(g \odot \gamma)(\mathbf{x}) = \int_{\boldsymbol{\chi}} g(\mathbf{x} - \mathbf{x}')\gamma(\mathbf{x}')d\mathbf{x}'. \tag{4}$$

By convention, one can train the CNN by minimizing the cost function on the training set $\{\boldsymbol{\mu}_i, \mathbf{u}_i^d\}_{i=1}^{n_d}$,

$$\min_{\Gamma} \sum_{i=1}^{n_d} \underbrace{\Big\| \mathbf{u}^{cnn}(\boldsymbol{\chi}, \boldsymbol{\mu}_i; \Gamma) - \mathbf{u}_i^d(\boldsymbol{\chi}) \Big\|_{\Omega_p}}_{\text{data-based loss: } \mathcal{L}_{data}}, \tag{5}$$

where $\|\cdot\|_\Omega$ is L_2 norm over spatial domain Ω_p. Such purely data-based learning requires enormous (n_d) labeled data $\mathbf{u}_i^d, i = 1, \cdots, n_d$, which are usually too expensive to obtain from either numerical simulations or experiments. As an alternative, the training can be formulated as a constrained optimization using governing equations and corresponding BCs,

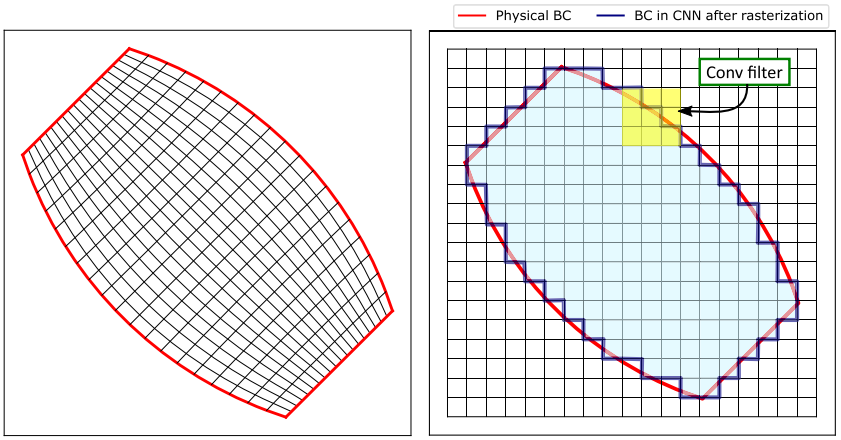
$$\begin{aligned} &\min_{\Gamma} \sum_{i=1}^{n_d} \underbrace{\Big\| \mathcal{F}\Big(\mathbf{u}^{cnn}(\boldsymbol{\chi}, \boldsymbol{\mu}_i; \Gamma), \nabla\mathbf{u}^{cnn}(\boldsymbol{\chi}, \boldsymbol{\mu}_i; \Gamma), \nabla^2\mathbf{u}^{cnn}(\boldsymbol{\chi}, \boldsymbol{\mu}_i; \Gamma), \cdots; \boldsymbol{\mu}_i\Big) \Big\|_{\Omega_p}}_{\text{equation-based loss: } \mathcal{L}_{pde}}, \\ &s.t. \ \mathcal{B}\Big(\mathbf{u}^{cnn}(\boldsymbol{\chi}, \boldsymbol{\mu}_i; \Gamma), \nabla\mathbf{u}^{cnn}(\boldsymbol{\chi}, \boldsymbol{\mu}_i; \Gamma), \nabla^2\mathbf{u}^{cnn}(\boldsymbol{\chi}, \boldsymbol{\mu}_i; \Gamma), \cdots; \boldsymbol{\mu}_i\Big) = 0, \ \text{ on } \partial\Omega_p. \end{aligned} \tag{6}$$

The spatial derivative terms (i.e., $\nabla\mathbf{u}^{cnn}, \nabla^2\mathbf{u}^{cnn}$) of the CNN ansatz are numerically approximated via convolution operation with predefined non-trainable filters, which are interpreted as discretized differential operators in the finite difference form [3,66,67]. Constraints of BCs can be either treated as penalty terms (soft BC enforcement) [2,3] or strictly encoded into the network architecture (i.e., hard BC enforcement) [4]. The optimization problem defined in Eq. (6) is solved based on stochastic gradient descent (SGD), e.g., Adam algorithm [70]. In the optimization, only residuals of the PDEs are evaluated without the need of solving them. Note that if a small amount of labeled data are available, the equation-based loss ($\mathcal{L}_{pde}$ in Eq. (6)) can be combined with the data-based loss ($\mathcal{L}_{data}$ in Eq. (5)), and then the combined loss function (A.K.A. physics-informed loss) will be minimized, as weakly-supervised learning. In this work, we only focus on purely PDE-driven learning without using any labeled data. Namely, neural networks are used to solve the PDEs in a parametric setting.

2.1.2. Limitations of classic CNN on irregular geometries with nonuniform grids

The classic CNNs had originally been developed for image recognition and processing, where the convolution filters are designed for images or image-like data. Namely, the discrete coordinates $\boldsymbol{\chi}$ in Eq. (2) have to be a set of uniform grid points within a rectangular domain. However, in scientific computing and physical modeling applications, the geometries are usually irregular with non-uniform grids (e.g., boundary-fitted mesh in Fig. 1a).

In such cases, classic CNN techniques are not directly applicable since the Euclidean-distance-based convolution filters are no longer invariant on non-uniform meshes. Although new convolution filters can be defined for a given boundary-fitted coordinate, how to generalize them across different domain shapes (e.g., in geometric parameterization) and how to



(a) Non-uniform grids (b) Rasterization

Fig. 1. Limitations of the classic CNN in solving PDEs on an irregular domain. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

construct differential operators for physics-based loss function are still challenging. To enable direct use of the classic CNN backbone, "rasterization" is often performed to preprocess the data in data-driven scenarios, converting irregular shapes into uniform-grid based 2D/3D images, where pixels/voxels are labeled by binaries [71–73] or Signed Distance Function (SDF) to represent geometries [74,75], as shown in Fig. 1b. Nonetheless, the binary or SDF based geometry representations fail to operate in physics-constrained learning due to the following issues and drawbacks:

1. After rasterization, the boundary shape becomes stepped and zigzag (Fig. 1), introducing big aliasing errors under low/moderate image resolution. Especially, when it comes to boundary-value problems, even a minor misrepresentation of the boundary may lead to large errors in the learned solution field.
2. Using binary/SDF representations, it is difficult or even impossible to impose PDE boundary conditions ($\mathcal{B}(\mathbf{u}^{cnn}) = 0$) and thus physics-constrained learning would fail, particularly when labeled data are scarce or even absent. This is because the PDE solutions are uniquely determined by given BCs, and without properly prescribed BCs the optimization problem is ill-posed.
3. When the shape of the physical domain (blue region in Fig. 1b) is far from its corresponding rectangular envelope, the artifacts introduced from the background region (blank region in Fig. 1b) may complicate the training process and make the optimization easier to trap in bad local minima. For instance, in binary-based representation, the derivatives computed in the background region are always zeros and thus the PDEs are exactly satisfied, which clearly are not the solutions of interest.
4. The mesh is always uniform and cannot be adjusted based on the physics. For example, in fluid dynamics, the mesh should be refined near the boundary to resolve the boundary layers.
5. Although a sufficiently high resolution can reduce representation errors and alleviate some of the aforementioned issues to a certain extent, it will significantly increase the training costs and cause memory issues.
6. Conventional rasterization/voxelization methods are not differentiable with respect to the input design parameters, limiting its use for geometry optimization and design purpose.

2.2. Physics-informed geometry-adaptive convolutional neural network

Motivated by the drawbacks and limitations of classic CNNs as discussed above, we propose a novel physics-informed geometry-adaptive CNN approach (PhyGeoNet), enabling CNN-based physics-informed learning for problems with non-uniform grids and irregular geometries. The key idea is to utilize coordinate transformation techniques to map solution fields from irregular physical domain to rectangular reference domain. Hence, the powerful classic uniform grid based CNN backbone can be directly leveraged by reformulating the physics-constrained optimization (Eq. (6)) on the reference domain. The geometry transformation is a deterministic process and can be precomputed, so it can be seen as a part of network architecture and no aliasing errors will be introduced. Moreover, various BCs are strictly enforced in a hard manner, which is in contrast to previous works, where the BCs are often treated as penalty terms and imposed softly.

2.2.1. Coordinate transformation between physical and reference domains

The forward/inverse mapping between coordinates of the irregular physical domain (Ω_p) and regular reference domain (Ω_r) can be defined as,

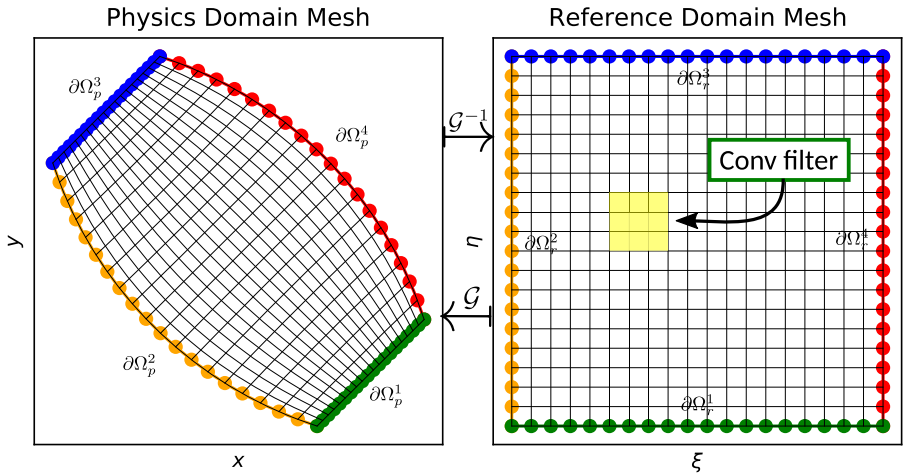


Fig. 2. A schematic diagram of the coordinate mappings between an irregular physical domain and regular reference domain. Classic convolution operations are only acted on the reference domain.

$$\mathbf{x} = \mathcal{G}(\boldsymbol{\xi}), \quad \boldsymbol{\xi} = \mathcal{G}^{-1}(\mathbf{x}), \tag{7}$$

where $\mathcal{G} : \Omega_r \mapsto \Omega_p$ denotes the forward map, and $\mathcal{G}^{-1} : \Omega_p \mapsto \Omega_r$ denotes the inverse map; $\mathbf{x} \in \Omega_p$ and $\boldsymbol{\xi} \in \Omega_r$ represent mesh coordinates of the physical domain and reference domain, respectively. Typically, non-uniform mesh grids are specified in the physical domain with arbitrary geometry, while the corresponding grids of the reference domain are uniformly spaced in a regular geometry (e.g., rectangular or cuboid). Given coordinate transformation functions ($\mathcal{G}/\mathcal{G}^{-1}$), the deterministic one-to-one mapping of coordinates from the reference domain to the physical domain can be uniquely determined and Jacobians of the transformation map are also available to reformulate the PDEs on the reference domain. However, in most cases, analytical forms of $\mathcal{G}$ or $\mathcal{G}^{-1}$ are not available, which have to be approximated numerically. Therefore, elliptic coordinate transformation [76] is applied here, and the general idea is to solve a boundary-value problem (i.e., elliptic equations) since the boundary conditions are given in both physical and reference domains.

Without loss of generality, two-dimensional (2-D) elliptic transformation is introduced, i.e., $\Omega_r, \Omega_p \subset \mathbb{R}^2$. Consider an irregular domain bounded by four edges $\partial\Omega_p^i, i = 1, \cdots, 4$ (Fig. 2a), and the corresponding reference domain is a rectangular bounded by edges $\partial\Omega_r^i, i = 1, \cdots, 4$ (Fig. 2b).

The coordinates of the physical domain and reference domains are denoted by $\mathbf{x} \doteq [x, y]$ and $\boldsymbol{\xi} \doteq [\xi, \eta]$, respectively. The one-to-one mapping on the boundary grids for physical and reference domain are usually known *a priori*, i.e.,

$$\boldsymbol{\xi}(\mathbf{x}) = \boldsymbol{\xi}_b \ \text{ for } \forall\, \mathbf{x} \in \partial\Omega_p^i, i = 1, \cdots, 4 \tag{8a}$$

$$\mathbf{x}(\boldsymbol{\xi}) = \mathbf{x}_b \ \text{ for } \forall\, \boldsymbol{\xi} \in \partial\Omega_r^i, i = 1, \cdots, 4 \tag{8b}$$

where $\boldsymbol{\xi}_b$ and $\mathbf{x}_b$ are given. To numerically approximate $\mathcal{G}$ or $\mathcal{G}^{-1}$, one can solve an elliptic equations with specified boundary conditions in Eq. (8). For example, the inverse map $\mathcal{G}^{-1}$ can be obtained by solving a diffusion equation,

$$\nabla^2 \boldsymbol{\xi}(\mathbf{x}) = 0, \tag{9}$$

with the BC defined in Eq. (8a). However, the inverse map $\mathcal{G}^{-1} : \Omega_p \mapsto \Omega_r$ is less useful since the set of physical grids $\mathbf{x}$ that can be converted to a set of uniform grids are usually unknown. Instead, it is practical to solve the forward map $\mathcal{G} : \Omega_r \mapsto \Omega_p$, which locates corresponding collocation points in physical domain given uniform mesh grids in the reference domain. By interchanging the independent and dependent variables in Eq. (9), the following diffusion equations in terms of physical coordinate $\mathbf{x}$ is derived,

$$\alpha \frac{\partial^2 x}{\partial \xi^2} - 2\beta \frac{\partial^2 x}{\partial \xi \partial \eta} + \gamma \frac{\partial^2 x}{\partial \eta^2} = 0, \tag{10a}$$

$$\alpha \frac{\partial^2 y}{\partial \xi^2} - 2\beta \frac{\partial^2 y}{\partial \xi \partial \eta} + \gamma \frac{\partial^2 y}{\partial \eta^2} = 0, \tag{10b}$$

where α, β, and γ are given by,

$$\alpha = \left(\frac{\partial x}{\partial \eta}\right)^2 + \left(\frac{\partial y}{\partial \eta}\right)^2,$$
$$\gamma = \left(\frac{\partial x}{\partial \xi}\right)^2 + \left(\frac{\partial y}{\partial \xi}\right)^2, \tag{11}$$
$$\beta = \frac{\partial x}{\partial \xi}\frac{\partial x}{\partial \eta} + \frac{\partial y}{\partial \xi}\frac{\partial y}{\partial \eta}.$$

By solving Eq. (10) with BC defined in Eq. (8b) numerically (i.e., iterative method), the discrete values of the forward map $\mathcal{G}$ are obtained. The proof that Eq. (10) holds given Eq. (9) are provided in Appendix A.

2.2.2. Reformulate physics-constrained learning on reference domain

To enable physics-constrained learning in the reference domain, the original PDEs defined in the physical domain (Eq. (1)) has to be recast as the form in the reference domain. Specifically, Jacobians of the transformation map $\mathcal{G}$ are computed in order to convert differential operators from the physical domain to the reference domain. For instance, the first derivatives in Ω_p is transformed into Ω_r as,

$$\frac{\partial}{\partial x} = \left(\frac{\partial}{\partial \xi}\right)\left(\frac{\partial \xi}{\partial x}\right) + \left(\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \eta}{\partial x}\right), \tag{12a}$$
$$\frac{\partial}{\partial y} = \left(\frac{\partial}{\partial \xi}\right)\left(\frac{\partial \xi}{\partial y}\right) + \left(\frac{\partial}{\partial \eta}\right)\left(\frac{\partial \eta}{\partial y}\right). \tag{12b}$$

Typically, finite differences are used to numerically calculate Jacobians, which has to be performed in the reference domain. Therefore, the inverse transformation is applied [77] to modify Eq. (12) as,

$$\frac{\partial}{\partial x} = \underbrace{\frac{1}{J}}_{\text{constant}}\left[\left(\frac{\partial}{\partial \xi}\right)\underbrace{\left(\frac{\partial y}{\partial \eta}\right)}_{\text{constant}} - \left(\frac{\partial}{\partial \eta}\right)\underbrace{\left(\frac{\partial y}{\partial \xi}\right)}_{\text{constant}}\right], \tag{13a}$$
$$\frac{\partial}{\partial y} = \underbrace{\frac{1}{J}}_{\text{constant}}\left[\left(\frac{\partial}{\partial \eta}\right)\underbrace{\left(\frac{\partial x}{\partial \xi}\right)}_{\text{constant}} - \left(\frac{\partial}{\partial \xi}\right)\underbrace{\left(\frac{\partial x}{\partial \eta}\right)}_{\text{constant}}\right], \tag{13b}$$

where $J = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi} \neq 0$ is the determinant of the Jacobian matrix and metrics $\frac{\partial y}{\partial \eta}$, $\frac{\partial y}{\partial \xi}$, $\frac{\partial x}{\partial \eta}$, and $\frac{\partial x}{\partial \xi}$ can be precomputed and remain constant if $\mathcal{G}$ is defined. Therefore, differential operators $\frac{\partial}{\partial \eta}$ and $\frac{\partial}{\partial \xi}$ are defined on the reference domain, which are implemented as classic CNN filters defined by finite difference (FD) stencils. To obtain the second or higher derivatives (e.g., $\frac{\partial^2}{\partial x^2}$ and $\frac{\partial^2}{\partial x y}$), the FD-based CNN filters (Eq. (13)) are applied to the solution field for multiple times, instead of using chain rule (Eq. (12)), which overcomplicates the implementation. More details can be found in Appendix B.

With these modified derivative terms, the optimization defined in Eq. (6) is reformulated on the reference domain as,

$$\begin{aligned} &\min_{\Gamma} \sum_{i=1}^{n_d} \underbrace{\left\| \tilde{\mathcal{F}}\left(\mathbf{u}^{cnn}(\boldsymbol{\Xi}, \boldsymbol{\mu}_i; \Gamma), \nabla \mathbf{u}^{cnn}(\boldsymbol{\Xi}, \boldsymbol{\mu}_i; \Gamma), \nabla^2 \mathbf{u}^{cnn}(\boldsymbol{\Xi}, \boldsymbol{\mu}_i; \Gamma), \cdots; \boldsymbol{\mu}_i\right)\right\|_{\Omega_r}}_{\text{equation-based loss on reference domain: } \tilde{\mathcal{L}}_{pde}}, \\ &s.t.\ \tilde{\mathcal{B}}\left(\mathbf{u}^{cnn}(\boldsymbol{\Xi}, \boldsymbol{\mu}_i; \Gamma), \nabla \mathbf{u}^{cnn}(\boldsymbol{\Xi}, \boldsymbol{\mu}_i; \Gamma), \nabla^2 \mathbf{u}^{cnn}(\boldsymbol{\Xi}, \boldsymbol{\mu}_i; \Gamma), \cdots; \boldsymbol{\mu}_i\right) = 0, \ \text{ on } \partial\Omega_r, \end{aligned} \tag{14}$$

where $\tilde{\mathcal{F}}$ and $\tilde{\mathcal{B}}$ are modified PDE and boundary operators on reference domain, and $\boldsymbol{\Xi}$ are discrete reference coordinates.

2.2.3. Hard enforcement of boundary conditions

The solutions are uniquely determined by given BCs, and thus proper enforcement of BCs is essential when labeled data are absent, otherwise, the optimization problem becomes ill-posed. In general, the constrained optimization defined in Eq. (14) can be converted to an unconstrained one by treating BCs as penalty terms, known as the *soft BC enforcement*. As an alternative, the constraints can be strictly enforced and BCs will be automatically satisfied by construction, known as the *hard BC enforcement*.

As discussed by Sun et al. [4], the soft BC enforcement has several drawbacks and slows the convergence. In this work, a hard BC enforcement is proposed, where both Dirichlet and Neumann BCs are strictly satisfied at the discrete level based on padding, referring to layers of pixels added to images when it is being processed by the CNN filters. As shown in Fig. 3 (left), Dirichlet BCs can be exactly imposed by applying constant padding, which does not vary during training. As for the Neumann BCs, the values of padding are derived from the solutions at internal nodes via finite differences, Although the padding values vary at each training epoch, the difference relations defined by Neumann BCs are strictly enforced.

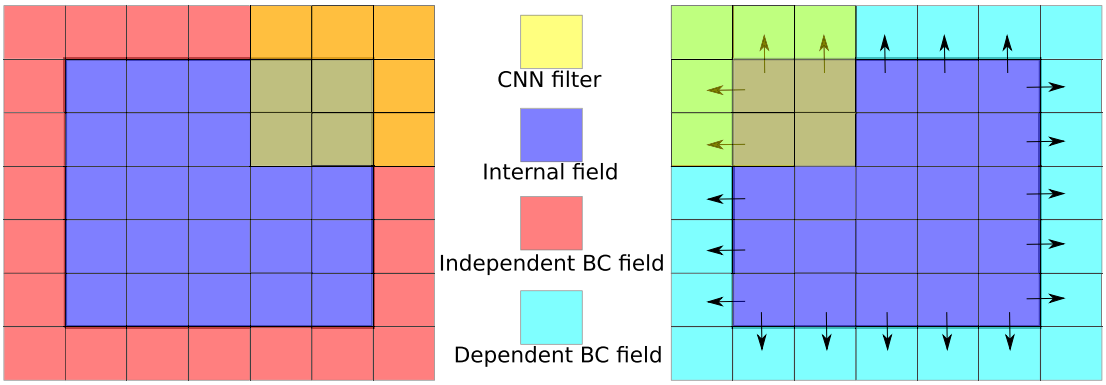


Fig. 3. Hard enforcement of BCs via padding. (left) Dirichlet BCs are strictly imposed by constant padding, which are fixed during training, and (right) Neumann BCs are imposed by padding that are derived from the internal field based on finite differences.

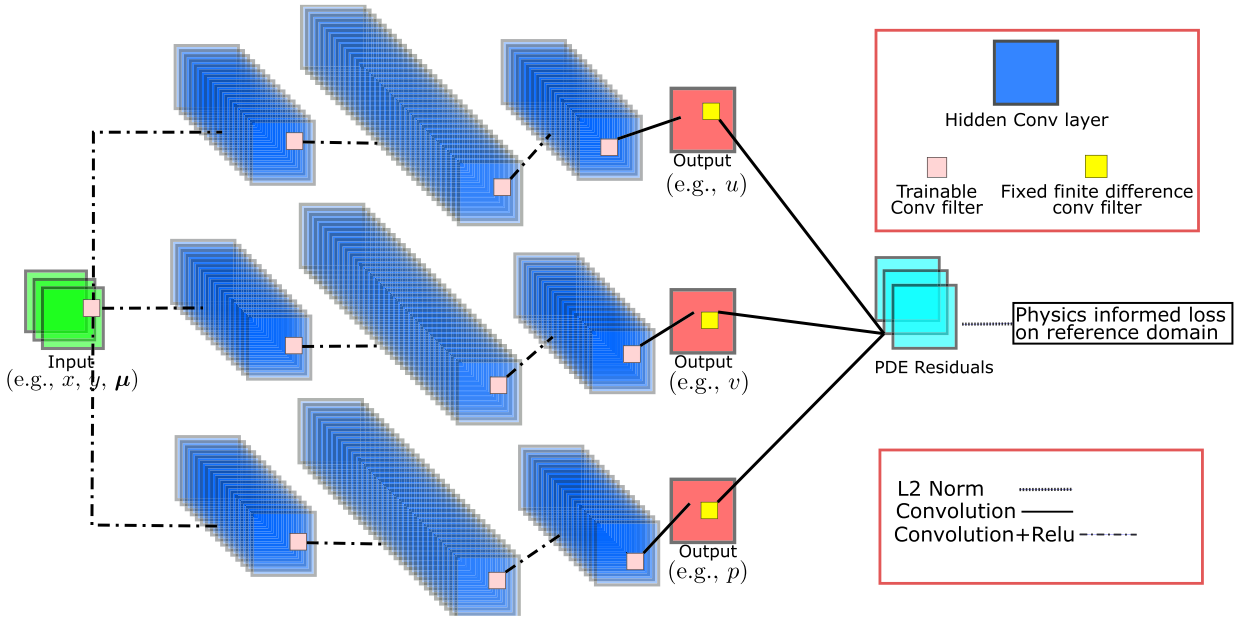


Fig. 4. The CNN architecture in reference domain, where forward propagation is from left to right. The solution field of each physical variable is described by a separate subnet.

2.2.4. Physics-informed CNN architecture in reference domain

The network architecture is built with input channels composed of physical parameters $\boldsymbol{\mu}$ and coordinates $\mathbf{x}$ of the physical domain.

The solution field of each state variable (e.g., pressure or velocity components) on the reference domain is represented by a separate sub-CNN, and thus the trainable network parameters (i.e., filters) are decoupled for different solution variables. This is because the values of different variables might be different in orders of magnitude (e.g., streamwise velocity component can be orders of magnitude greater than spanwise velocity component). The superiority of using separate subnets for multivariate regression has been demonstrated in both data-driven and data-free scenarios [4,78]. Each sub-net has an identical structure of three hidden convolution layers, which is adapted from a classic CNN architecture proposed by Shi et al. [79] originally for single image super-resolution. Specifically, each layer has a trainable filter with the same kernel size of 5×5, and 2-D conv operations with padding of 2 and stride of 1 are applied to ensure the output layer has the same size (height $\times$ width) as the input. All the sub-nets are trained simultaneously with a unified physics-based loss function constructed by PDE residual layers using fixed finite difference filters. The trainable network parameters are initialized from a uniform distribution of $\mathcal{U}\left(-\sqrt{\frac{1}{25C_{in}}}, \sqrt{\frac{1}{25C_{in}}}\right)$, where C_{in} is number of input channels. Fig. 4 gives an overview of the proposed network architecture.

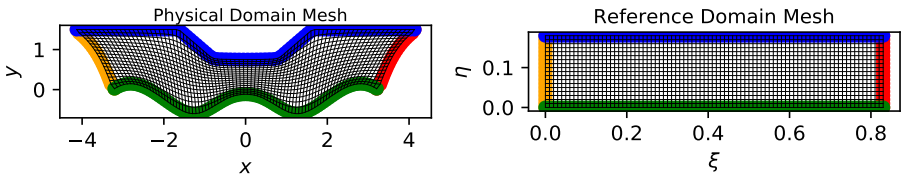


Fig. 5. The physical domain (left) and reference domain (right) for non-parametric heat equation. The corresponding edges of two domains are indicated by the same color.

3. Numerical results

In this section, several elliptic and parabolic PDE systems are studied to demonstrate the merit of the proposed PhyGeoNet for label-free learning on irregular geometries. To begin with, we first present PhyGeoNet solutions of the non-parametric heat equation and Navier-Stokes equations on irregular domains with arbitrary shapes. Then, the experiments on more challenging cases with varying parameters are explored. Specifically, the solutions of the heat equation with parameterized Dirichlet boundary conditions, Navier-Stokes equations with varying geometries, and Poisson equations with spatially-varying source term are learned without any data. To evaluate the learning/prediction performance, finite volume (FV)-based numerical simulations are conducted as the benchmark for comparisons, and the relative error metric is defined as,

$$\text{Error} = \sqrt{\frac{||\tilde{\mathbf{u}} - \mathbf{u}^{cnn}||_{L^2}}{||\tilde{\mathbf{u}}||_{L^2}}}, \tag{15}$$

where $\tilde{\mathbf{u}}$ is the ground truth (FV solution), $\mathbf{u}$ is the PhyGeoNet predictions, and $||\cdot||_{L^2}$ represents the L2 norm. The FV simulations are performed on an open-source FV platform, OpenFOAM [80]. In particular, the convection term is discretized based on Gauss theorem with second-order bounded linear upwind interpolation (i.e., Gauss linearUpwind Scheme in OpenFOAM), and the diffusion term is discretized using central Gauss linear interpolation with the explicit non-orthogonal correction scheme for the surface normal gradient (i.e., Gauss linear corrected). The Semi-Implicit Method for Pressure Linked Equations (SIMPLE) algorithms were used for solving the incompressible Navier-Stokes equations [81], and the Rhie and Chow interpolation with collocated grids was employed to prevent the pressure–velocity decoupling [82]. The proposed PhyGeoNet is implemented in PyTorch [83] and the training is conducted on an NVIDIA GeForce RTX 2080 Graphics Processing Unit (GPU) card. The computational parameter settings for all cases are summarized in Table C.4 (see Appendix C). The code and datasets for this work will become available at https://github.com/Jianxun-Wang/phygeonet upon publication.

3.1. Non-parametric solutions

3.1.1. Heat equation (Case 1)

Prediction of temperature distribution across an irregular surface requires to solve the heat equation Ω_p, which is of importance to industrial design of, e.g., chip cooling and panel heating [84,85]. Here we consider the heat equation defined as,

$$\begin{aligned} &\nabla \cdot (\nabla T(\mathbf{x})) = 0, \ \mathbf{x} \in \Omega_p, \\ &T(\mathbf{x}) = T_{bc}(\mathbf{x}), \ \mathbf{x} \in \partial\Omega_p, \end{aligned} \tag{16}$$

where T is temperature field and Dirichlet BC T_{bc} is given. The physical domain Ω_p has an irregular shape with boundary-fitted mesh as shown in Fig. 5 (left panel), for which PDE-constrained learning is unable to be formulated using classic CNNs [3,15,67,86]. In our proposed PhyGeoNet, the PDE-based loss function and network optimization are defined in the regular reference domain (right panel of Fig. 5), which is mapped from the irregular physical domain using elliptic transformation, and thus the solution field can be learned without any labeled data. Fig. 6 shows the comparison between the PhyGeoNet-predicted T field and FV-based solution (benchmark). It can be seen that the temperature on the upper boundary (blue) is set as $T = 0$, while on all other sides (green, red, and orange) $T = 1$. Compared to the FV benchmark, the heat distribution can be accurately captured by the PhyGeoNet with the relative prediction error of 0.098, demonstrating that the non-parametric heat equation can be solved by PhyGeoNet on irregular domains. The learning curve for this case is plotted in Fig. D.26a.

3.1.2. Navier-Stokes equations (Case 2)

There has been growing interests in using CNNs to simulate/model fluid flows governed by Navier-Stokes equations, but most of them are formulated in a data-intensive supervised fashion [57,74,87]. Even so, the representation of irregular geometries is challenging and has often to rely on pixel/voxel-based approaches using signed distance functions or descriptor distance functions, which may introduce considerable aliasing errors. Here, our proposed PhyGeoNet is trained in an unsupervised manner to learn the solutions of the steady incompressible Navier-Stokes equations,

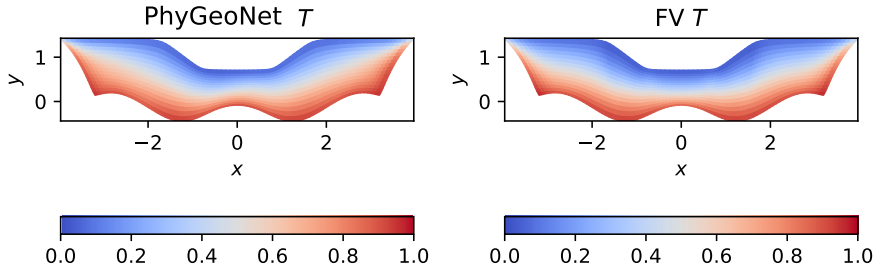


Fig. 6. Comparison of temperature fields obtained by PhyGeoNet and FV-based simulations.

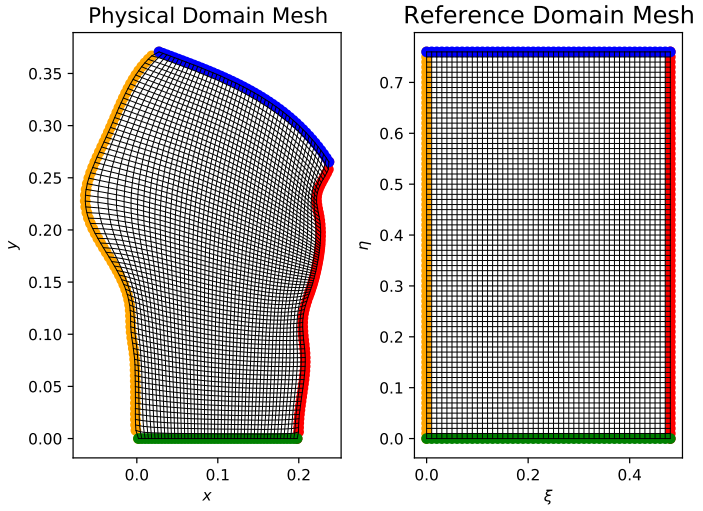


Fig. 7. The physical domain (left) and reference domain (right) for non-parametric Navier-Stokes equations. The corresponding boundary edges of two domains are indicated by the same color.

$$\nabla \cdot \mathbf{v} = 0, \tag{17a}$$

$$(\mathbf{v} \cdot \nabla)\mathbf{v} = \nabla \cdot (\nu \nabla \mathbf{v}) - \nabla p, \tag{17b}$$

where $\mathbf{v}$ is velocity vector, p is pressure, and ν is fluid viscosity. The fluid flow is solved on an irregular domain with a 2-D vascular geometry, as shown in Fig. 7. The boundary condition on the lower edge (green) is set as the inlet with a constant inflow $\mathbf{v} = [0, 1]$. The no-slip wall boundary condition is prescribed on the left and right boundaries (orange and red). The outlet on the upper boundary (blue) is given by $\nabla \mathbf{v} \cdot \mathbf{n} = 0$ and $p = 0$, where $\mathbf{n}$ is the local wall-normal vector. The input to the PhyGeoNet only includes geometry (i.e., coordinates) in the physical domain.

The PhyGeoNet predicted solutions are shown in Fig. 8, where the corresponding FV-based CFD results are plotted for comparison. From the inlet, the fluid flow starts to move up towards the outlet. The boundary layers are gradually developed on both sides and velocity magnitude decreases close to the wall due to the friction between fluid and walls. Because of the conservation of mass, the magnitude of the center velocity is greater than that of the inlet velocity. As for the pressure, both left and right corners are relatively high-pressure regions, and the pressure is dropping down as the flow approaches the outlet due to the conservation of momentum. The curvature of the left wall located at the upper left of the domain bends the flow to the right slightly, which also leads to a relatively high-pressure region. Overall, PhyGeoNet is able to capture the flow field reasonably well compared to the CFD benchmark for different Reynolds numbers in laminar regime. The relative errors for velocity magnitude and pressure are listed in Table 1, which provides a quantitative assessment of the performance. We can see that the solution error slightly grows as the Reynolds number (Re) increases, which is because the convection becomes more dominant, posing challenges on PDE-driven learning based on central difference filters. Special numerical treatment, e.g., upwind differencing, may improve the results in this regard. The learning curve (history of equation residuals) at Re = 20 is given in Fig. D.26b.

3.2. Parametric solutions

The real power of DNN-based approaches lies in learning PDE solutions in a parametric setting, since the trained DNN can be used as a surrogate model for rapid many-query applications. To be specific, the PhyGeoNet will take the PDE parameters $\boldsymbol{\mu}$ as a part of the input and output the corresponding solution $\mathbf{u}(\boldsymbol{\mu})$, approximating the parameter-to-solution

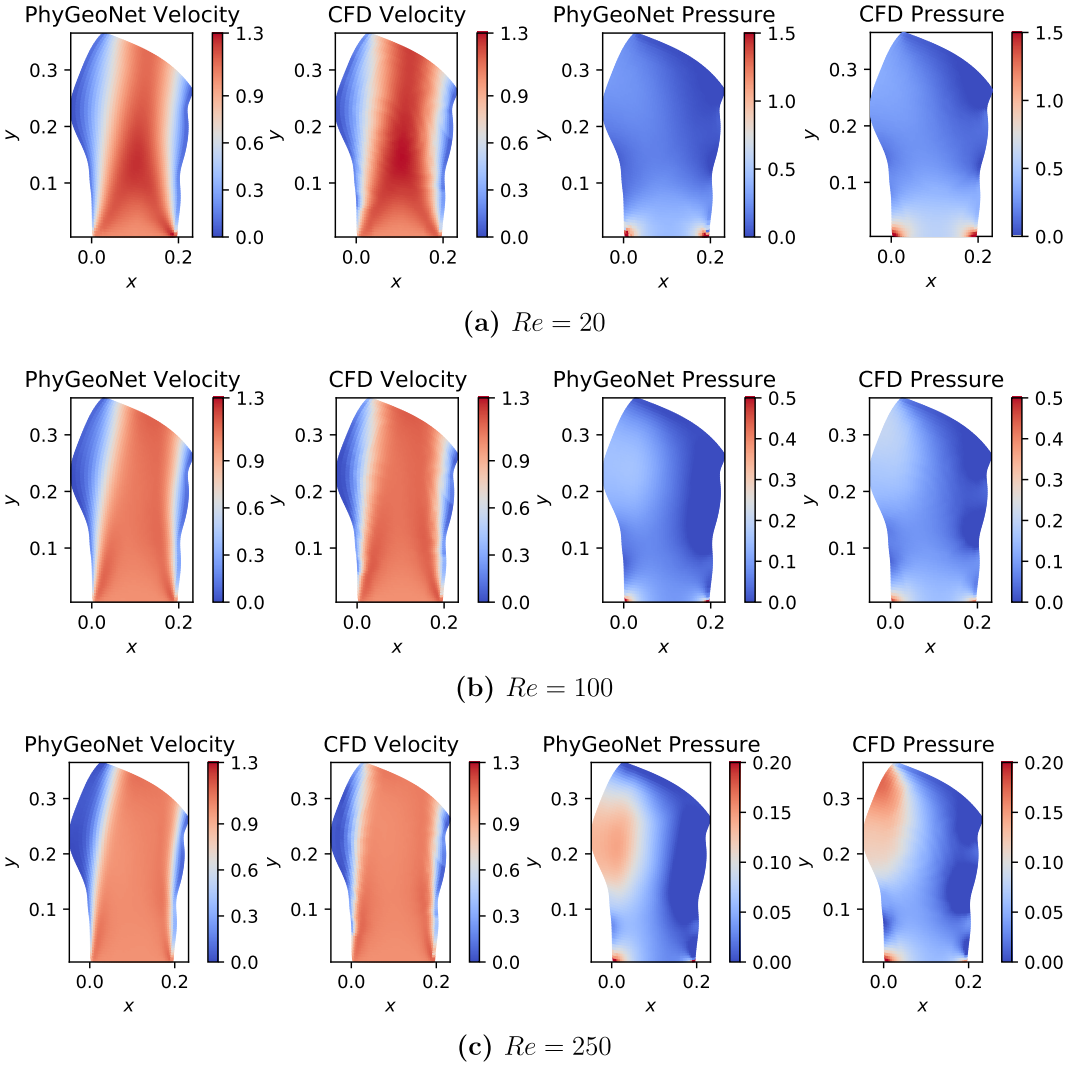


Fig. 8. The comparison between PheGeoNet and CFD solutions of velocity (left column) and pressure (right column).

Table 1
Relative error of learned solution for non-parametric Navier-Stokes equations.

Variable	Velocity magnitude	Pressure
Relative error ($Re = 20$)	0.0783	0.347
Relative error ($Re = 100$)	0.0858	0.340
Relative error ($Re = 250$)	0.1150	0.410

mapping $f : \boldsymbol{\mu} \mapsto \mathbf{u}$. To learn this mapping, the network will be trained on randomly selected parameter points $\{\boldsymbol{\mu}\}_{train}$ by minimizing the equation residuals as defined in Eq. (14), where no labels are required. Therefore, the training set $\{\boldsymbol{\mu}\}_{train}$ only includes input parameter points instead of output labels, which is in contrast to traditional data-driven DL models. After being fully trained, the PhyGeoNet can take new parameters $\{\boldsymbol{\mu}\}_{test}$ to rapidly predict the corresponding solutions nearly without additional cost. In this subsection, we will demonstrate this capability of the PhyGeoNet for learning and predicting solutions of PDEs with varying parameters/boundary conditions.

3.2.1. Heat equation with parameterized boundary conditions (Case 3)

In this subsection, the PhyGeoNet is trained to learn the parametric solutions of the heat equation, where the boundary condition is varying. A more complex irregular domain of a different topology is investigated here. Unlike the two geometries studied above, which both belong to the simply-connected domain, the annulus shape considered here (Fig. 9) is a doubly-connected domain [88,89], which can be seen a Riemann surface,

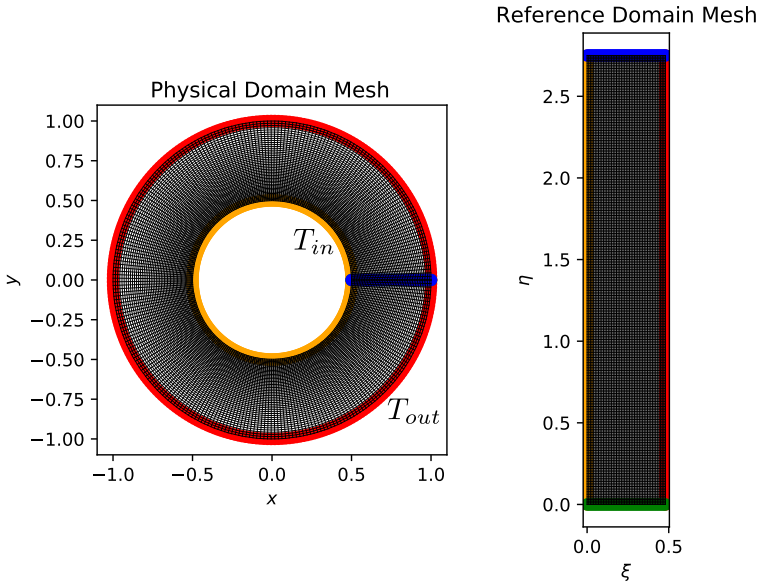


Fig. 9. The physical domain (left) and reference domain (right) for parametric heat equation. The corresponding boundary edges of two domains are indicated by the same color. Periodic boundary conditions are imposed on green and blue edges.

$$r < \left|\mathbf{x} - \mathbf{c}\right| < R, \tag{18}$$

where $\mathbf{c}$ is the center of the annulus and r and R are inner and outer radii, respectively. To solve this problem, the doubly-connected domain is transformed back to a simply-connected domain by cutting off the annulus and imposing periodic boundary conditions at where it is cut (see Fig. 9). The boundary temperature on the inner circle is parameterized as a varying constant of $T_{in} \in [1, 7]$, while for outer circle boundary, a fixed temperature of $T_{out} = 0$ is imposed. Now, the PhyGeoNet is built as a surrogate, whose input should reflect variations of boundary conditions. In contrast to the non-parametric cases, where only the domain coordinates are provided to the network as the input, the PhyGeoNet for parametric cases should include both the geometry information (i.e., coordinates) and variable parameters. In this case, the field of a linear interpolation from the inner boundary T_{in} to the outer one T_{out} is provided as the input to the PhyGeoNet, which uniquely represents the variable boundary conditions.

Unlike data-driven learning algorithms that are constrained by data availability, the PhyGeoNet can be trained at any input points (i.e., T_{in} here) due to its label-free nature. Even so, it is still interesting to investigate the prediction performance at unseen input points, where PDE residuals have not been minimized. Therefore, the PhyGeoNet is trained only on two input samples ($T_{in} = 1$ and 7) in an unsupervised fashion and evaluated at seven parameter points across $T_{in} = [1, 7]$. The learning curve, i.e., convergence history of the PDE-based loss, is plotted in Fig. D.26c. Fig. 10 shows the PhyGeoNet predictions compared with the FV-simulated benchmarks. The high temperature from the inner circle boundary is gradually diffused to zero towards the outer circle boundary. For all training and (unseen) test inner temperatures (T_{in}), the PhyGeoNet-predicted temperature contours agree with the numerical benchmarks from FV-based simulations very well. The relative errors of predictions at different parameters are lower than 0.05, as shown in Fig. 11. This demonstrates the potential of using PhyGeoNet as a surrogate for fast forward propagation of unseen input scenarios by leveraging strong expressibility and universal approximation capability of neural networks.

3.2.2. Navier-Stokes equations with varying geometries (Case 4)

In the second numerical experiment, the proposed PhyGeoNet is applied for surrogate modeling of laminar flows with varying geometries, which is significant in many different applications. For example, in the context of image-based cardiovascular flow modeling, medical images have started to be widely utilized to construct geometry models for downstream CFD simulations, which, however, usually introduces large uncertainties due to image noise, segmentation errors, and other artifacts. To quantify and propagate geometric uncertainty in simulated quantities of interests (QoIs) of the blood flow solutions are critical but often require a massive number of forward CFD simulations [90]. Although various surrogate models are proposed to reduce the computational burden [91,92], a considerable amount of labeled training data are still required. Here, the PhyGeoNet is employed to tackle this challenge without using any labels (i.e., CFD simulation data). Specifically, the steady incompressible Navier-Stokes equations (Eq. (17)) with parameterized vascular geometries are solved, and the trained CNN is able to rapidly predict flow velocity and pressure fields for new geometries. Here idealized geometries of vascular stenosis and aneurysm are parameterized as,

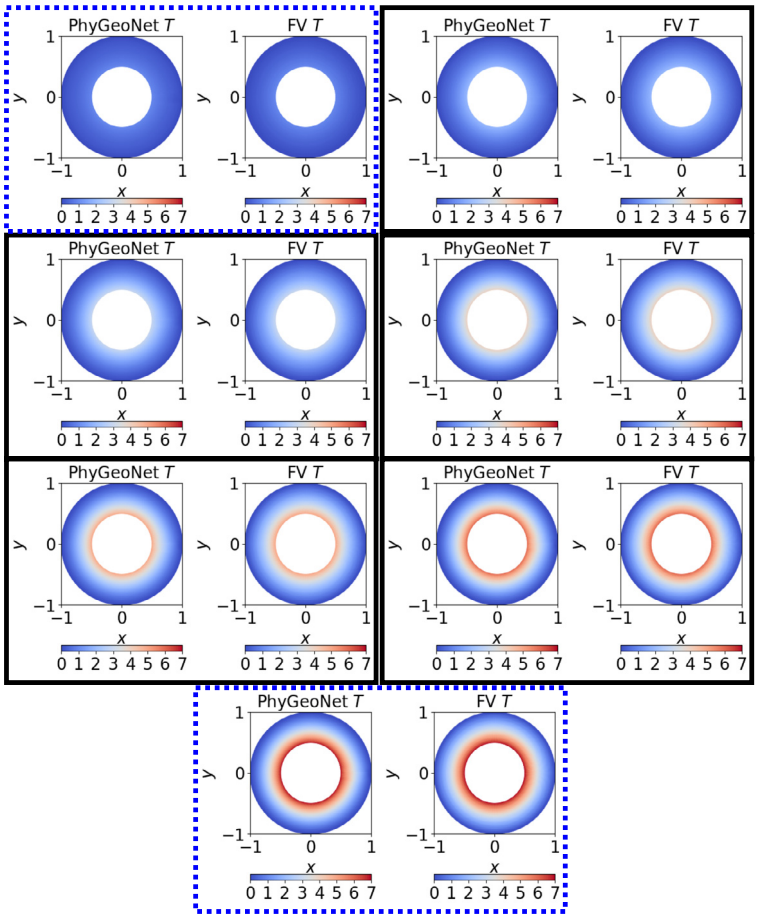


Fig. 10. The inner circle boundary temperature (T_{in}) is varying in $[1, 7]$ with a interval of $\delta T_{in} = 1$. For cases in blue dashed blocks, the PhyGeoNet is trained in a PDE-driven fashion without data, while cases in black solid blocks are pure test points without training.

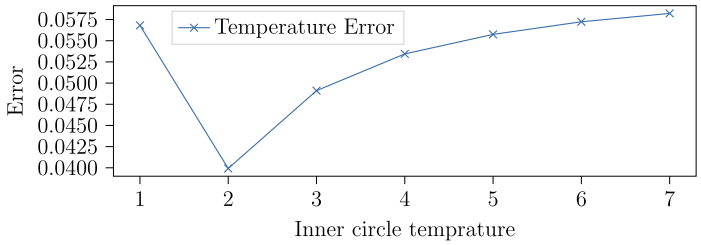


Fig. 11. Relative errors of PhyGeoNet predictions of different inner BCs.

$$\begin{aligned} x_l &= s \cdot \cos(2\pi\, y_l) - 0.5, \\ x_r &= -s \cdot \cos(2\pi\, y_r) + 0.5, \end{aligned} \tag{19}$$

where (x_l, y_l) and (x_r, y_r) are coordinates of left and right vessel walls with $y_l, y_r \in [-0.25, 0.25]$; $s \in [-0.1, 0.1]$ is a scalar controlling the degree of stenosis or aneurysm. The top and bottom boundaries are defined as $(x_t, y_t) \in [-0.5, 0.5] \times \{0.25\}$ and $(x_b, y_b) \in [-0.5, 0.5] \times \{-0.25\}$. As shown in Fig. 12, the geometry of the physical domain can be controlled by changing the scalar parameter s ($s > 0$ for stenosis and $s < 0$ for aneurysm). The boundary conditions of pressure and velocity are set as follows: the velocity inlet of $\mathbf{u} = [0, 0.4]$ and $\nabla p \cdot \mathbf{n} = 0$ is imposed at the lower edge (green); no-slip wall BC ($\mathbf{u} = [0, 0]$ and $\nabla p \cdot \mathbf{n} = 0$) is prescribed on the curved boundaries of both sides; the outlet is defined at the upper edge with $\nabla \mathbf{u} \cdot \mathbf{n} = 0$ and $p = 0$. Again, $\mathbf{n}$ is the local wall-normal vector.

Given that the physical domain is varying in shape, coordinates of the physical domain is used as the input for the PhyGeoNet to indicate different vascular geometries. Similarly, the Navier-Stokes-driven (Eq. (17)) training is only conducted on three geometries as shown in blue dashed blocks in Figs. 13 and 14, and the trained modeled are tested by six different

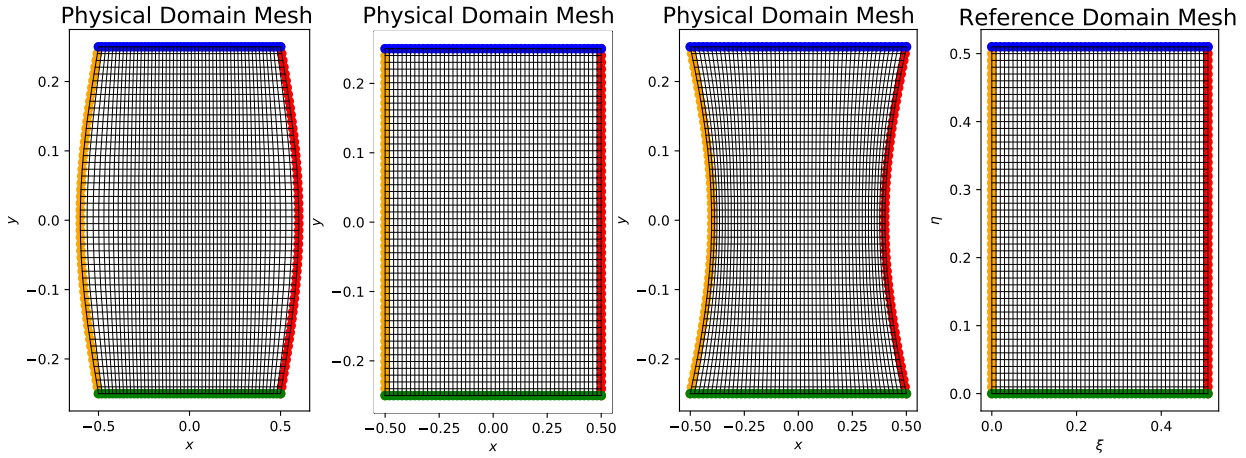


Fig. 12. From left to right: the first is aneurysm ($s = -0.1$), the second is neutral ($s = 0.0$), the third is stenosis ($s = 0.1$), and the last is the reference domain.

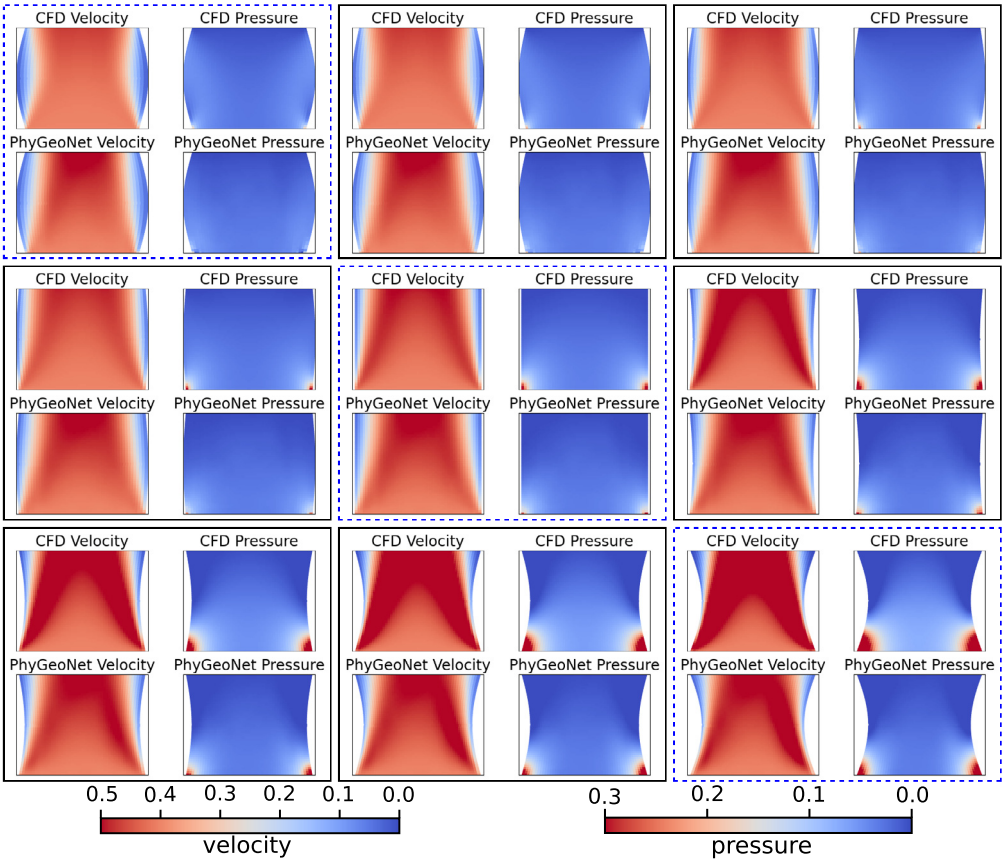


Fig. 13. The velocity and pressure contours from aneurysm ($s = -0.1$) to stenosis ($s = 0.1$) with interval of $\delta s = 0.025$ ($Re = 20$). For cases in blue dashed blocks, the PhyGeoNet is trained in a PDE-driven fashion without data, while cases in black solid blocks are pure test points without training.

new interpolated geometries. The velocity and pressure contour plots for Re = 20 and 200 are presented in Figs. 13 and 14, respectively. The learning curves and physics-based loss convergence histories are given in Fig. D.26d.

For all cases, the PhyGeoNet predictions generally agree with the CFD benchmarks and the contours are visually close to each other. Although no training data is used, the PhyGeoNet can capture the boundary layer variations due to the geometry changes, high-pressure regions located at the two lower corners, and the velocity acceleration near the outlet for all different shapes reasonably well.

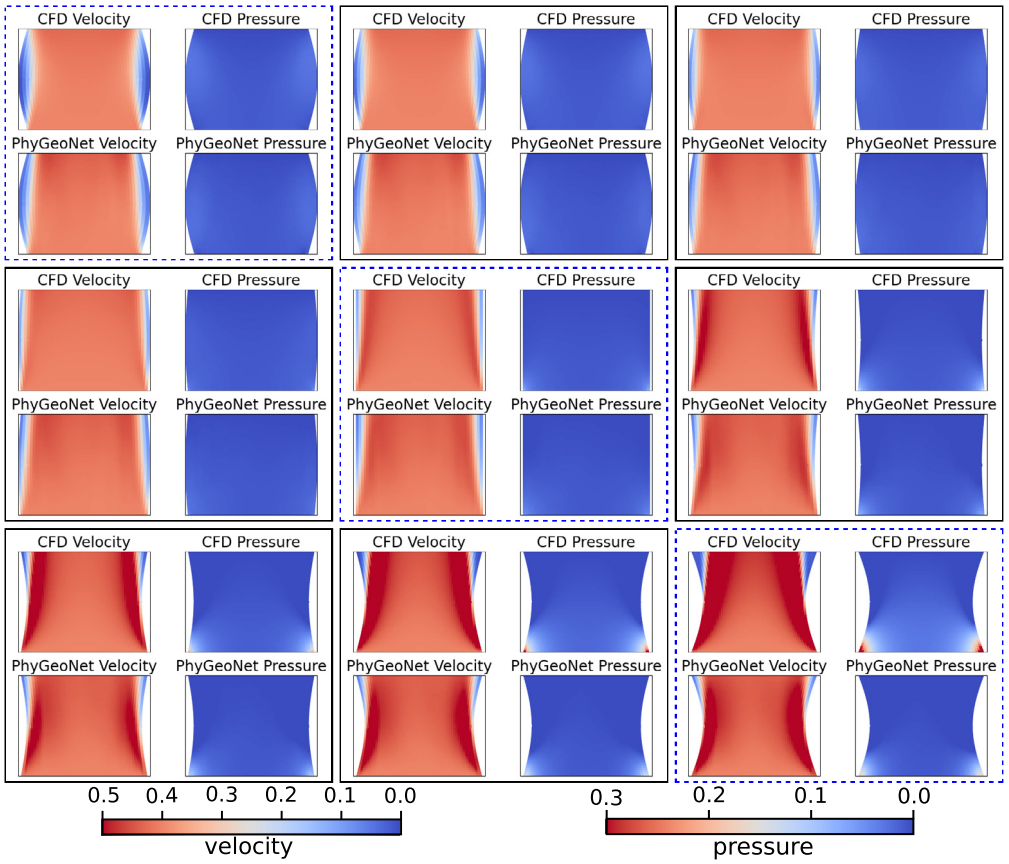
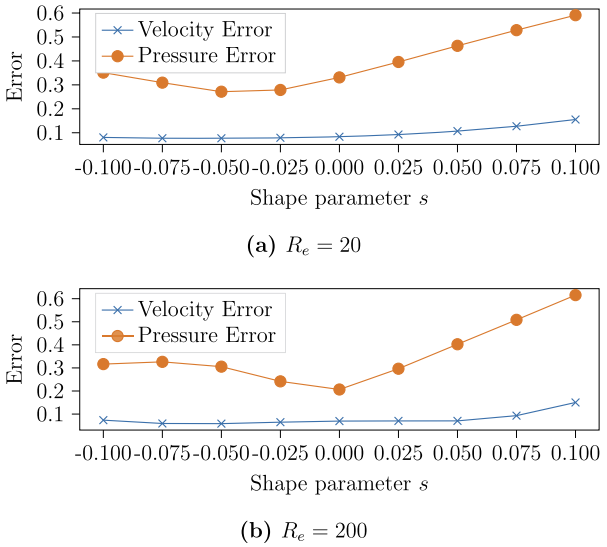


Fig. 14. The velocity and pressure contours from aneurysm ($s = -0.1$) to stenosis ($s = 0.1$) with interval of $\delta s = 0.025$ ($Re = 200$). For cases in blue dashed blocks, the PhyGeoNet is trained in a PDE-driven fashion without data, while cases in black solid blocks are pure test points without training.



(a) $R_e = 20$

(b) $R_e = 200$

Fig. 15. Relative prediction error of velocity and pressure v.s. the geometry control parameter s ($s > 0$ for stenosis and $s < 0$ for aneurysm).

The relative errors of the predicted velocity and pressure versus geometry parameter s are given in Fig. 15. We can see that the velocity is accurately predicted with a low relative error, particularly for aneurysm cases ($s < 0$). However, the prediction error of pressure in terms of absolute value is relatively large, though the pressure distribution over the

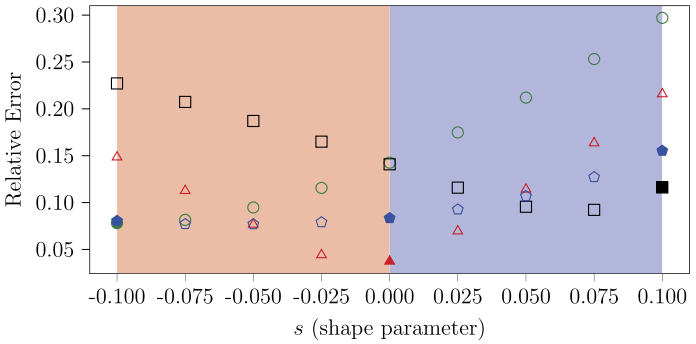


Fig. 16. Comparison of the model performance for four different training scenarios: **(1)** training only in the stenosis region ($s > 0$, ▭) with one training sample (■) and eight testing samples (□); **(2)** training only in the aneurysm region ($s < 0$, ▭) with one training sample (●) and eight testing samples (○); **(3)** training only on the straight vessel ($s = 0$) with one training sample (▲) and eight testing samples (△); **(4)** training on both regions with three training samples (⬟) and six testing samples (⬠).

entire domain can be captured reasonably well (Figs. 13 and 14). For both velocity and pressure, the prediction performance deteriorates as the geometry becomes more stenotic ($s > 0$).

The results showed above are from the network trained by seeing all three different families of the input geometries, i.e., straight, stenosis, and aneurysm vessels. Although our label-free PhyGeoNet is not constrained by the availability of labeled data and thus can be freely trained by selecting any input scenarios to leverage the power of DNNs for interpolatory problems, it's still interesting to investigate the "extrapolation" performance in case of insufficient training due to computational limitations. Here we will compare the prediction performance for four different training scenarios (see Fig. 16): (1) the model is trained only on one input geometry (■) in the stenosis region ($s > 0$, ▭) and evaluated on eight unseen geometries (□) in both stenosis and aneurysm regions; (2) the model is trained only on one input geometry (■) in the aneurysm region ($s < 0$, ▭) and tested on eight unseen geometries (□) in both stenosis and aneurysm regions; (3) the model is trained on straight vessels (▲) and evaluated on eight unseen stenosis and aneurysm vessels (△); (4) the model is trained on three representative geometries (⬟) from three families and evaluated on the other six unseen geometries (⬠). The comparison results for cases with Reynolds numbers around $Re = 20$ are shown in Fig. 16. (The results of the cases with $Re = 200$ are similar and thus are omitted here for brevity.) As expected, the prediction error increases when the testing points move away from the training input points, similar to what we observed in data-driven cases. It is clear that the model performance is much better for the predictions of interpolation points than that of the extrapolation ones. Since the availability of training labels will not limit our present DNN model, which thus has strong flexibility to select "important" input points based on experimental design algorithms [91], for further optimization of training cost and predictive accuracy.

3.2.3. *Poisson equations with spatially-varying source (Case 5)*

In the last numerical example, we will apply the PhyGeoNet to learn the solutions in a high-dimension parameter space. Consider a Poisson equation (Eqn. (20)) with homogeneous Dirichlet boundary conditions ($T_c = 10$) and spatially-varying source term $f(\mathbf{x})$,

$$\begin{aligned} \nabla \cdot (\nabla T(\mathbf{x})) + f(\mathbf{x}) = 0,\ \mathbf{x} \in \Omega_p, \\ T(\mathbf{x}) = T_c,\ \mathbf{x} \in \partial\Omega_p, \end{aligned} \tag{20}$$

where Ω_p represents the physical domain as shown in Fig. 17a. The goal here is to learn the parameter-to-solution map $f(\mathbf{x}) \rightarrow T(\mathbf{x})$, where the source field has a dimension of 900 (30×30) on the discrete domain. The high-dimensional spatially-varying source term $f(\mathbf{x})$ is modeled as a Gaussian random field,

$$f(\mathbf{x}) \sim \mathcal{GP}(\mathbf{0}, K(\mathbf{x}, \mathbf{x}')), \quad K(\mathbf{x}, \mathbf{x}') = \sigma_0^2 \exp(\frac{(\mathbf{x} - \mathbf{x}')}{2l^2}), \tag{21}$$

where $K(\mathbf{x}, \mathbf{x}')$ is the exponential kernel function with the stationary variance of $\sigma_0 = 100$ and length scale of $l = 0.5$. Because of the smoothness of the field, the Gaussian process can be expressed in a compact form based on Karhunen-Loeve (K-L) expansion [93],

$$f(\mathbf{x}) = \sum_{i=1}^{n_k \rightarrow \infty} \sqrt{\lambda_i} \phi_i(\mathbf{x}) \omega_i, \tag{22}$$

where λ_i and $\phi_i(\mathbf{x})$ are eigenvalues and eigenfunctions of the kernel K; ω_i are an uncorrelated random variables with zero mean and unit variance. The K-L expansion can be truncated to its first ten modes to capture over 99% energy of the original field. Namely, the 900-dimensional spatially-varying source field can be reconstructed based on the coefficients of the first

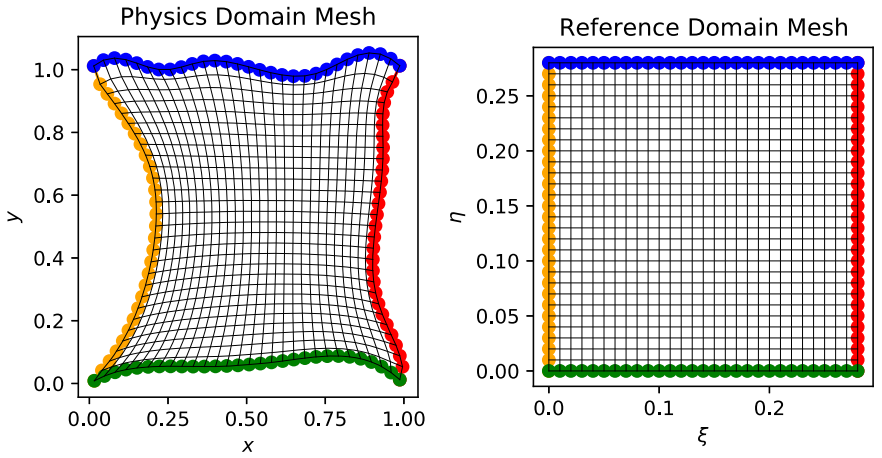


Fig. 17. The physical domain (left) and reference domain (right) for the parametric Poisson equation. The corresponding edges of two domains are indicated by the same color.

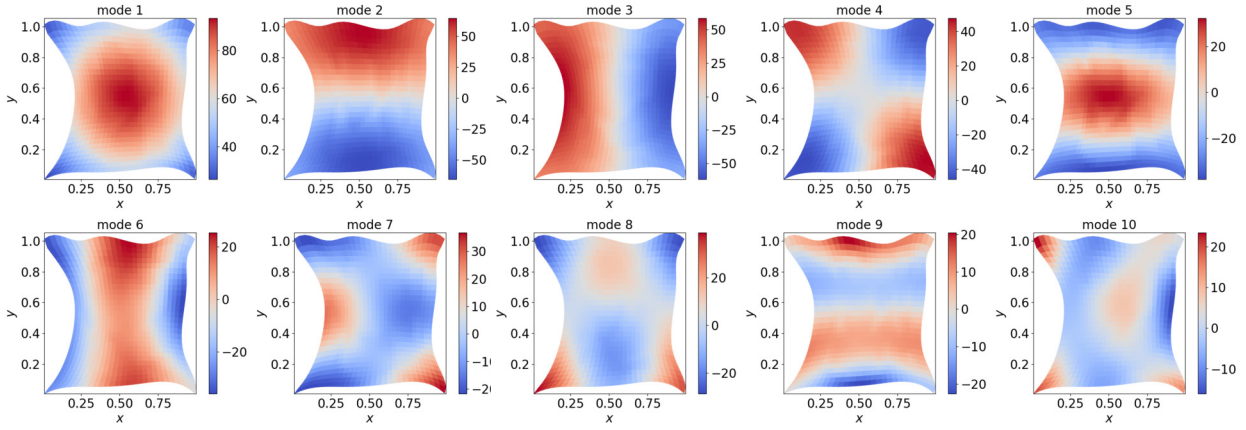


Fig. 18. The first ten K-L mode functions ($\sqrt{\lambda}\phi(\mathbf{x})$) evaluated on the mesh.

ten K-L modes (Fig. 18), and thus the intrinsic dimension of the parameter space is reduced to ten (i.e., $\boldsymbol{\omega} = [\omega_1, ..., \omega_{10}] \in \mathbb{R}^{10}$). Even so, to capture the solution response surface in a ten-dimensional parameter space is challenging. In our learning framework, the PhyGeoNet takes the input of the source field $f(\mathbf{x})$ and output the solution field($T(\mathbf{x})$). To train the model, we minimize the PDE loss on 256 randomly-sampled input source fields. Once the model is trained, we can not only solve the solutions of the 256 source fields simultaneously, but also build a surrogate model to rapidly predict the solution of any new source field. To test the model, we randomly sample 744 unseen source fields and use the trained PhyGeoNet to predict the corresponding solution fields. The prediction results on a subset of the training and testing samples are shown in Figs. 19 and 20, respectively, compared against the FV-based reference. Overall, the mean relative error for the training samples is 0.0687 with the variance of 0.0025. For testing samples, the mean relative error slightly increases to 0.0703 with the variance of 0.0041. These results demonstrate the great potential of the PhyGeoNet as the surrogate for high-dimensional problems.

4. Discussion

4.1. Comparison between PhyGeoNet and PINN

As mentioned in Section 1, the majority of the existing works on PDE-constrained learning have adopted a pointwise formulation using fully-connected neural networks (FC-NN), e.g., physics-informed neural networks (PINN) [2,4]. Although the pointwise FC-NN can leverage automatic differentiation to compute derivatives analytically, the training may not be scalable for complex, large-scale problems. The CNN-based formulations explored in this work has the potential to largely enhance the training efficiency. To demonstrate the merit of the proposed CNN-based PhyGeoNet, quantitative performance comparisons between the PINN and PhyGeoNet are conducted. Specifically, the PINN with the network structure proposed in [4] is compared with the PhyGeoNet on solving the steady Navier-Stokes equation defined in Section 3.1.2. Two different

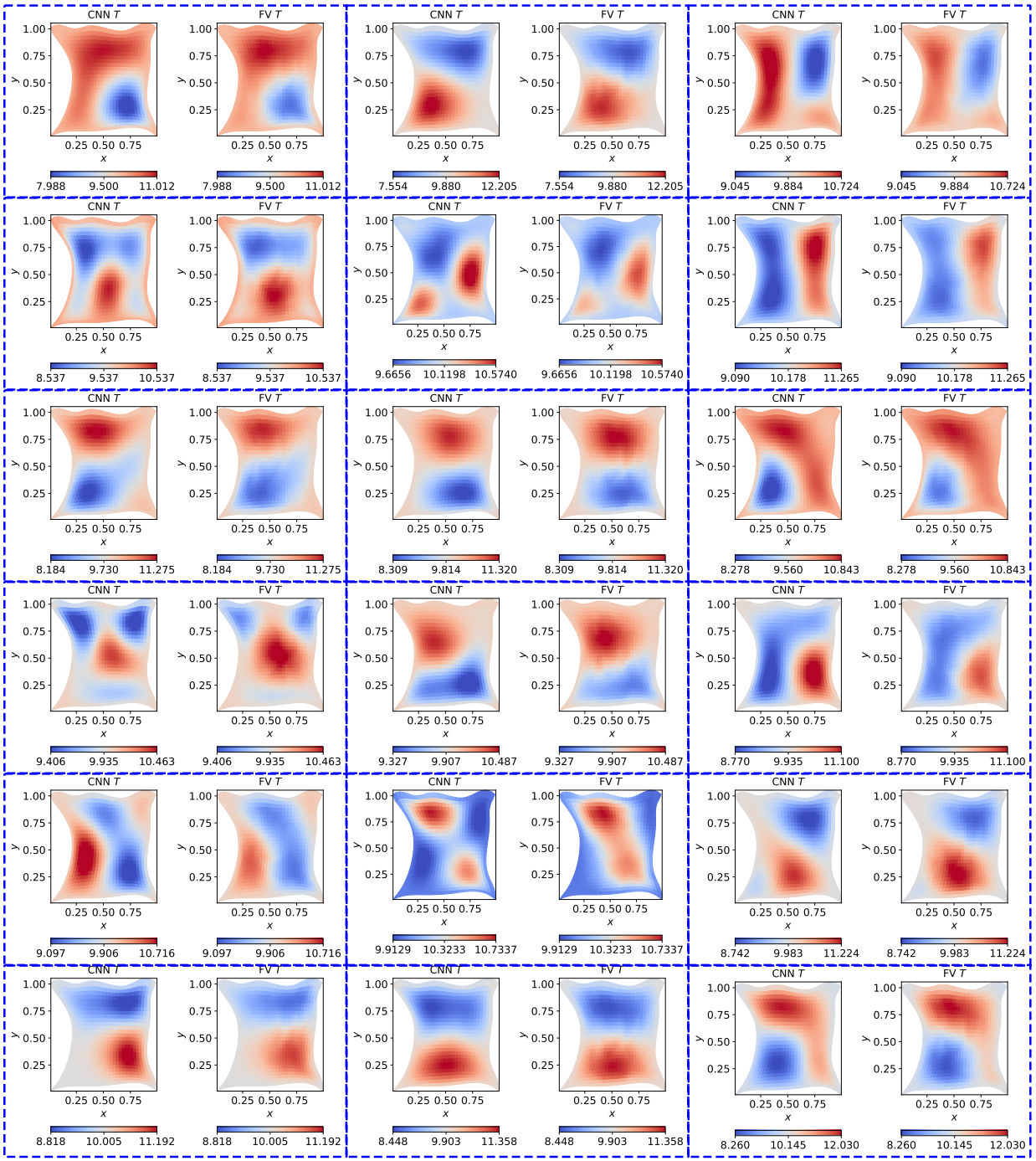


Fig. 19. PhyGeoNet solution predictions on a subset of training samples of the spatially-varying source term.

scenarios are considered: (1) to compare the training cost of reaching the convergence, and (2) to compare the predictive accuracy with the same training budget.

4.1.1. Training cost comparison for convergence

We first investigate the training cost and prediction performance of the PINN (FC-NN) and PhyGeoNet (CNN), when both cases are fully converged. The contour comparisons of velocity and pressure solutions are shown in Fig. 21. Both the PINN and PhyGeoNet can capture the general flow pattern. However, notable discrepancies are observed in the PINN-predicted velocity field, especially in the region where the velocity is developing, while the PhyGeoNet prediction result is more accurate and agree with the CFD benchmark very well. Moreover, the PINN fails to accurately predict the pressure

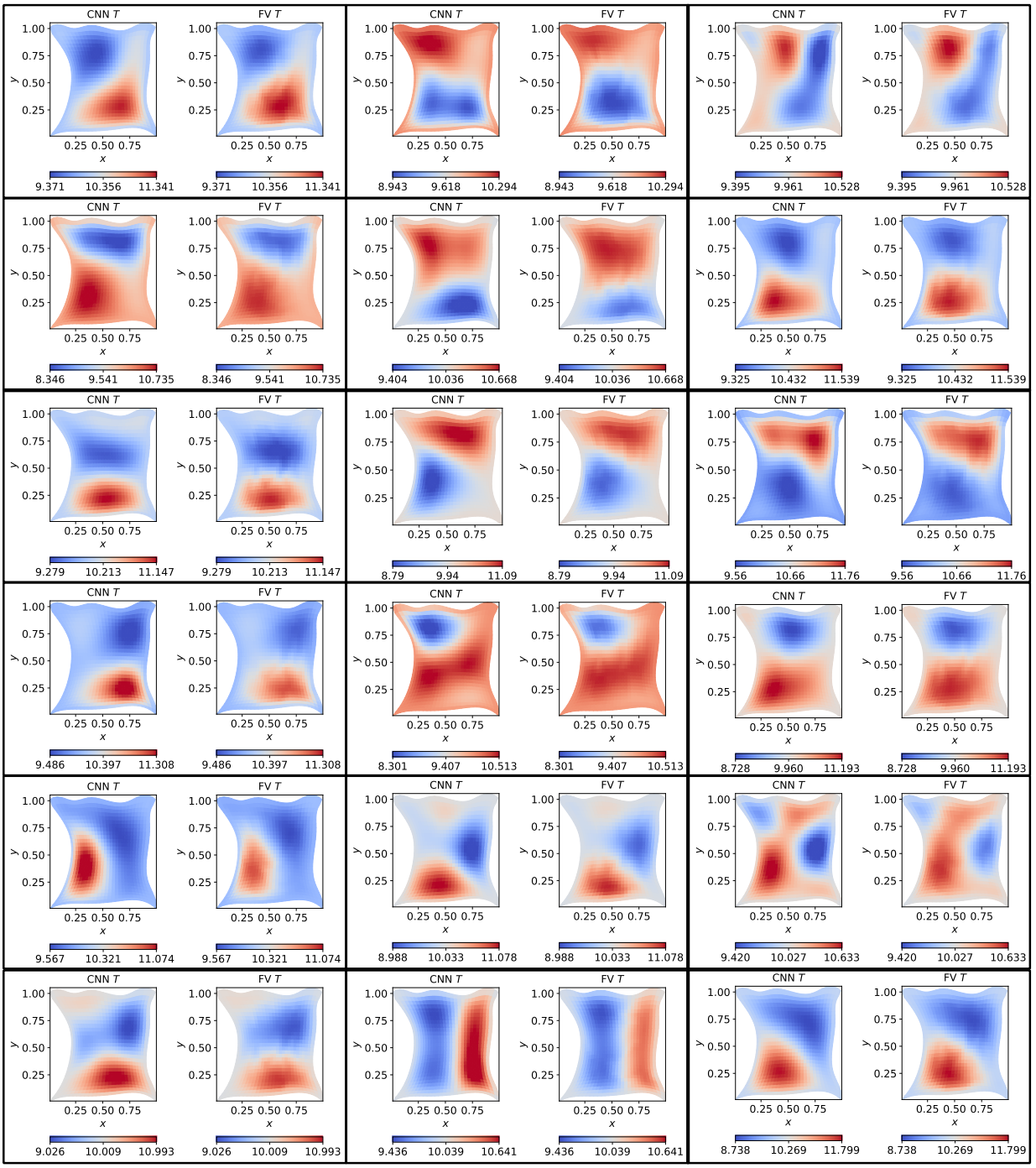


Fig. 20. PhyGeoNet solution predictions on a subset of testing samples of the spatially-varying source term.

distribution, particularly in the near-inlet zone. In contrast, the PhyGeoNet predicted pressure contour also well agrees with the CFD result.

The comparison of the training costs for both networks being fully trained is listed in Table 2. We found that PINN needs about 10 times more iterations than the PhyGeoNet does to reach the convergence. It takes about 2.5 hours to train the PINN, while it only needs around 15 minutes to train the PhyGeoNet but with higher predictive accuracy. This comparison highlights that the PhyGeoNet converges much faster than the PINN does. The main reason is that the convolution operation with shift-invariant filters enables the update of the entire field in each iteration, which is far more efficient than the pointwise training.

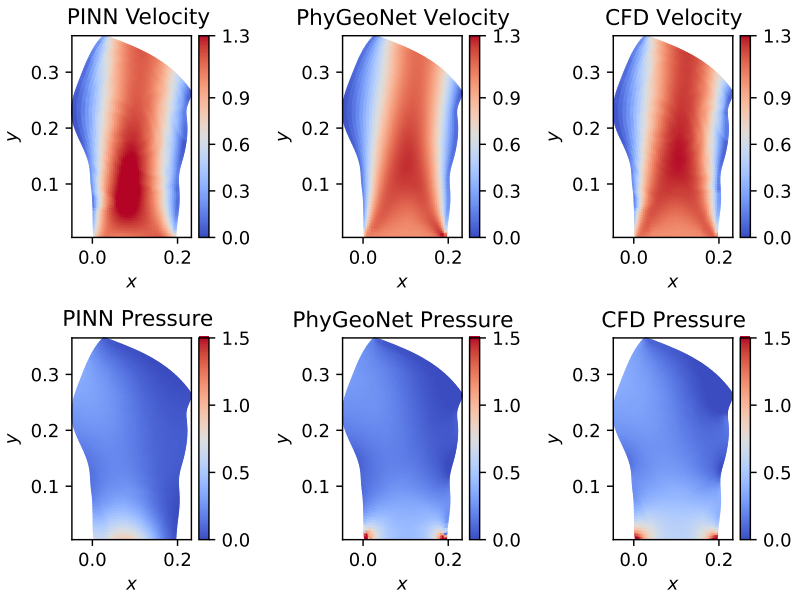


Fig. 21. Velocity (the first row) and pressure (the second row) contours of PINN (left column), PhyGeoNet (middle column), and CFD benchmark (right column). Both PINN and PhyGeoNet are fully trained at $R_e = 20$.

Table 2
Comparison of training costs and prediction performance between PINN and PhyGeoNet, both of which are fully trained.

Information	Name	
	PhyGeoNet	PINN
Number of iterations	1.5×10^3	1.6×10^4
Wall-clock time (s)	9.12×10^2	9.454×10^3
Velocity relative error	0.078	0.130
Pressure relative error	0.346	0.632
Device name	Nvida GeForce RTX 2080	

Table 3
Comparison of predictive accuracy of PINN and PhyGeoNet with the same amount of training budget.

Information	Name	
	PhyGeoNet	PINN
Number of iterations	1.5×10^3	1.6×10^3
Wall-clock time (s)	9.12×10^2	10.34×10^2
Velocity relative error	0.078	0.321
Pressure relative error	0.346	0.850
Device name	Nvida GeForce RTX 2080	

4.1.2. Predictive accuracy comparison with the same training budget

To further assess the proposed PhyGeoNet, we conduct another comparison experiment by fixing the total computational budget for training. Namely, both the PhyGeoNet and PINN are trained with a similar amount of iterations, and the training costs (wall time) are thus roughly the same. It can be seen from Fig. 22 that the PhyGeoNet can accurately predict both velocity and pressure fields after 1.5×10^3 training iterations. However, with the same training budget, the PINN's predictions are clearly not accurate and the relative errors of the predicted velocity and pressures are over 30% and 85%, respectively, as shown in Table 3. The comparison study has demonstrated the advantages of the CNN-based PhyGeoNet over the PINN using FC-NN formulations. Since the computational budget is usually limited in practical applications. the PhyGeoNet is preferred due to its fast convergence and higher accuracy.

4.2. Success, limitation, and future perspectives of the current framework

The main contribution of this work lies in the algorithmic development that enables the direct use of classic CNN backbones to solve PDEs on irregular domains. Although the current work focuses on *label-free PDE-driven* learning (i.e., solving non-parametric/parametric PDEs without data), the proposed method can also be applied to enable *data-driven*

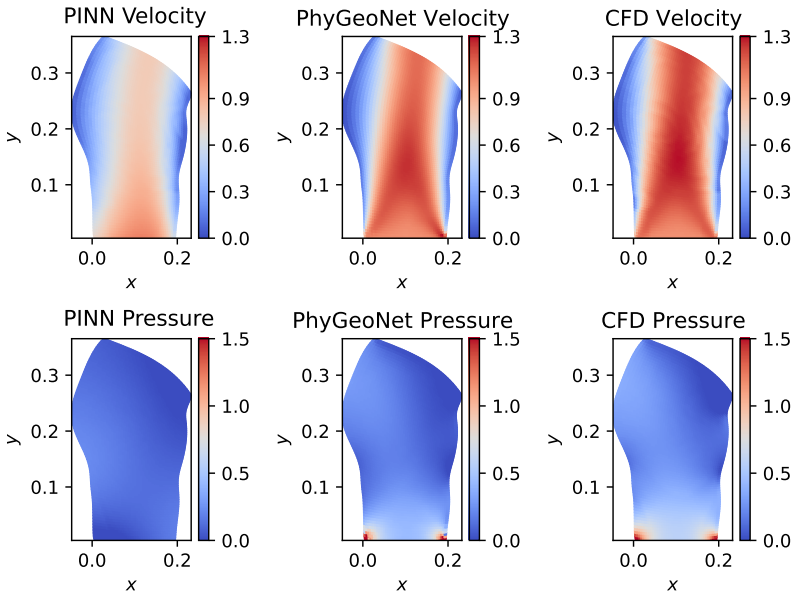


Fig. 22. Velocity (the first row) and pressure (the second row) contours of PINN (left column), PhyGeoNet (middle column), and CFD benchmark (right column). Both PINN and PhyGeoNet are trained with the same number of iterations ($\approx 1.6 \times 10^3$ iterations) at $R_e = 20$.

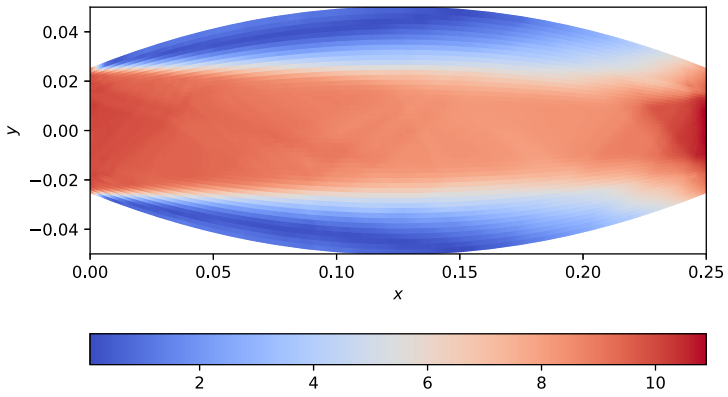
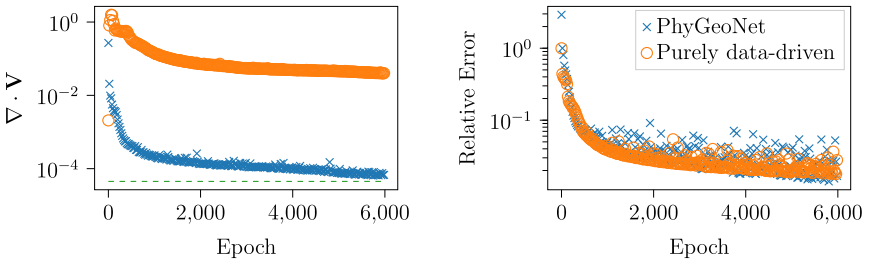


Fig. 23. RANS mean flow field of a diverging-converging channel at $Re = 50000$.

CNNs with PDE constraints for dealing with problems in irregular domains. For example, the divergence-free condition has been utilized to constrain the data-driven learning for incompressible turbulent flows in recent literature [61], which, however, were limited to image-like regular domains for classic CNNs. The coordinate transformation scheme presented in this work can be applied to impose this constraint on non-rectangular domains, enabling PDE-constrained, data-driven CNN solutions. Here we will use an additional test case to demonstrate this idea. Specifically, we consider a 2D turbulent flow in a non-rectangular domain at $Re = 50000$, obtained from the RANS simulation with the k-ϵ turbulence model [94]. The flow domain and mean velocity field are given in Fig. 23. We use CNN to fit the flow data in a data-driven manner, where the divergence-free constraint is enforced in this irregular domain. As shown in Fig. 24, although the training errors reach the same level, the constrained data-driven solution is more physical compared to the purely data-driven one, since the mass conservation is better satisfied. However, a slightly larger oscillation is observed when the training is constrained by the physical laws.

Although showing great promise, the current framework has several limitations, and many technical challenges are still present. First, this work focuses on parametrized steady-state PDEs, and the current learning architecture cannot deal with dynamic systems. Nonetheless, the coordinate transformation based DL solutions can also be applied to solve spatiotemporal PDEs with some further developments. For example, we could use the finite difference (e.g., central differencing) to formulate time derivatives in the physics-based loss function and build Long-Short Term Memory (LSTM) network architecture to capture temporal coherence. Recent studies have demonstrated the effectiveness of LSTM for dynamic problems using physics-informed learning [6]. We also can formulate an auto-regressive network architecture [15] to learn the temporal dependence, and existing numerical time-stepping schemes, e.g., Euler or Runge-Kutta methods, can be leveraged. Second,



(a) Convergence of mass conservation. (b) Convergence history of relative error.

Fig. 24. Comparison of data-driven CNN solutions with and without the divergence-free constraint for RANS turbulence data on an irregular domain.

the current approach has challenges to deal with very complex domains (e.g., with more than five C_0 continuous edges), since it is difficult to establish a one-to-one mapping for such geometries and the elliptic coordinate transformation cannot be directly applied. To address this issue, we can utilize domain decomposition, which will decompose the complex domain into multiple sub-blocks, and each of them contains four C_0 edges. In each sub-block, the current coordinate transformation can be applied, and the continuity between two sub-blocks will be strictly imposed using CNN padding. Third, we acknowledge that our current framework has not been applied to solve the Navier-Stokes equations in turbulence regimes without labels. To achieve this goal, grant technical challenges are present and significant developments are expected: (1) if directly resolving the Navier-Stokes equations at turbulence levels, a very high-resolution mesh is required to resolve the Kolmogorov scales, which would cause computational and memory issues in the current formulation; (2) if modeling instead of resolving sub-scale turbulence structures, appropriate closure or subgrid-scale models need to be built into the networks, and corresponding model-form uncertainty should be considered. We will explore these directions in the future work and continue to fill the current gaps for more complex and realistic applications.

5. Conclusion

This paper presented a novel method of physics-informed CNN (PhyGeoNet) for solving parametric PDEs on irregular domains without any labeled data. An elliptic mapping is introduced to formulate the transformation between the irregular physical domain and regular reference domain, enabling the direct use of powerful classic CNN backbones for non-rectangular geometries and non-uniform grids. Since the elliptic mapping is obtained numerically, the proposed method can be applied to complex geometries that are not parameterizable. The effectiveness and merit of the proposed PhyGeoNet have been demonstrated by solving a number of PDE systems on irregular domains, including nonparametric/parametric heat equations and Navier-Stokes equations. Moreover, the PhyGeoNet was compared with the state-of-the-art PINN in terms of accuracy and efficiency. The comparison results have shown that the convergence speed of the PhyGeoNet is more than an order of magnitude faster than that of the PINN (with FC-NN formulation), and the accuracy of the PhyGeoNet is much higher if the total training budget is fixed to be the same. These advantages highlight the potential of PhyGeoNet on solving more complex problems that are scalable.

CRediT authorship contribution statement

Han Gao: Data curation, Formal analysis, Methodology, Software, Validation, Visualization, Writing – original draft. **Luning Sun:** Formal analysis, Methodology, Software, Writing – original draft. **Jian-Xun Wang:** Conceptualization, Funding acquisition, Methodology, Project administration, Supervision, Writing – review & editing.

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.

Acknowledgement

The authors would like to acknowledge the funds from National Science Foundation (NSF contract CMMI-1934300) and startup funds from the College of Engineering at University of Notre Dame in supporting this study. We also gratefully acknowledge the discussion with Dr. Matthew J. Zahr in the early stage of this research. The authors finally thank the anonymous reviewers for their insightful comments and suggestions to improve the quality of this paper.

Appendix A. Proof of forward elliptic transformation

Proof. The derivatives of reference coordinates with respect to physical coordinates can be expressed as [77],

$$\frac{\partial \xi}{\partial x} = \frac{1}{J}\frac{\partial y}{\partial \eta}, \tag{A.1a}$$

$$\frac{\partial \eta}{\partial x} = -\frac{1}{J}\frac{\partial y}{\partial \xi}, \tag{A.1b}$$

$$\frac{\partial \xi}{\partial y} = -\frac{1}{J}\frac{\partial x}{\partial \eta}, \tag{A.1c}$$

$$\frac{\partial \eta}{\partial y} = \frac{1}{J}\frac{\partial x}{\partial \xi}, \tag{A.1d}$$

where $J = \frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi} \neq 0$ is the determinant of the Jacobian matrix.

Based on Eq. (13), derivatives of the forward map $\mathcal{G}$ can be calculated. Substitute Eq. (A.1a) into Eq. (13a), we have

$$\begin{aligned}\frac{\partial^2 \xi}{\partial x^2} = \frac{1}{J^3}\Big[&- \frac{\partial^2 x}{\partial \eta^2}\left(\frac{\partial y}{\partial \xi}\right)^2\frac{\partial y}{\partial \eta} + \frac{\partial^2 y}{\partial \eta^2}\frac{\partial x}{\partial \eta}\left(\frac{\partial y}{\partial \xi}\right)^2 + 2\frac{\partial^2 x}{\partial \xi \partial \eta}\frac{\partial y}{\partial \xi}\left(\frac{\partial y}{\partial \eta}\right)^2 - \\ &2\frac{\partial^2 y}{\partial \xi \partial \eta}\frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial^2 x}{\partial \xi^2}\left(\frac{\partial y}{\partial \eta}\right)^3 + \frac{\partial^2 y}{\partial \xi^2}\frac{\partial x}{\partial \eta}\left(\frac{\partial y}{\partial \eta}\right)^2\Big]\end{aligned} \tag{A.2}$$

Substitute Eq. (A.1c) into Eq. (13b), we can get

$$\begin{aligned}\frac{\partial^2 \xi}{\partial y^2} = \frac{1}{J^3}\Big[&\frac{\partial^2 y}{\partial \eta^2}\left(\frac{\partial x}{\partial \xi}\right)^2\frac{\partial x}{\partial \eta} - \frac{\partial^2 x}{\partial \eta^2}\frac{\partial y}{\partial \eta}\left(\frac{\partial x}{\partial \xi}\right)^2 - 2\frac{\partial^2 y}{\partial \xi \partial \eta}\frac{\partial x}{\partial \xi}\left(\frac{\partial x}{\partial \eta}\right)^2 + \\ &2\frac{\partial^2 x}{\partial \xi \partial \eta}\frac{\partial y}{\partial \eta}\frac{\partial x}{\partial \xi}\frac{\partial x}{\partial \eta} + \frac{\partial^2 y}{\partial \xi^2}\left(\frac{\partial x}{\partial \eta}\right)^3 - \frac{\partial^2 x}{\partial \xi^2}\frac{\partial y}{\partial \eta}\left(\frac{\partial x}{\partial \eta}\right)^2\Big]\end{aligned} \tag{A.3}$$

Substitute Eq. (A.2) and Eq. (A.3) into Eq. (9) and then multiply J^3 on both side, we get

$$\begin{aligned}&- \frac{\partial^2 x}{\partial \eta^2}\left(\frac{\partial y}{\partial \xi}\right)^2\frac{\partial y}{\partial \eta} + \frac{\partial^2 y}{\partial \eta^2}\frac{\partial x}{\partial \eta}\left(\frac{\partial y}{\partial \xi}\right)^2 + 2\frac{\partial^2 x}{\partial \xi \partial \eta}\frac{\partial y}{\partial \xi}\left(\frac{\partial y}{\partial \eta}\right)^2 - \\ &2\frac{\partial^2 y}{\partial \xi \partial \eta}\frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial^2 x}{\partial \xi^2}\left(\frac{\partial y}{\partial \eta}\right)^3 + \frac{\partial^2 y}{\partial \xi^2}\frac{\partial x}{\partial \eta}\left(\frac{\partial y}{\partial \eta}\right)^2 + \\ &\frac{\partial^2 y}{\partial \eta^2}\left(\frac{\partial x}{\partial \xi}\right)^2\frac{\partial x}{\partial \eta} - \frac{\partial^2 x}{\partial \eta^2}\frac{\partial y}{\partial \eta}\left(\frac{\partial x}{\partial \xi}\right)^2 - 2\frac{\partial^2 y}{\partial \xi \partial \eta}\frac{\partial x}{\partial \xi}\left(\frac{\partial x}{\partial \eta}\right)^2 + \\ &2\frac{\partial^2 x}{\partial \xi \partial \eta}\frac{\partial y}{\partial \eta}\frac{\partial x}{\partial \xi}\frac{\partial x}{\partial \eta} + \frac{\partial^2 y}{\partial \xi^2}\left(\frac{\partial x}{\partial \eta}\right)^3 - \frac{\partial^2 x}{\partial \xi^2}\frac{\partial y}{\partial \eta}\left(\frac{\partial x}{\partial \eta}\right)^2 = 0\end{aligned} \tag{A.4}$$

Then substitute Eq. (A.1b) into Eq. (13a), we get

$$\begin{aligned}\frac{\partial^2 \eta}{\partial x^2} = \frac{1}{J^3}\Big[&\frac{\partial^2 x}{\partial \eta^2}\left(\frac{\partial y}{\partial \xi}\right)^3 - 2\frac{\partial^2 x}{\partial \eta \partial \xi}\left(\frac{\partial y}{\partial \xi}\right)^2\frac{\partial y}{\partial \eta} - \frac{\partial^2 y}{\partial \eta^2}\frac{\partial x}{\partial \xi}\left(\frac{\partial y}{\partial \xi}\right)^2 \\ &+ \frac{\partial^2 x}{\partial \xi^2}\frac{\partial y}{\partial \xi}\left(\frac{\partial y}{\partial \eta}\right)^2 + 2\frac{\partial^2 y}{\partial \xi \partial \eta}\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial^2 y}{\partial \xi^2}\frac{\partial x}{\partial \xi}\left(\frac{\partial y}{\partial \eta}\right)^2\Big].\end{aligned} \tag{A.5}$$

Substitute Eq. (A.1d) into Eq. (13b), we get

$$\begin{aligned}\frac{\partial^2 \eta}{\partial y^2} = \frac{1}{J^3}\Big[&- \frac{\partial^2 y}{\partial \eta^2}\left(\frac{\partial x}{\partial \xi}\right)^3 + 2\frac{\partial^2 y}{\partial \eta \partial \xi}\left(\frac{\partial x}{\partial \xi}\right)^2\frac{\partial x}{\partial \eta} + \frac{\partial^2 x}{\partial \eta^2}\frac{\partial y}{\partial \xi}\left(\frac{\partial x}{\partial \xi}\right)^2 \\ &- \frac{\partial^2 y}{\partial \xi^2}\frac{\partial x}{\partial \xi}\left(\frac{\partial x}{\partial \eta}\right)^2 - 2\frac{\partial^2 x}{\partial \xi \partial \eta}\frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \xi}\frac{\partial x}{\partial \eta} + \frac{\partial^2 x}{\partial \xi^2}\frac{\partial y}{\partial \xi}\left(\frac{\partial x}{\partial \eta}\right)^2\Big].\end{aligned} \tag{A.6}$$

Substitute Eq. (A.5) and Eq. (A.6) into Eq. (9) and then multiply J^3 on both side, we get

$$\begin{aligned}
&\frac{\partial^2 x}{\partial \eta^2}\left(\frac{\partial y}{\partial \xi}\right)^3 - 2\frac{\partial^2 x}{\partial \eta \partial \xi}\left(\frac{\partial y}{\partial \xi}\right)^2\frac{\partial y}{\partial \eta} - \frac{\partial^2 y}{\partial \eta^2}\frac{\partial x}{\partial \xi}\left(\frac{\partial y}{\partial \xi}\right)^2 \\
&+ \frac{\partial^2 x}{\partial \xi^2}\frac{\partial y}{\partial \xi}\left(\frac{\partial y}{\partial \eta}\right)^2 + 2\frac{\partial^2 y}{\partial \xi \partial \eta}\frac{\partial x}{\partial \xi}\frac{\partial y}{\partial \xi}\frac{\partial y}{\partial \eta} - \frac{\partial^2 y}{\partial \xi^2}\frac{\partial x}{\partial \xi}\left(\frac{\partial y}{\partial \eta}\right)^2 + \\
&- \frac{\partial^2 y}{\partial \eta^2}\left(\frac{\partial x}{\partial \xi}\right)^3 + 2\frac{\partial^2 y}{\partial \eta \partial \xi}\left(\frac{\partial x}{\partial \xi}\right)^2\frac{\partial x}{\partial \eta} + \frac{\partial^2 x}{\partial \eta^2}\frac{\partial y}{\partial \xi}\left(\frac{\partial x}{\partial \xi}\right)^2 \\
&- \frac{\partial^2 y}{\partial \xi^2}\frac{\partial x}{\partial \xi}\left(\frac{\partial x}{\partial \eta}\right)^2 - 2\frac{\partial^2 x}{\partial \xi \partial \eta}\frac{\partial y}{\partial \xi}\frac{\partial x}{\partial \xi}\frac{\partial x}{\partial \eta} + \frac{\partial^2 x}{\partial \xi^2}\frac{\partial y}{\partial \xi}\left(\frac{\partial x}{\partial \eta}\right)^2 = 0
\end{aligned} \tag{A.7}$$

Multiply Eq. (A.7) by $\frac{\partial y}{\partial \eta}$, multiply Eq. (A.4) by $\frac{\partial y}{\partial \xi}$ and sum together we can finally prove Eq. (10b) with the fact $J \neq 0$,

$$J\left(\alpha\frac{\partial^2 y}{\partial \xi^2} - 2\beta\frac{\partial^2 y}{\partial \xi \partial \eta} + \gamma\frac{\partial^2 y}{\partial \eta^2}\right) = 0. \tag{A.8}$$

Multiply Eq. (A.7) by $\frac{\partial x}{\partial \eta}$, multiply Eq. (A.4) by $\frac{\partial x}{\partial \xi}$ and sum together we can finally prove Eq. (10a) with the fact $J \neq 0$,

$$J\left(\alpha\frac{\partial^2 x}{\partial \xi^2} - 2\beta\frac{\partial^2 x}{\partial \xi \partial \eta} + \gamma\frac{\partial^2 x}{\partial \eta^2}\right) = 0. \quad \square \tag{A.9}$$

Note that, J is not fixed at the stage of building mappings, we can calculate its partial derivatives via Eq. (13a) and Eq. (13b) to get its derivatives,

$$\frac{\partial J}{\partial \xi} = \frac{\partial^2 x}{\partial \xi^2}\frac{\partial y}{\partial \eta} + \frac{\partial x}{\partial \xi}\frac{\partial^2 y}{\partial \eta \partial \xi} - \frac{\partial^2 y}{\partial \xi^2}\frac{\partial x}{\partial \eta} - \frac{\partial y}{\partial \xi}\frac{\partial^2 x}{\partial \eta \partial \xi}, \tag{A.10a}$$

$$\frac{\partial J}{\partial \eta} = \frac{\partial^2 x}{\partial \xi \partial \eta}\frac{\partial y}{\partial \eta} + \frac{\partial x}{\partial \xi}\frac{\partial^2 y}{\partial \eta^2} - \frac{\partial^2 y}{\partial \xi \partial \eta}\frac{\partial x}{\partial \eta} - \frac{\partial y}{\partial \xi}\frac{\partial^2 x}{\partial \eta^2}. \tag{A.10b}$$

However, after the mapping is built up, J remains as a constant non-zero point-wise value.

Appendix B. Convolution filter for derivatives on reference domain

For internal nodes on the reference domain, the first derivatives are approximated by 4th order central differences,

$$\frac{\partial u}{\partial \xi} \approx \frac{-u_{\xi+2\delta\xi,\eta} + 8u_{\xi+\delta\xi,\eta} - 8u_{\xi-\delta\xi,\eta} + u_{\xi-2\delta\xi,\eta}}{12\delta\xi} + O((\delta\xi)^4), \tag{B.1a}$$

$$\frac{\partial u}{\partial \eta} \approx \frac{-u_{\xi,\eta+2\delta\eta} + 8u_{\xi,\eta+\delta\eta} - 8u_{\xi,\eta-\delta\eta} + u_{\xi,\eta-2\delta\eta}}{12\delta\eta} + O((\delta\eta)^4), \tag{B.1b}$$

which can be expressed by an convolution filter as shown in Fig. B.25.

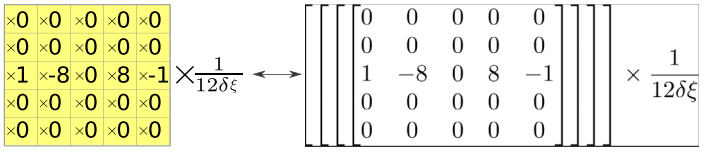


Fig. B.25. Finite difference based convolution filter for the differential operator: $\frac{\partial}{\partial \xi}$.

For nodes near and on the boundary, one-sided differences (i.e., upwind/downwind) are applied, and thus the artifacts due to padding (i.e., ghost cells) can be avoided. For example, third-order one-sided finite differences on the lower and left boundaries are given by

$$\frac{\partial u}{\partial \xi} \approx \frac{-11u_{\xi,\eta} + 18u_{\xi+\delta\xi,\eta} - 9u_{\xi+2\delta\xi,\eta} + 2u_{\xi+3\delta\xi,\eta}}{6\delta\xi} + O((\delta\xi)^3), \tag{B.2a}$$

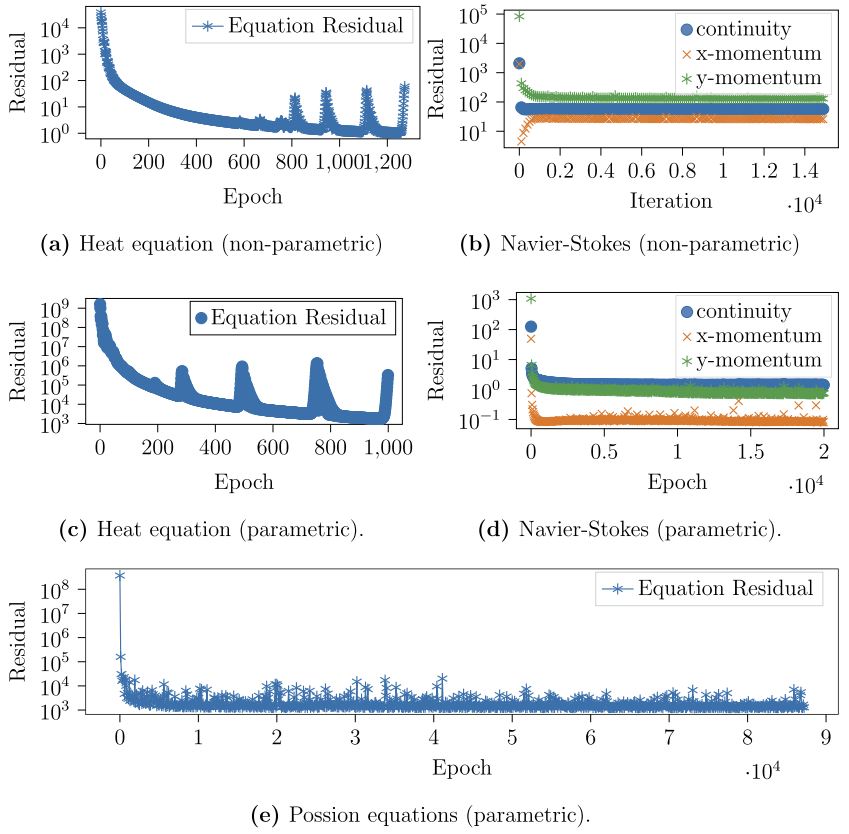
$$\frac{\partial u}{\partial \eta} \approx \frac{-11u_{\xi,\eta} + 18u_{\xi,\eta+\delta\eta} - 9u_{\xi,\eta+2\delta\eta} + 2u_{\xi,\eta+3\delta\eta}}{12\delta\eta} + O((\delta\eta)^3). \tag{B.2b}$$

Appendix C. Computational hyperparameter setting for PhyGeoNet

Table C.4
Hyperparameters setting for PhyGeoNet in each numerical experiment.

Hyperparameters	Case name				
	Case1	Case2	Case3	Case4	Case5
Number of iterations for training	1300	15000	1000	20000	87252
Number of training parameters	1	1	2	3	256
Number of testing parameters	1	1	7	9	744
Batch size	1	1	2	3	32
Learning Rate	0.001				

Appendix D. Convergence history of PhyGeoNet training



(a) Heat equation (non-parametric)

(b) Navier-Stokes (non-parametric)

(c) Heat equation (parametric).

(d) Navier-Stokes (parametric).

(e) Possion equations (parametric).

Fig. D.26. Convergence histories of the PhyGeoNet training.

References

[1] J. Sirignano, K. Spiliopoulos, DGM: a deep learning algorithm for solving partial differential equations, J. Comput. Phys. 375 (2018) 1339–1364.
[2] 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.
[3] 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.
[4] 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. Eng. 361 (2020) 112732.
[5] S.L. Brunton, B.R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annu. Rev. Fluid Mech. 52 (2020) 477–508.
[6] R. Zhang, Y. Liu, H. Sun, Physics-informed multi-LSTM networks for metamodeling of nonlinear structures, arXiv preprint arXiv:2002.10253, 2020.
[7] K. Hornik, M. Stinchcombe, H. White, et al., Multilayer feedforward networks are universal approximators, Neural Netw. 2 (5) (1989) 359–366.
[8] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Syst. 2 (4) (1989) 303–314.

[9] J.-X. Wang, J.-L. Wu, H. Xiao, Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data, Phys. Rev. Fluids 2 (3) (2017) 034603.
[10] J.-X. Wang, J.-L. Wu, J. Ling, G. Iaccarino, H. Xiao, Physics-informed machine learning for predictive turbulence modeling: toward a complete framework, in: Proceedings of the Summer Program, CTR, Stanford, 2016, p. 1.
[11] R. Roscher, B. Bohn, M.F. Duarte, J. Garcke, Explainable machine learning for scientific insights and discoveries, IEEE Access 8 (2020) 42200–42216.
[12] G.C. Peng, M. Alber, A.B. Tepole, W.R. Cannon, S. De, S. Dura-Bernal, K. Garikipati, G. Karniadakis, W.W. Lytton, P. Perdikaris, et al., Multiscale modeling meets machine learning: what can we learn?, Arch. Comput. Methods Eng. (2020) 1–21.
[13] R. Zhang, Y. Liu, H. Sun, Physics-guided convolutional neural network (PhyCNN) for data-driven seismic response modeling, arXiv preprint arXiv:1909.08118, 2019.
[14] L. Sun, J.-X. Wang, Physics-constrained Bayesian neural network for fluid flow reconstruction with sparse and noisy data, Theor. Appl. Mech. Lett. 10 (3) (2020) 161–169, https://doi.org/10.1016/j.taml.2020.01.031.
[15] N. Geneva, N. Zabaras, Modeling the dynamics of PDE systems with physics-constrained deep auto-regressive networks, J. Comput. Phys. 403 (2020) 109056, https://doi.org/10.1016/j.jcp.2019.109056, https://linkinghub.elsevier.com/retrieve/pii/S0021999119307612.
[16] H. Lee, I.S. Kang, Neural algorithm for solving differential equations, J. Comput. Phys. 91 (1) (1990) 110–131.
[17] I.E. Lagaris, A. Likas, D.I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE Trans. Neural Netw. 9 (5) (1998) 987–1000.
[18] I.E. Lagaris, A.C. Likas, D.G. Papageorgiou, Neural-network methods for boundary value problems with irregular boundaries, IEEE Trans. Neural Netw. 11 (5) (2000) 1041–1049.
[19] C. Michoski, M. Milosavljević, T. Oliver, D.R. Hatch, Solving differential equations using deep neural networks, Neurocomputing (2020).
[20] N. Wang, D. Zhang, H. Chang, H. Li, Deep learning of subsurface flow via theory-guided neural network, J. Hydrol. 584 (2020) 124700.
[21] M. Raissi, Z. Wang, M.S. Triantafyllou, G.E. Karniadakis, Deep learning of vortex-induced vibrations, J. Fluid Mech. 861 (2019) 119–137.
[22] M. Raissi, H. Babaee, P. Givi, Deep learning of turbulent scalar mixing, Phys. Rev. Fluids 4 (12) (2019) 124501.
[23] X. Jin, S. Cai, H. Li, G.E. Karniadakis, NSFnets (Navier-Stokes Flow nets): physics-informed neural networks for the incompressible Navier-Stokes equations, arXiv preprint, arXiv:2003.06496, 2020.
[24] M. Raissi, A. Yazdani, G.E. Karniadakis, Hidden fluid mechanics: learning velocity and pressure fields from flow visualizations, Science 367 (6481) (2020) 1026–1030.
[25] 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. Eng. 358 (2020) 112623.
[26] 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.
[27] Z. Fang, J. Zhan, Deep physical informed neural networks for metamaterial design, IEEE Access 8 (2019) 24506–24513.
[28] D. Liu, Y. Wang, Multi-fidelity physics-constrained neural network and its application in materials modeling, J. Mech. Des. 141 (12) (2019).
[29] Y. Chen, L. Lu, G.E. Karniadakis, L.D. Negro, Physics-informed neural networks for inverse problems in nano-optics and metamaterials, arXiv preprint arXiv:1912.01085, 2019.
[30] Q. Zheng, L. Zeng, G.E. Karniadakis, Physics-informed semantic inpainting: application to geostatistical modeling, arXiv preprint arXiv:1909.09459, 2019.
[31] X. Chen, J. Duan, G.E. Karniadakis, Learning and meta-learning of stochastic advection-diffusion-reaction systems from sparse measurements, arXiv preprint arXiv:1910.09098, 2019.
[32] R. Snaiki, T. Wu, Knowledge-enhanced deep learning for simulation of tropical cyclone boundary-layer winds, J. Wind Eng. Ind. Aerodyn. 194 (2019) 103983.
[33] Z. Mao, A.D. Jagtap, G.E. Karniadakis, Physics-informed neural networks for high-speed flows, Comput. Methods Appl. Mech. Eng. 360 (2020) 112789.
[34] X. Meng, G.E. Karniadakis, A composite neural network that learns from multi-fidelity data: application to function approximation and inverse PDE problems, J. Comput. Phys. 401 (2020) 109020.
[35] A.M. Tartakovsky, C.O. Marrero, P. Perdikaris, G.D. Tartakovsky, D. Barajas-Solano, Learning parameters and constitutive relationships with physics informed deep neural networks, arXiv preprint arXiv:1808.03398, 2018.
[36] J. Berg, K. Nyström, Data-driven discovery of PDEs in complex datasets, J. Comput. Phys. 384 (2019) 239–252.
[37] L. Lu, P. Jin, G.E. Karniadakis, DeepONet: learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators, arXiv preprint arXiv:1910.03193, 2019.
[38] L. Yang, X. Meng, G.E. Karniadakis, B-PINNs: bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data, arXiv preprint arXiv:2003.06097, 2020.
[39] Y. Yang, P. Perdikaris, Adversarial uncertainty quantification in physics-informed neural networks, J. Comput. Phys. 394 (2019) 136–152.
[40] D. Zhang, L. Lu, L. Guo, G.E. Karniadakis, Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems, J. Comput. Phys. 397 (2019) 108850.
[41] L. Lu, X. Meng, Z. Mao, G.E. Karniadakis, DeepXDE: a deep learning library for solving differential equations, arXiv preprint arXiv:1907.04502, 2019.
[42] J. Berg, K. Nyström, A unified deep artificial neural network approach to partial differential equations in complex geometries, Neurocomputing 317 (2018) 28–41.
[43] L. Yang, D. Zhang, G.E. Karniadakis, Physics-informed generative adversarial networks for stochastic differential equations, SIAM J. Sci. Comput. 42 (1) (2020) A292–A317.
[44] J. Han, A. Jentzen, E. Weinan, Solving high-dimensional partial differential equations using deep learning, Proc. Natl. Acad. Sci. 115 (34) (2018) 8505–8510.
[45] Y. Zang, G. Bao, X. Ye, H. Zhou, Weak adversarial networks for high-dimensional partial differential equations, J. Comput. Phys. (2020) 109409.
[46] E. Samaniego, C. Anitescu, S. Goswami, V. 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. Eng. 362 (2020) 112790.
[47] R. Khodayi-Mehr, M.M. Zavlanos, VarNet: variational neural networks for the solution of partial differential equations, arXiv preprint arXiv:1912.07443, 2019.
[48] E. Kharazmi, Z. Zhang, G. Karniadakis, Variational physics-informed neural networks for solving partial differential equations, arXiv preprint arXiv:1912.00873, 2019.
[49] K. Li, K. Tang, T. Wu, Q. Liao, D3M: a deep domain decomposition method for partial differential equations, IEEE Access (2019).
[50] E. Kharazmi, Z. Zhang, G.E. Karniadakis, hp-VPINNs: variational physics-informed neural networks with domain decomposition, arXiv preprint arXiv:2003.05385, 2020.
[51] V. Dwivedi, B. Srinivasan, Physics informed extreme learning machine (PIELM)–a rapid method for the numerical solution of partial differential equations, Neurocomputing (2019).
[52] R. Zhang, J. Hajjar, H. Sun, Machine learning approach for sequence clustering with applications to ground-motion selection, J. Eng. Mech. 146 (6) (2020) 04020040.

[53] S. Wang, Y. Teng, P. Perdikaris, Understanding and mitigating gradient pathologies in physics-informed neural networks, arXiv preprint arXiv:2001.04536, 2020.
[54] M.A. Nabian, H. Meidani, A deep learning solution approach for high-dimensional random differential equations, Probab. Eng. Mech. 57 (2019) 14–25.
[55] S. Karumuri, R. Tripathy, I. Bilionis, J. Panchal, Simulator-free solution of high-dimensional stochastic elliptic partial differential equations using deep neural networks, J. Comput. Phys. 404 (2020) 109120.
[56] C. Rao, Y. Liu, Three-dimensional convolutional neural network (3D-CNN) for heterogeneous material homogenization, arXiv preprint arXiv:2002.07600, 2020.
[57] B. Kim, V.C. Azevedo, N. Thuerey, T. Kim, M. Gross, B. Solenthaler, Deep fluids: a generative network for parameterized fluid simulations, in: Computer Graphics Forum, vol. 38, Wiley Online Library, 2019, pp. 59–70.
[58] R. Sharma, A.B. Farimani, J. Gomes, P. Eastman, V. Pande, Weakly-supervised deep learning of heat transport via physics informed loss, arXiv preprint arXiv:1807.11374, 2018.
[59] K.-i. Fukui, J. Tanaka, T. Tomita, M. Numao, Physics-guided neural network with model discrepancy based on upper troposphere wind prediction, in: 2019 18th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2019, pp. 414–419.
[60] A. Subramaniam, M.L. Wong, R.D. Borker, S. Nimmagadda, S.K. Lele, Turbulence enrichment using physics-informed generative adversarial networks, arXiv (2020) arXiv–2003.
[61] A.T. Mohan, N. Lubbers, D. Livescu, M. Chertkov, Embedding hard physical constraints in neural network coarse-graining of 3D turbulence, arXiv preprint arXiv:2002.00021, 2020.
[62] J.-L. Wu, K. Kashinath, A. Albert, D. Chirila, H. Xiao, et al., Enforcing statistical constraints in generative adversarial networks for modeling chaotic dynamical systems, J. Comput. Phys. 406 (2020) 109209.
[63] H. Yao, Y. Ren, Y. Liu, FEA-Net: a deep convolutional neural network with physics prior for efficient data driven PDE learning, in: AIAA Scitech 2019 Forum, 2019, p. 0680.
[64] J. He, J. Xu, MgNet: a unified framework of multigrid and convolutional neural network, Sci. China Math. 62 (7) (2019) 1331–1354.
[65] A. Joshi, V. Shah, S. Ghosal, B. Pokuri, S. Sarkar, B. Ganapathysubramanian, C. Hegde, Generative models for solving nonlinear partial differential equations, in: Second Workshop on Machine Learning and the Physical Sciences (NeurIPS 2019), Vancouver, Canada, 2019, https://ml4physicalsciences.github.io/2019/files/NeurIPS_ML4PS_2019_118.pdf.
[66] Z. Long, Y. Lu, X. Ma, B. Dong, PDE-net: learning PDEs from data, arXiv preprint arXiv:1710.09668, 2017.
[67] Z. Long, Y. Lu, B. Dong, PDE-Net 2.0: learning PDEs from data with a numeric-symbolic hybrid deep network, J. Comput. Phys. 399 (2019) 108925.
[68] G. Singh, S. Gupta, M. Lease, C.N. Dawson, TIME: a transparent, interpretable, model-adaptive and explainable neural network for dynamic physical processes, arXiv preprint arXiv:2003.02426, 2020.
[69] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, Universal differential equations for scientific machine learning, arXiv preprint arXiv:2001.04385, 2020.
[70] D.P. Kingma, J. Ba Adam, A method for stochastic optimization, arXiv preprint arXiv:1412.6980, 2014.
[71] J. Han, J. Tao, C. Wang, FlowNet: a deep learning framework for clustering and selection of streamlines and stream surfaces, IEEE Trans. Vis. Comput. Graph. (2018).
[72] J. Han, C. Wang, TSR-TVD: temporal super-resolution for time-varying data analysis and visualization, IEEE Trans. Vis. Comput. Graph. 26 (1) (2019) 205–215.
[73] J. Han, J. Tao, H. Zheng, H. Guo, D.Z. Chen, C. Wang, Flow field reduction via reconstructing vector data from 3-D streamlines using deep learning, IEEE Comput. Graph. Appl. 39 (4) (2019) 54–67.
[74] 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.
[75] X. Guo, W. Li, F. Iorio, Convolutional neural networks for steady flow approximation, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 481–490.
[76] J.F. Thompson, F.C. Thames, C.W. Mastin, Automatic numerical generation of body-fitted curvilinear coordinate system for field containing any number of arbitrary two-dimensional bodies, J. Comput. Phys. 15 (3) (1974) 299–319.
[77] John D. Anderson, Computational Fluid Dynamics: The Basics with Applications, McGraw-Hill Science, 1995.
[78] 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 Proceedings of IEEE Pacific Visualization Symposium (PacificVis), 2020, pp. 71–80, https://doi.ieeecomputersociety.org/10.1109/PacificVis48177.2020.8737.
[79] W. Shi, J. Caballero, F. Huszár, J. Totz, A.P. Aitken, R. Bishop, D. Rueckert, Z. Wang, Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1874–1883.
[80] H. Jasak, A. Jemcov, Z. Tukovic, et al., OpenFOAM: A C++ Library for complex physics simulations, in: International Workshop on Coupled Methods in Numerical Dynamics, vol. 1000, IUC, Dubrovnik, Croatia, 2007, pp. 1–20.
[81] R.H. Pletcher, J.C. Tannehill, D. Anderson, Computational Fluid Mechanics and Heat Transfer, CRC Press, 2012.
[82] C. Rhie, W.L. Chow, Numerical study of the turbulent flow past an airfoil with trailing edge separation, AIAA J. 21 (11) (1983) 1525–1532.
[83] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, https://openreview.net/forum?id=BJJsrmfCZ, 2017.
[84] J. Miriel, L. Serres, A. Trombe, Radiant ceiling panel heating–cooling systems: experimental and simulated study of the performances, thermal comfort and energy consumptions, Appl. Therm. Eng. 22 (16) (2002) 1861–1873.
[85] I. Chowdhury, R. Prasher, K. Lofgreen, G. Chrysler, S. Narasimhan, R. Mahajan, D. Koester, R. Alley, R. Venkatasubramanian, On-chip cooling by superlattice-based thin-film thermoelectrics, Nat. Nanotechnol. 4 (4) (2009) 235.
[86] S. Mo, N. Zabaras, X. Shi, J. Wu, Integration of adversarial autoencoders with residual dense convolutional networks for estimation of non-gaussian hydraulic conductivities, Water Resour. Res. 56 (2) (2020) e2019WR026082.
[87] J. Xu, K. Duraisamy, Multi-level convolutional autoencoder networks for parametric prediction of spatio-temporal dynamics, arXiv preprint arXiv:1912.11114, 2019.
[88] E.H. Spanier, Algebraic Topology, Springer Science & Business Media, 1989.
[89] W. Rudin, Real and Complex Analysis, Tata McGraw-Hill Education, 2006.
[90] S. Bozzi, U. Morbiducci, D. Gallo, R. Ponzini, G. Rizzo, C. Bignardi, G. Passoni, Uncertainty propagation of phase contrast-MRI derived inlet boundary conditions in computational hemodynamics models of thoracic aorta, Comput. Methods Biomech. Biomed. Eng. 20 (10) (2017) 1104–1112.
[91] H. Gao, X. Zhu, J.-X. Wang, A bi-fidelity surrogate modeling approach for uncertainty propagation in three-dimensional hemodynamic simulations, Comput. Methods Appl. Mech. Eng. 366 (2020) 113047.
[92] C.M. Fleeter, G. Geraci, D.E. Schiavazzi, A.M. Kahn, A.L. Marsden, Multilevel and multifidelity uncertainty quantification for cardiovascular hemodynamics, arXiv preprint arXiv:1908.04875, 2019.
[93] M.E. Tipping, C.M. Bishop, Probabilistic principal component analysis, J. R. Stat. Soc., Ser. B, Stat. Methodol. 61 (3) (1999) 611–622.
[94] B.E. Launder, D.B. Spalding, The numerical computation of turbulent flows, in: Numerical Prediction of Flow, Heat Transfer, Turbulence and Combustion, Elsevier, 1983, pp. 96–116.