Oldenburg *et al.*
Adv. Model. and Simul. in Eng. Sci. (2022) 9:8
https://doi.org/10.1186/s40323-022-00221-z

Advanced Modeling and Simulation in Engineering Sciences

RESEARCH ARTICLE Open Access

Geometry aware physics informed neural network surrogate for solving Navier–Stokes equation (GAPINN)

Jan Oldenburg[1*], Finja Borowski[1], Alper Öner[2], Klaus-Peter Schmitz[1] and Michael Stiehm[1]

*Correspondence:
jan.oldenburg@uni-rostock.de

[1] Institute for ImplantTechnology and Biomaterials e.V., Rostock, Germany
[2] Heart Center/Department of Cardiology, Rostock University Medical Center, Rostock, Germany

Abstract

Many real world problems involve fluid flow phenomena, typically be described by the Navier–Stokes equations. The Navier–Stokes equations are partial differential equations (PDEs) with highly nonlinear properties. Currently mostly used methods solve this differential equation by discretizing geometries. In the field of fluid mechanics the finite volume method (FVM) is widely used for numerical flow simulation, so-called computational fluid dynamics (CFD). Due to high computational costs and cumbersome generation of the discretization they are not widely used in real time applications. Our presented work focuses on advancing PDE-constrained deep learning frameworks for more real-world applications with irregular geometries without parameterization. We present a Deep Neural Network framework that generate surrogates for non-geometric boundaries by data free solely physics driven training, by minimizing the residuals of the governing PDEs (i.e., conservation laws) so that no computationally expensive CFD simulation data is needed. We named this method geometry aware physics informed neural network—GAPINN. The framework involves three network types. The first network reduces the dimensions of the irregular geometries to a latent representation. In this work we used a Variational-Auto-Encoder (VAE) for this task. We proposed the concept of using this latent representation in combination with spatial coordinates as input for PINNs. Using PINNs we showed that it is possible to train a surrogate model purely driven on the reduction of the residuals of the underlying PDE for irregular non-parametric geometries. Furthermore, we showed the way of designing a boundary constraining network (BCN) to hardly enforce boundary conditions during training of the PINN. We evaluated this concept on test cases in the fields of biofluidmechanics. The experiments comprise laminar flow (Re = 500) in irregular shaped vessels. The main highlight of the presented GAPINN is the use of PINNs on irregular non-parameterized geometries. Despite that we showed the usage of this framework for Navier Stokes equations, it should be feasible to adapt this framework for other problems described by PDEs.

Keywords: Physics informed neural network, Parametric, Non-parametric, Fluid flow, Stenosis

Introduction

Many real world problems involve fluid flow phenomena, for instance, in aerospace, maritime engineering and biomedical engineering. Such problems mathematically can be described by the Navier–Stokes equations as a basis for computational simulation. The Navier–Stokes equations are partial differential equations (PDE) with highly nonlinear properties. Currently used computational methods solve these differential equations using numerical methods with discretization techniques like finite volume method (FVM) mostly used in computational fluid dynamics (CFD). Depending on the complexity of the problem involved, these methods tend to be very computationally expensive and require a high level of methodological understanding in the generation of the discretization (meshing). These factors inhibit an intensive utilization in time-relevant applications such as surgery planning or real-time simulation during the operation in the cardiovascular field.

Surrogate models, which approximate the output in relation to the input, could increase the use in these fields. Quantities like the fluid velocity and pressure could be captured very fast by these methods. Three different types of surrogate modeling techniques can be distinguished: Reduced Order Models (ROM), data fit models and Deep Neural Network (DNN) based models.

ROM's are used to reduce the order of the problem in an unsupervised manner, among others Proper Orthogonal Decomposition (POD) and Principal Component Analysis (PCA) are the most popular. However, these methods appear to be limited by stability and robustness difficulties [1]. Moreover, they are intrusive in the code and limited in speedup potential with high nonlinearities, as summarized by Sun et al. [2, 3].

Data fit models create a fit between input and output based on simulations. The most popular methods are polynomial basis, radial basis functions, Gaussian process and stochastic polynomial chaos expansion. These methods mostly do not require any change in the simulation solver.

Deep learning-based surrogate models promise the ability to handle strong nondimensional problems by their basic structure as it is required for the Navier–Stokes equations. These methods are heavily developed and pushed in the fields of computer vision. Traditional deep learning methods use a large data set and the information contained in the data, which is referred to as data-driven learning. In the case of a surrogate model as a specific problem, many numerical simulations would be required to train a neural network as a function approximator for PDEs. In particular in the field of CFD, the generation of training data would eliminate the performance enhancement of DNN. The enormous effort involved in generating the training data initiated in the 1990s the further development of applying neural networks directly as a solver for differential equations [4, 5]. This concept takes into account that the modelled physics are known a-priori and due to the known governing equations constrains could be formulated or drive the training of the neural network. With increasing computational power, this approach became more and more feasible in the last years, leading to the development of a new type of neural networks, the so-called physics informed neural networks (PINNs), which were first launched by Raissi et al. [6]. They solved one-dimensional (1D) PDEs such as viscous Burger's equation, and PDE-constrained inverse problems by using only few amounts of training data. In order to perform surrogate modeling, a working

group used the PDE-bounded DNN for uncertainty propagation in steady-state heat equations [7]. Furthermore, Gao et al. proposed an approach of surrogate modeling for Navier–Stokes equations for irregular geometries by means of coordinate transformation to rectangular grids, which allowed an effective use of convolutional neural networks and efficient derivative computation (Finite Difference) due to rectangular grids [8]. This approach is limited by the regular meshing method even for irregular geometries. Most techniques showed the possibilities to solve PDE without the use of training data only on canonical problems with rectangular domains. Sun et al. showed that it is feasible to train PINNs for parameterized geometries without simulation or experimental data, by adding the parametric variable as input to the network for surrogate modeling of incompressible steady flows [2]. Motivated from this work we solved the underlying Navier–Stokes equations for nonparametric geometries. Analogue to Sun et al. we trained surrogates using data free solely physics driven methods, by minimizing the residuals of the governing PDEs (i.e., conservation laws). As a result no computationally expensive CFD simulation data is needed.

In this paper we introduce a framework to create a fluid flow surrogate model based on irregular geometry with no parameterization using PINNs without the need for training data, which to the best of the author's knowledge has not been done so far. We named our method geometry aware physics informed neural network—GAPINN.

Our current work focuses on advancing the PDE-constrained deep learning framework for more real-world applications with irregular geometric shapes. The paper is organized as follows. First the physical background and its mathematical description is presented. In the next section the architecture of the deep learning algorithm proposed here is explained. In the following chapters the numerical experiments will be described and are followed by the results obtained from the proposed GAPINN framework in comparence with CFD-Simulations and vanilla PINN. Finally, conclusions are drawn in the last chapter.

Network architecture

Physical background

In this work we will focus on a biomedical problem, the flow inside blood vessels with stenosis leading to pathological blood flow. These vascular stenoses could lead to major health complaints, especially in the field of cardiology or neurology. The anatomy of stenosed vessels is highly individual and can be considered as non-parametric geometries. The blood can be described with the conservation equations of fluid mechanics, in particular with the Navier–Stokes equations. We want to point out that the proposed workflow could also be adapted to problems arising from other fields in continuum mechanics such as structural mechanics or heat and mass transfer.

The Navier–Stokes equations can be written in its residual formulation like below:

$$\mathcal{F}(\mathbf{U},\mathrm{p}) = 0 := \begin{cases} \nabla \cdot \mathbf{U} = 0, & \mathbf{X}, t \in \boldsymbol{\Omega}_{f,t}\boldsymbol{\Theta} \in \mathbb{R}^{\boldsymbol{d}}, \\ \frac{\partial \mathbf{U}}{\partial t} + (\mathbf{U}\cdot\nabla)\mathbf{U} + \frac{1}{\rho}\nabla \mathrm{p} - \upsilon\nabla^2\mathbf{U} + \mathbf{b}f = 0, & \mathbf{X}, t \in \boldsymbol{\Omega}_{f,t}\boldsymbol{\Theta} \in \mathbb{R}^{\boldsymbol{d}} \end{cases} \tag{1}$$

With $\mathbf{X}$ being spatial coordinates and t the time. The PDE is constraint due to $\boldsymbol{\Theta}$ describing parameters like boundary and initial conditions and fluid properties. In the formula

$\mathbf{U}(t, \mathbf{X}, \boldsymbol{\Theta})$ is the fluid velocity, with its components u, v, w, in three dimensions and $p(t, \mathbf{X}, \boldsymbol{\Theta})$ is the corresponding pressure. ρ and υ are the density and the kinematic viscosity of the fluid respectively. The method has been tested for 2-dimensional steady state cases.

GAPINN framework

The GAPINN framework consists of three separate networks, see Fig. 1: (1) as one of the most important parts, to solve for varying non-parametric geometries, a Shape Encoding Network (SEN); (2) a Physics Informed Neural Network (PINN) in order to solve the differential equation of a given fluid mechanical problem; (3) and a Boundary Constrain Network (BCN) to constrain the boundary and initial conditions for each given non-parametric geometric boundaries.

The shape of a fluid domain, where the Navier–Stokes equations to be solved, was defined by spatial positions $\mathbf{X}_{j,(i,b)}$ whereas the subscript i denotes locations of the internal field of the fluid domain, b denotes locations on the boundary and j indicates different cases defined by a set of varying non-parametric geometries of fluid domains (made up from i and b). The framework outputs velocities $\mathbf{U}_{j,(i,b)}$ and pressure $p_{j,(i,b)}$ for this spatial locations. In this framework the SEN was used to reduce the dimensions of the given non-parametric geometric boundaries. This latent representation was than concatenated with the spatial positions and serves as input for the PINN and the BCN, see Fig. 1.

The workflow for training the GAPINN framework was that way that the SEN was trained first, generating a latent representation. The BCN was trained second and the PINN trained last, given the information's from the SEN and BCN as depicted in Fig. 1. After the training of SEN and BCN was performed, no more adjustment of weights and biases were done for these networks.

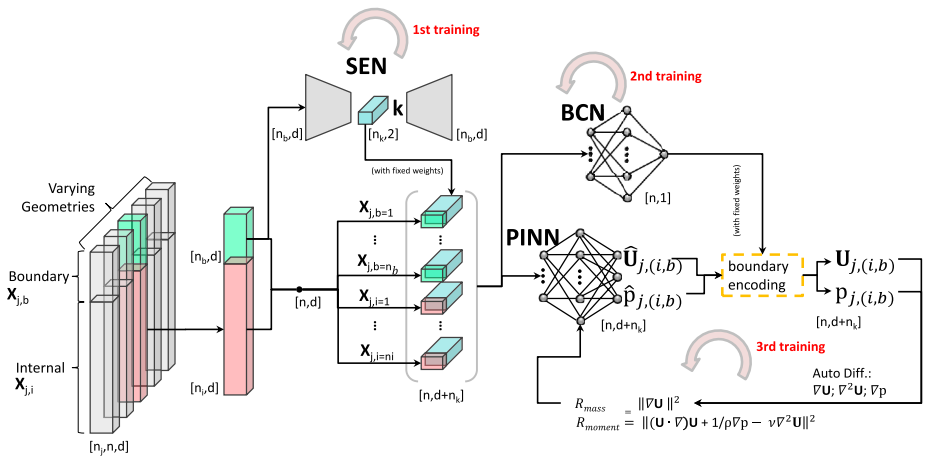

Fig. 1 Schematic description of the architecture of the proposed DNN Method (GAPINN) to generate surrogates of PDEs with irregular non-parameterized geometries using PINNs. The network consists of three subnetworks which are trained separately. The SEN is a Variational-Auto-Encoder type reducing the geometry dimensions to a latent vector k. PINN takes k and spatial positions to solve the PDE and building the surrogate and BCN, by also taking spatial information and k, helps constraining boundary conditions in the PINN. Dimensions at each operation are noted in brackets

We first describe the SEN more in detail to help the reader to understand how different fluid domain geometries can be interpreted by a PINN. Moreover, how this facilitates the development of a surrogate model that is able to solve fluid mechanical partial differential equations for various non-parametric geometries without the need of training data.

Shape encoding network (SEN)

As input we assumed a non-parametric but well-defined fluid domain. We aimed to get a latent representation of each geometric shape, by using the technique of Variational-Auto-Encoder (VAE) [9]. VAE are a common technique in the field of computer vision to reduce high dimensional information to a lower dimensional representation in an unsupervised learning process. A VAE is built from two main components namely the encoder, which actually reduces the dimensions and the decoder, which reconstructs the input from a lower dimensional representation.

A reason why we recommend VAE instead of Auto-Encoder (AE) is that we had found poor validation performance for AE. In order to obtain a feasible latent representation, in terms of interpolation capabilities for geometric similar shapes, we used a VAE with a regularization term which fits the latent vector to a known distribution.

To ensure that the low dimensional representation is robust against permutation we had chosen a PointNet-like architecture [10]. The main concept of this type of network is the usage of Multilayer Perceptron neural networks with shared weights and a globally acting "symmetrical" pooling function in order to construct the lower dimensional representation from a set of points. For this study we used one dimensional convolution operators with a kernel size of 1 and stride of 1, which in principle is an implementation of a multilayer perceptron with shared weights. A schematic depiction of the encoder network used in the experiments is shown in Fig. 2. The input of the network was a subset of points representing the boundary ($\mathbf{X}_{j,b}$ with size n_b). Followed by four 1d convolution layers which were increasing the spatial dimensional axis from d to 512 channels (Conv1:128, Conv2:128, Conv3:256, Conv4:512). Subsequently, a max pooling operation was used in a channel wise manner to aggregate information shared by all points, resulting in an array with 512 channels. This was followed by two fully connected neural networks (FCNN) with one hidden layer. The FCNN reduced the pooled vector to the desired lower dimensional representation size n_k of the latent vector $\mathbf{k}$. The pooled vector was split into prediction of mean and variance. This describes the

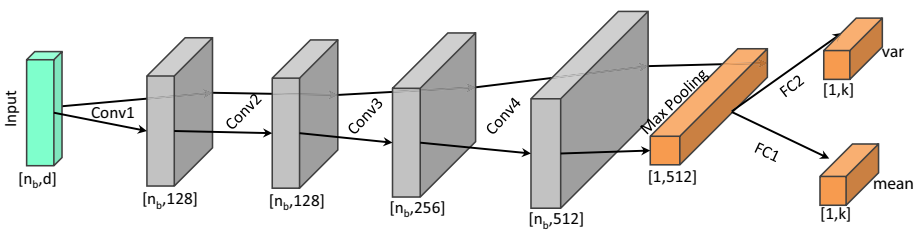

Fig. 2 Schematic visualization of the use encoder architecture used here, build from input given into 4 × 1d Convolutions, channel-wise max pooling and two fully connected networks with three layers each (layers: 512, 256, k). The dimension after every operation is given in brackets

posterior distribution predicted by the encoder. One can sub sequentially follow next steps by using either both vectors (mean vector and variance) or only the mean vector. For our cases applying the mean vectors reaches proper results.

During training the encoder was constrained to learn a given distribution (in our experiments this was a Gaussian distribution). From the output of the encoder the decoder reconstructs the input point cloud. The decoder consisted of FCNN with three hidden layers. After training of the SEN the decoder was no longer needed and only the encoder module was of importance. The reconstruction and the real input were compared using the chamfers distance function. During training the summation of the Kullback–Leibler divergence [11], between the approximate and true prior distribution, ($\mathcal{L}_{KL}$) weighted with β, and the chamfer distance loss ($\mathcal{L}_{REC}$) [12], here called $\mathcal{L}_{VAE}$, were minimized.

$$\mathcal{L}_{VAE} = \mathcal{L}_{REC} - \beta\mathcal{L}_{KL} \tag{2}$$

With $\mathcal{L}_{REC}$ described by following formula:

$$\mathcal{L}_{REC} = \sum_{S\in X_b} \min_{\hat{S}\in\widehat{X_b}} \left\|S-\hat{S}\right\|_2^2 + \sum_{\hat{S}\in\widehat{X_b}} \min_{S\in X_b} \left\|S-\hat{S}\right\|_2^2 \tag{3}$$

With S being points from $\mathbf{X}_b$ and their reconstruction $\hat{S}$ from $\widehat{\mathbf{X}_b}$ due to the decoder. That means that $\mathcal{L}_{REC}$ describes the distance between a point from the subset $\mathbf{X}_\mathrm{b}$ and its nearest neighbor in the reconstructed $\widehat{\mathbf{X}_b}$; and the other way around.

Physics informed neural network (PINN)

The PINN was of fully connected feedforward type (FCNN). Input to the PINN were on the one hand spatial positions $\mathbf{X}_{\mathrm{j,(i,b)}}$ and on the other hand the encoded information $\mathbf{k}$ for each of the geometry domains. These two inputs were concatenated along the dimensional axis of $\mathbf{X}$ before inserted into the network. The dimension of the input is $[\mathrm{n,d+n_k}]$, with n being the number of points from a subset of $\mathbf{X}_{\mathrm{j,i}}$ and $\mathbf{X}_{\mathrm{j,b}}$, d the number of spatial dimensions and $\mathrm{n_k}$ being the dimension of the latent vector $\mathbf{k}$. The network maps the inputs to the velocity $\mathbf{U}_{\mathrm{j,(i,b)}}$ and pressure $\mathrm{p}_{\mathrm{j,(i,b)}}$. For more general information on feed forward neural networks we recommend [13] or for more specific aspects regarding PINN's [6, 14]. Layers between the input and output were referred to as hidden layers. As activation function after each hidden layer we used a sigmoid linear unit function [15], with the output coming from logistic sigmoid of the input multiplied with the input itself.

During the training the weights ($\mathbf{W}$) and biases ($\mathbf{b}$) were adapted so that the loss will be minimal, see Eq. 5. The loss function is describing the physics, in form of a differential equation in its residual formulation. In the case of steady state Navier–Stokes equations the loss is described due to the conservation of mass and momentum in the fluid domain, see Eq. 4.

$$\mathcal{L}_{phy}(\mathbf{W},\mathbf{b}) = \left\|\nabla\mathbf{U}^2\right\| + \left\|(\mathbf{U}\cdot\nabla)\mathbf{U} + \frac{1}{\rho}\nabla p - \vartheta\nabla^2\mathbf{U}\right\|^2 \tag{4}$$

$$W^* = \arg\min_{W} \mathcal{L}_{phy}(\mathbf{W}, \mathbf{b}) \tag{5}$$

For this loss function several first and second order derivatives of the output (**U**,p) with respect to spatial coordinates **X** were needed. These calculations were performed be means of the concept of automatic differentiation (AD). AD is a common technique used in the field of machine-learning, mainly to get gradients of the network with respect to the weights and biases. This technique relies on the concept of the calculation of derivatives inside a computational graph and is implemented in most state of the art deep-learning libraries such as TensorFlow, PyTorch or Theano; here we worked with PyTorch. For solving the optimization problem (Eq. 5), we used the Adam algorithm [16].

Boundary enforcing

Boundary conditions can be imposed mainly in two ways. First, by adding an extra penalty loss term $\mathcal{L}_{soft}$ to Eq. 4, which affects the PINN to learn conditions on the boundary, by minimizing $\mathcal{L}(\mathbf{W}, \mathbf{b})$ during training, with W and b being weights and biases of the neural network see Eq. 6. Sun et al. showed several major drawbacks of this so-called soft boundary imposing. As mentioned by Sun et al. this approach is not feasible to ensure the accurate definition of initial and boundary conditions due to its implicit manner. Furthermore, the optimization performance could depend on relative importance of the boundary loss term and PDE loss term [2]. This could be addressed by weighting the terms, but also a-prior weighting will mostly not be known.

$$\mathcal{L}(\mathbf{W}, \mathbf{b}) = \mathcal{L}_{phy} + \mathcal{L}_{soft} \tag{6}$$

The second approach of imposing boundary conditions is to hard enforce these in the neural network. This approach can be implemented by constructing a set of functions that are taking the outputs of the PINN $\hat{\mathbf{U}}_{\mathrm{b,i}}$ and $\hat{\mathrm{p}}_{\mathrm{b,i}}$ as input and, while automatically satisfying boundary conditions, computing the output $\mathbf{U}_{\mathrm{b,i}}$ and $\mathrm{p_b}$, see Eq. 7. It is beneficial that these functions are "smooth" on $\mathbf{X}_{\mathrm{j,(i,b)}}$. For most boundary problems we can use functions that indicate where a boundary condition should be constraint, here referred to as $B\left(\hat{\mathrm{U}}_{b,i}, \hat{\mathrm{p}}_{b,i}\right)$. In addition to that functions for applying the correct value on the boundary, here expressed as $\boldsymbol{C}\left(\mathbf{X}_{\mathrm{j,(i,b)}}\right)$, need to be set. This concept is motivated according to [17].

$$\begin{matrix}\mathbf{U}_{j(b,i)} \\ \mathrm{p}_{j(b,i)}\end{matrix} = B\left(\hat{\boldsymbol{U}}_{\mathrm{j,(i,b)}}, \hat{\mathrm{p}}_{\mathrm{j,(i,b)}}, \mathbf{X}_{\mathrm{j,(i,b)}}\right) + \boldsymbol{C}\left(\mathbf{X}_{\mathrm{j,(i,b)}}\right) \tag{7}$$

For hard constrain boundary conditions for common fluid mechanic domains we propose to use a pre-trained neural network which is predicting the minimal Euclidean distance of each $\mathbf{X}_{\mathrm{j,i}}$ to the walls which are considered fixed, here indicated as the function $\mathrm{BCN}\left(\mathbf{X}_{\mathrm{j,(i,b)}}\right)$. We used a simple FCNN for this task. The advantage of this approach lies in the capability of tracking gradients in order to use the concept of automatic differentiation during neural network training. Therefore we did not compute the distance on the fly while training.

The network takes as input the spatial positions $\mathbf{X}_{\mathrm{j,(i,b)}}$ and the latent vector $\mathbf{k}$. The prediction of BCN was compared to pre-computed Euclidean distances of the spatial positions to the boundaries with fixed zero velocity by using mean squared error. Reduction of the mean squared error was done in the training process of the BCN, by adapting the weights of the neural network. The exact form of the functions B and C are highly dependent on the investigated fluid mechanical problem to be solved. For detailed description on how to construct the boundary functions for a specific problem we refer to the following experiment chapter.

Experiments

In the experiments we focused on a biomedical blood flow problem—the flow inside vessels with irregular geometries including stenosis and/or aneurysm sections in steady state condition within a two-dimensional fluid domain. The physics of blood flow (Newtonian assumption) can be described using the following formulation of the incompressible Navier–Stokes equations.

$$\mathcal{F}(\mathbf{U},\mathrm{p})=0:=\begin{cases}\nabla\cdot\mathbf{U}=0\\(\mathbf{U}\cdot\nabla)\mathbf{U}+\frac{1}{\rho}\nabla\mathrm{p}-\upsilon\nabla^2\mathbf{U}=0,\end{cases}\tag{8}$$

We generated 1000 geometries, by fitting a spline function through randomly generated marking points, building an irregular vessel wall, see Fig. 3. On each of the 4 boundaries we sampled 100 points ($\mathbf{n}_{\mathrm{j=1000,}}$ $\mathbf{n}_{\mathrm{b=400}}$). The spatial dimensions were normalized to the inlet size (D). The inlet was positioned at $\mathrm{x}=0$. The Reynolds number was set to $\mathrm{Re}=500$. The velocities were normalized according to the mean velocity at the inlet at which a parabolic velocity flow profile in x-direction was applied. On the walls we defined $\mathbf{U}_{\mathrm{wall}}=0$ m/s. The pressure at the outlet was fixed to zero. The predicted pressure was normalized to pressure difference (p*) according to flow through a straight channel without stenosis and aneurysm. Neumann boundary conditions were implemented as soft boundary condition in the neural network and were imposed as zero gradient pressure at inlet and walls and zero gradient velocity at the outlet. For simplification, it is assumed that the blood is a Newtonian fluid.

We used the trained BCN to help constrain velocity boundary conditions, see Eq. 7, by applying the following formulation for the functions $B\left(\hat{\boldsymbol{U}}_{\mathrm{j,(i,b)}},\hat{\mathrm{p}}_{\mathrm{j,(i,b)}},\mathbf{X}_{\mathrm{j,(i,b)}}\right)$:

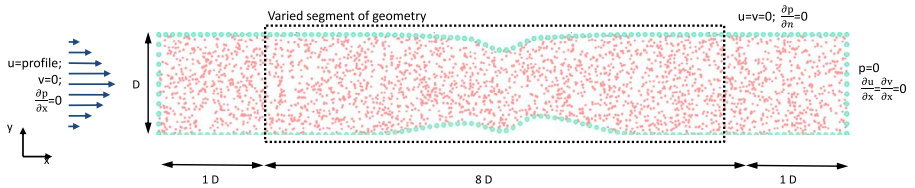

Fig. 3 Schematic of irregular geometric vessel with a fixed set of boundary and fluid domain points. With Dirichlet and Neumann boundary conditions: parabolic velocity profile at the inlet, zero velocity at walls and zero pressure at the outlet as well as zero gradient in reference to normal vectors (*n*) of geometry for pressure on inlet and vessel walls and velocity at the outlet. Red points are indicating internal points and green points are indicating points on the boundary

$$B\left(\hat{\boldsymbol{U}}_{\mathrm{j,(i,b)}}, \hat{\mathrm{p}}_{\mathrm{j,(i,b)}}, \mathbf{X}_{\mathrm{j,(i,b)}}\right) = \begin{cases} \text{for } \mathbf{U}{:}BCN\left(\mathbf{X}_{\mathrm{j,(i,b)}}\right)\hat{\boldsymbol{U}}_{\mathrm{j,(i,b)}}\mathrm{x}_{\mathrm{j,(i,b)}} \\ \text{for } p : \hat{\mathrm{p}}_{\mathrm{j,(i,b)}}\left(\mathrm{x}_{\mathrm{j,(i,b)}} - 10\mathrm{D}\right) \end{cases} \tag{9}$$

With function C, in Eq. 7, enforcing a parabolic velocity profile at the inlet, with R = D/2, and zero pressure at the outlet**.**

$$\boldsymbol{C}\left(\mathbf{X}_{\mathrm{j,(i,b)}}\right) = \begin{cases} \text{for u:} \left(1 - \left(\frac{\left(BCN\left(\mathbf{X}_{\mathrm{j,(i,b)}}\right) - R\right)}{R}\right)^2\right) * \mathrm{u}_{\max} \\ \text{for v:0} \\ \text{for p:0} \end{cases} \tag{10}$$

For training of the SEN we only included $\mathbf{X}_{\mathrm{j,b}}$ into the SEN. As already mentioned, the weights of the SEN were fixed when further used for training of the BCN and PINN. Training the BCN we used the boundary points as input for the SEN to generate the latent representation (here: $\mathrm{n_k} = 60$), and used all points in the training set to train the BCN. The prediction of BCN was compared to precomputed Euclidean distances of $\mathbf{X}_{\mathrm{j,i}}$ to the boundaries $\mathbf{X}_{\mathrm{j,b}}$ with fixed zero velocity by using mean squared error.

Training was done on single GPU (NVIDIA Quadro RTX 5000). Network and hyper parameters used in the experiments are listed belohnt, see Table 1.

To validate the prediction performance of the data-free trained GAPINN we performed numerical simulations on randomly generated vessels by means computational fluid dynamics (CFD) using OpenFOAM 9 [18] for comparison with geometries used in training and ANSYS 18.0 Fluent with vessel geometries not included in training. The mesh consisted of 150,000 hexahedron elements. This validation set of vessels was not included in the training process.

To show the advantage of the developed framework, we also performed experiments on exactly the same dataset and training parameters while using Vanilla PINN framework. The Vanilla PINN had the same number of hidden layers and neurons in the hidden layers as the PINN used in the GAPINN approach. Furthermore, in addition to the hard boundary constraint strategy as presented here, we applied soft constraint strategy for both GAPINN and classical PINN.

Table 1 Network and hyper parameter settings used in the experiments

	SEN	**PINN**	**BCN**
Network parameter			
Layers	4 × 1d Convolutions (feature size: 128, 128, 256, 512) FCN (3 hidden layer with 256 neurons, $\mathrm{n_k} = 60$); $\beta = 0.001$	2 hidden layer (each 256 neurons)	2 hidden layer (each 25 neurons)
Training parameter			
Batch size	50	10	10
Optimizer	Adam [16] Betas = (0.9, 0.999) No weight decay No amsgrad	Adam Betas = (0.9, 0.999) No weight decay No amsgrad	Adam Betas = (0.9, 0.999) No weight decay No amsgrad
Learning rate	1e−3	1e−3	1e−3
Epochs	650	1300	3000

Results and discussion

Here we present results to evaluate the performance of the proposed DNN framework named GAPINN. We performed experiments on 2D steady and laminar flow inside vessels with irregular geometries referring to biomedical blood flow problem. We trained the three networks (SEN, BCN, PINN) on 1000 different geometries, referred to as training set. For evaluation of GAPINN we generated vessel that are not included in training, referred to as validation set. The output from the GAPINN were velocity u and v as well as pressure p. For comparing, the output data of GAPINN and CFD were interpolated on the same grid using linear interpolation.

Figure 4 shows the comparison between GAPINN predictions and CFD solutions on samples from the training and from the validation set with velocity in stream-wise component u on the left and span-wise velocity component v on the right.

It can be seen that the main flow phenomena predictions using GAPINN agree reasonable well with the CFD solutions. For instance, with decreasing diameter, the velocity u increases, this phenomena is shown using conventional CFD simulation as well as GAPINN. The mean squared error (mse) of comparison between CFD and GAPINN on the shown validation samples are $\mathrm{mse_u} = 3.61\mathrm{e}^{-3}$ $\mathrm{mse_v} = 2.30\mathrm{e}^{-4}$. We think that an intensive study on architecture and hyper parameter optimization could increase the prediction performance but that this is out of scope of the current work. The experiments shown here are for proof of concept only.

Figure 5 shows the comparison between GAPINN predictions and CFD solutions of pressure distribution on samples from the training and validation set.

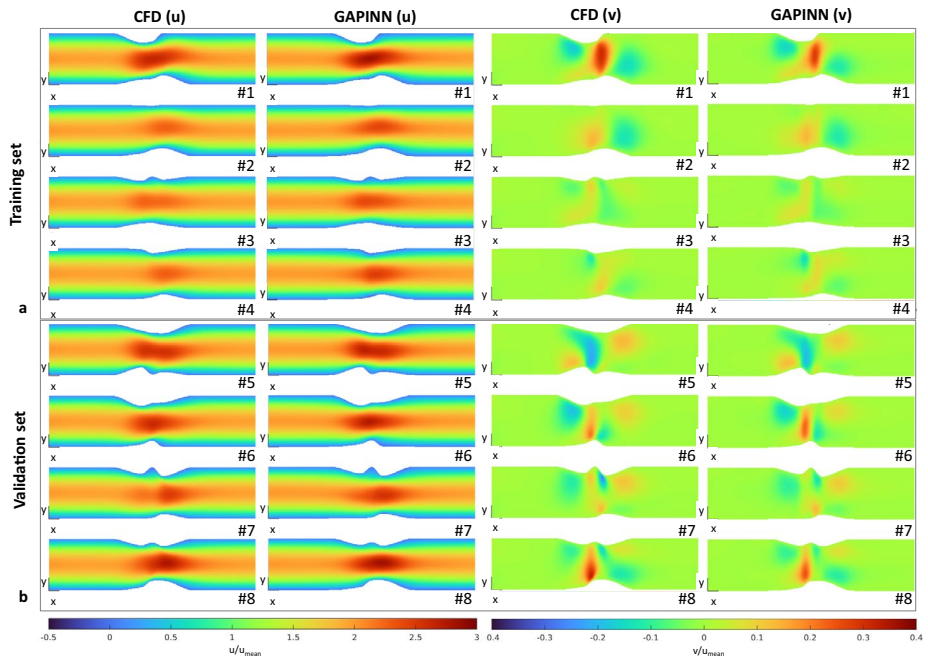

Fig. 4 Comparison between GAPINN predictions and CFD solutions eight randomly picked samples (#1–#8) of irregular vessels from training (**a**) and validation (**b**) with velocity in streamwise component u on the left and spanwise velocity component v on the right

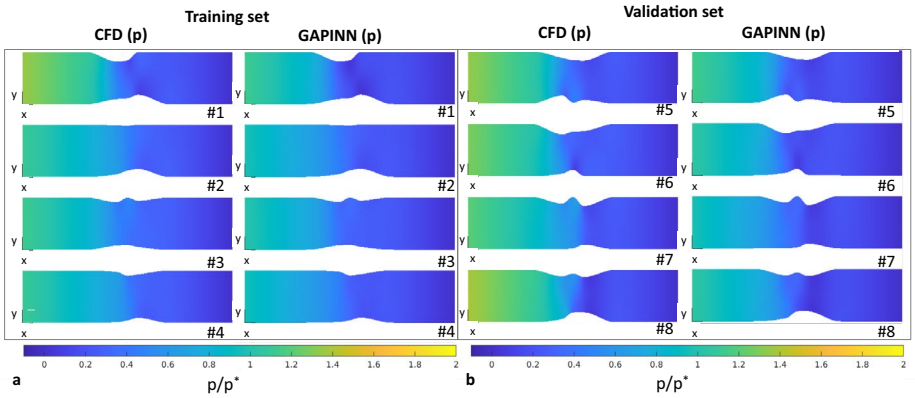

Fig. 5 Comparison between GAPINN predictions and CFD solutions on eight randomly picked samples (#1–#8) of irregular vessels from training (**a**) and validation (**b**) with velocity in streamwise component u on the left and spanwise velocity component v on the right

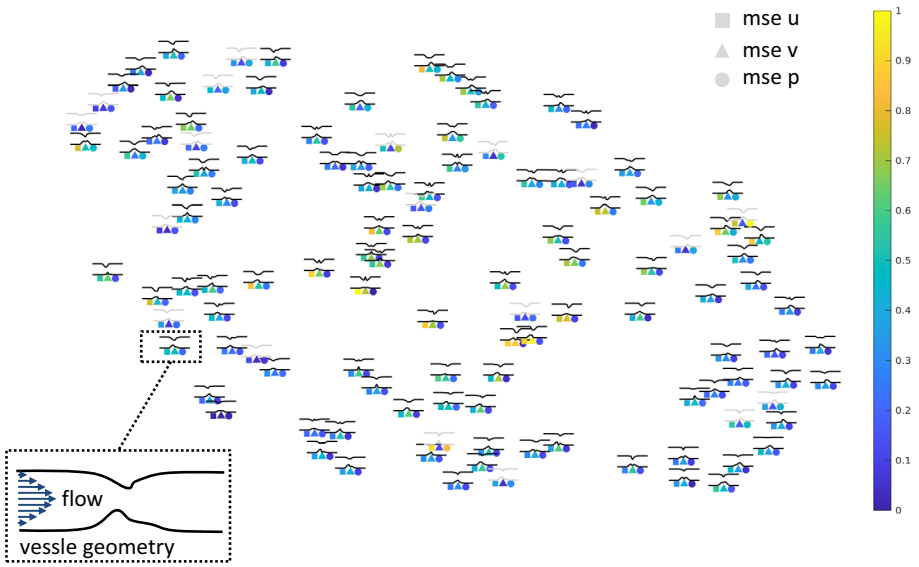

Fig. 6 two dimensional embedding of shape encoded latent vector k by means of t-SNE technique for visualization. Gray shapes indicating shapes from validation set and black shapes indicating shapes from training set. Under each shape is the mean squared error (mse) with respect to CFD-Simulations plotted (squares: mse u, triangle: mse v and circles: mse p). The color is linearly scaled between 0 and 1, the corresponding maximal values are mse_{max} u = 0.15, mse_{max} v = 0.07 and mse_{max} p = 0.27

The GAPINN predict the pressure distribution in qualitative manner quite well. An discrepancy can be observed in the quantitative value. The mean squared error on the validation set is $\text{mse}_{\text{p}} = 1.6\text{e}{-2}$.

To get an overview of the performance of the GAPINN on the encoded geometries, we use t-SNE [19] to embed the latent vector **k** for visualization of our validation (20 CFD simulations) and training (100 CFD simulations) set into a 2D space. Figure 6 shows the embedding space. It can be seen that similar vessel shapes are clustered. Furthermore,

the varieties of vessel geometries are widely distributed over the embedded space which is due to the use of VAE. Plotted under each shape is the mean squared error with respect to CFD-Simulations (squares: mse u, triangle: mse v and circles: mse p). The color is linearly scaled between 0 and 1, the corresponding maximal values are $mse_{max\ u} = 0.15$, $mse_{max\ v} = 0.07$ and $mse_{max\ p} = 0.27$.

The following Fig. 7 shows the comparison of the loss during the training process between the conventional PINN framework and the proposed GAPINN framework to create surrogate models that resolve the Navier–Stokes equations on non-parameterized geometries without the use of training data. Both soft and hard boundary constraint methods were used for this experiment. The error on the training data set during training for the Vanilla PINN framework is several orders of magnitude higher than for the proposed GAPINN framework. The Vanilla PINN framework fails to solve the Navier–Stokes equations on varying geometries in non-parameterized form. We also see advantage in using hard boundary constraint over soft boundary constraint for the GAPINN framework. It should be noted that the soft boundary method is very sensitive to the weighting of the loss terms for boundary constraint and the physical loss, as demonstrated by Sun et al. In this experiment we weighted the boundary term by a factor of 100, which showed the best results for our cases.

The advantage of the GAPINN framework is that the quantities of interest can be estimated at any arbitrary point within the domain. Only the number of points describing the geometry should sufficiently represent the geometry and must be evaluated based on the given problem. But in general, the network architecture of all networks shown here is not limited in the number of points to be evaluated, in contrast to [8]. A disadvantage using point based computation compared to methods based on structured spatial representation is the higher computational effort to calculate the derivatives by means of AD, during training of the PINNs. Gao et al. showed this advantage by the

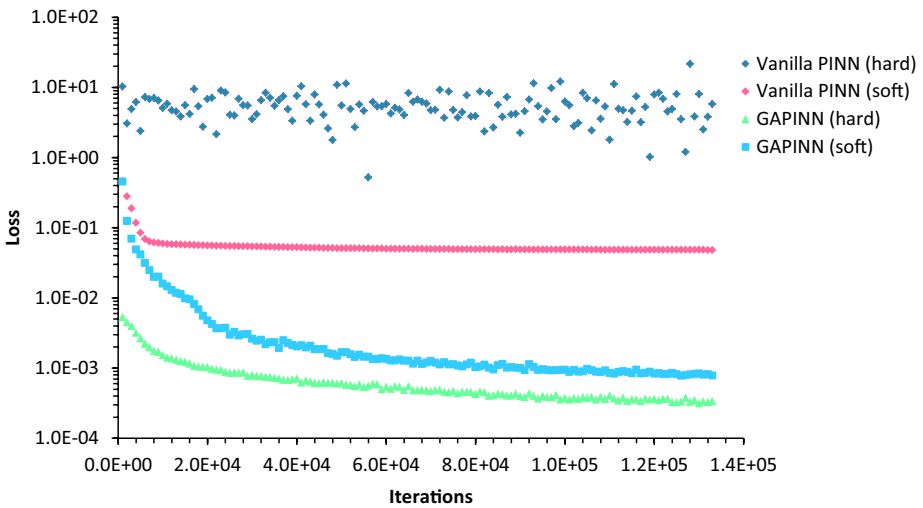

Fig. 7 Comparison of loss over epoch during the network training for both vanilla PINN and the proposed GAPINN using hard and soft constraints for generating surrogate models for solving Navier–Stokes equations on non-parameterized geometries without the use of training data. Depicted is the error after each epoch

usage of Finite Difference methods implemented in efficient kernel operations. The chosen encoder structure provides a permutation invariant processing of the geometry describing points [10]. Depending on the problem, other architectures are also possible as encoders. For example, it is reasonable to use convolution operations for geometries that do not contain permutation problems, e.g. if a unique sorting of the spatial information can be done. It would also be conceivable to obtain the latent representation by 2D or 3D CNN on the basis of image data representing the geometry, which could be feasible for example in imaging based medical examinations. Apart from reduction of geometric dimension using VAE, it could be possible to use other techniques of dimension reduction such as PCA.

Jin et al. showed another benefit of using PINNs for surrogate models, if the boundary conditions are not well known or are badly imposed, a conventional solver would fail [20]. The PINN can also produce results, although it has to be carefully evaluated to ensure the meaningfulness of the predictions. They also demonstrated another advantage by using PINNs to resolve for unknown Reynolds numbers based on scattered velocity data [20]. Thus, the concept of PINNs facilitates the prediction of many unknown quantities where conventional methods would be cumbersome. Our concept extends the usefulness of PINNs to irregular non-parameterized geometries.

We proved the concept of GAPINN for experiment with $Re = 500$ but we want to point out that this framework could be adapted to other Reynolds numbers within the laminar regimen. It could also be possible to use the Reynolds number as input parameter in addition to latent vector and spatial coordinates. Sun et al. experimented on this approach for parametrized stenosis and achieved promising results [2]. The application of this method on turbulent flow must be investigated in future studies.

It has been shown by Sun et al. that a soft boundary constraint yields worse results compared to a hard constrained boundary condition [2]. Based on this, we have shown a simple method to hard constrain the boundary conditions for irregular non-parameterized geometries. This method is for example suitable for many common flow situations. However, it should not be concluded that network architecture with soft boundary constraints cannot be used, the advantages should be individually evaluated.

Lastly, we want to discuss the difference in computational performance for the experiment presented here. Solving the quasi 2d simulation, with 150 000 hexahedron elements, required 210 CPU seconds until convergence by using FVM implemented in OpenFOAM [18]. The training for the GAPINN method with 133,000 iterations, after which the loss converged, lasted approx. 13 h. By optimizing the training code, faster training could be possible. If we compare this training time on one GPU to the potential time for generating 1000 simulations (like the set included in training) we have a total of 86 h on one CPU (Intel Xeon Gold 6226R). It should be noted that it is hard to directly compare CFD-simulation time done on CPU with surrogate neural network training on GPU because of the use of different computation devices but also by the result which is given by each method. It should be noted that the comparative CFD simulations were not optimized in terms of their computational cost.

Once the GAPINN was trained, the estimation is very fast, but depends on the number of points entered. The calculation with the GAPINN method of the analogue spatial distribution as used for CFD required 0.32 GPU seconds. While conventional

CFD simulations need to be done on each geometry separately, GAPINN allows to train a surrogate model which can be used for similar non-parametric geometries after training, see validation with geometries not involved in training (Figs. 4 and 5). If one considers the time consumption for discretization steps as required step in the pre-processing of a CFD analysis, the advantage of GAPINN becomes even clearer. Particularly with regard to complex 3D geometries, a very high advantage is foreseeable.

Conclusion and future work

This article presented a novel DNN framework (GAPINN) to create a fluid flow surrogate model based on irregular geometries using PINNs without the need for training data. The framework involves three network types. The first network (SEN) reduces the dimensions of the irregular geometries to a latent representation. In this work we prove the feasibility of VAE-type networks for this task. We proposed the concept of using this latent representation in combination with spatial coordinates as input for the second network, a PINN. Using PINN we showed that it is possible to train a surrogate model purely driven on the reduction of the residuals of the underlying PDE. Furthermore, we showed the way of designing a boundary constraining network (BCN), the third network, to hardly enforce boundary conditions during training of the PINN. We evaluated this concept on an exemplarily experiment in the fields of biofluidmechanics. For this we generated 1000 different irregular shaped vessels in 2D. The experiments were done at Re 500. The main highlight of the presented GAPINN is the use of PINNs on irregular non-parameterized geometries. Despite that we showed the usage of this framework for Navier–Stokes equations, we see no major problems to adapt to usage for other problems described by PDE. In the future our aim is to study the possibility to use GAPINNs on time varying geometries and boundary conditions such as in pulsatile regimen of cardiovascular flows.

Abbreviations

PINN	Physics informed neural network
SEN	Shape encoding network
BCN	Boundary constraining network
FVM	Finite volumes method
PDE	Partial differential equation
DNN	Deep Neural Network
PCA	Principal Component Analysis
ROM	Reduced Order Method
POD	Proper Orthogonal Decomposition
FCNN	Fully connected neural network
GAPINN	Geometry aware physics informed neural network
AD	Automatic differentiation
FCNN	Fully connected neural network

Acknowledgements
Not applicable.

Author contributions
MS and FB made substantial contributions to the design of the experiments, helped interpreting the results of used analysis methods and helped drafting the manuscript and also substantively revised the manuscript and also perform CFD-simulations. AÖ and KPS made substantial contributions to the conception of the work. JO made substantial contributions to the conception and design of the work. He invented the network architecture used in this work. JO performed neural network training and analysis and interpretation of data. He also drafted the manuscript. All authors read and approved the final manuscript.

Funding
Open Access funding enabled and organized by Projekt DEAL. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017578.

Availability of data and materials
The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 18 March 2022 Accepted: 24 May 2022
Published: 21 June 2022

References

1. Lassila T, Manzoni A, Quarteroni A, Rozza G. Model order reduction in fluid dynamics: challenges and perspectives. In: Quarteroni A, Rozza G, editors. Reduced order methods for modeling and computational reduction. Cham: Springer International Publishing; 2014. p. 235–73. https://doi.org/10.1007/978-3-319-02090-7_9.
2. Sun L, Gao H, Pan S, Wang J-X. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. Comput Methods Appl Mech Eng. 2020;361(4): 112732. https://doi.org/10.1016/j.cma.2019.112732.
3. Chaturantabut S, Sorensen DC. Nonlinear model reduction via discrete empirical interpolation. SIAM J Sci Comput. 2010;32(5):2737–64. https://doi.org/10.1137/090766498.
4. Lee H, Kang IS. Neural algorithm for solving differential equations. J Comput Phys. 1990;91(1):110–31. https://doi.org/10.1016/0021-9991(90)90007-N.
5. Lagaris IE, Likas AC, Papageorgiou DG. Neural-network methods for boundary value problems with irregular boundaries. IEEE Trans Neural Netw. 2000;11(5):1041–9. https://doi.org/10.1109/72.870037.
6. Raissi M, Perdikaris P, Karniadakis GE. Physics informed deep learning (part I): data-driven solutions of nonlinear partial differential equations. 2017. http://arxiv.org/pdf/1711.10561v1.
7. Nabian MA, Meidani H. Physics-driven regularization of deep neural networks for enhanced engineering design and analysis. J Comput Inf Sci Eng. 2020;20(1):436. https://doi.org/10.1115/1.4044507.
8. Gao H, Sun L, Wang J-X. PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. J Comput Phys. 2021;428(5): 110079. https://doi.org/10.1016/j.jcp.2020.110079.
9. Kingma DP, Welling M. Auto-encoding variational Bayes. 2013. http://arxiv.org/pdf/1312.6114v10.
10. Qi CR, Su H, Mo K, Guibas LJ. PointNet: deep learning on point sets for 3D classification and segmentation. 2016. http://arxiv.org/pdf/1612.00593v2.
11. Kullback S, Leibler RA. On information and sufficiency. Ann Math Stat. 1951;22:79–86.
12. Fan H, Su H, Guibas L. A point set generation network for 3D object reconstruction from a single image. 2016. http://arxiv.org/pdf/1612.00603v2.
13. Heaton J. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: deep learning. Genet Program Evolvable Mach. 2018;19(1–2):305–7. https://doi.org/10.1007/s10710-017-9314-z.
14. Raissi M, Perdikaris P, Karniadakis GE. Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. J Comput Phys. 2019;378:686–707. https://doi.org/10.1016/j.jcp.2018.10.045.
15. Elfwing S, Uchibe E, Doya K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. 2017. http://arxiv.org/pdf/1702.03118v3.
16. Kingma DP, Ba J. Adam: a method for stochastic optimization. 2014. http://arxiv.org/pdf/1412.6980v9.
17. Berg J, Nyström K. A unified deep artificial neural network approach to partial differential equations in complex geometries. Neurocomputing. 2018;317(9):28–41. https://doi.org/10.1016/j.neucom.2018.06.056.
18. Weller HG, Tabor G, Jasak H, Fureby C. A tensorial approach to computational continuum mechanics using object-oriented techniques. Comput Methods Appl Mech Eng. 1998;12(6):620. https://doi.org/10.1063/1.168744.
19. van der Maaten L. Barnes-Hut-SNE. 2013. http://arxiv.org/pdf/1301.3342v2.
20. Jin X, Cai S, Li H, Karniadakis GE. NSFnets (Navier–Stokes flow nets): physics-informed neural networks for the incompressible Navier–Stokes equations. J Comput Phys. 2021;426: 109951. https://doi.org/10.1016/j.jcp.2020.109951.

Publisher's Note