Journal of Computational Science 95 (2026) 102817

Contents lists available at ScienceDirect

Journal of Computational Science

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

Reliable physics-informed neural networks for Navier–Stokes simulations. Can we trust AI-generated numerical simulations?☆

Tomasz Służalec [a,b], Daria Wójcik [a], Carlos Uriarte [c,d], Marcin Łoś [a], Anna Paszyńska [a], Maciej Paszyński [a,*]

[a] AGH University of Krakow, Krakow, Poland
[b] Institute of Fundamental Technological Research, Polish Academy of Sciences, Warsaw, Poland
[c] Basque Center for Applied Mathematics, Bilbao, Spain
[d] School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Perth, Australia

ARTICLE INFO

Keywords:
Reliable AI
Physics informed neural networks
Robust loss
Collocation-based robust variational physics informed neural networks
Finite element method
Navier–stokes equations
Cavity flow problem

ABSTRACT

The Navier–Stokes equations are one of the most important computational problems. They are employed for modeling the flow of the air around the airplane, for the verification of the airplane's safety. The Navier–Stokes equations or their simplified version, the shallow water equations, are used for tsunami prediction simulations. In the extended version, the Navier–Stokes–Boussinesq equations are employed for weather prediction simulations. In this paper, we show how challenging it is for PINNs with a robust loss function to solve the model Navier–Stokes problem, the lid-driven cavity flow, especially at high Reynolds numbers. If PINNs cannot solve a model lid-driven cavity problem, how can we trust the AI-generated numerical simulations? We investigate the common understanding and trust into AI generated numerical simulations among the computer science students, and we relate these findings to the conclusions from the numerical simulations. It is very important to understand the accuracy of AI simulators at a reliable level to prevent future disasters resulting from their misuse. Finally, we provide Google Colab and GitHub links to numerical codes, namely PINNs and CRVPINNs in Python, and a C++ FEM code in FeniCs.

1. Introduction

We investigate the ability of Physics-Informed Neural Networks, augmented with a robust loss function, to solve the high-Reynolds-number Navier–Stokes equations. The aim of this article is to compare the common view among computer science students on the reliability of calculations in engineering applications performed by Artificial Intelligence (AI) with the actual results of calculations performed for model computational problems. As a model computational problem, we consider the cavity flow problem described by the Navier–Stokes equations. We compare the AI-generated numerical results with the reference solution computed using the state-of-the-art Finite Element Method (FEM) solver in FeniCS. We consider different Reynolds numbers to investigate the true error of the Variational PINN with a robust loss function augmented with the collocation method.

Recently, Physics Informed Neural Networks (PINN) [1,2] have been proposed as an artificial intelligence tool to solve Partial Differential Equations (PDEs) based on strong residuals, and Variational Physics Informed Neural Networks (VPINNs) based on weak residuals [3]. The methods have multiple applications, including fluid flow [4], high speed fluid flow [5], material science [6,7], and wave propagation problems [8], among others. The number of papers in the field of PINNs is growing exponentially [9–14]. The common interest in using neural networks in scientific computing raises the question: What accuracy can be achieved with these methods? How do they compare with state-of-the-art finite element method solvers? What is a common understanding (and belief) regarding the accuracy delivered by neural networks when dealing with PDE-based simulations?

In this paper, we aim to investigate these questions by formulating a model lid driven problem [15–18] and employing this model formulation to test the accuracy of PINN solutions. The reason for selecting a model Navier–Stokes problem is the following. The Navier–Stokes equations are used to compute how air flows around the airplane [19]. This is particularly relevant to estimating airplane safety and requires high-Reynolds-number computations due to the airplane's high velocity and a need of solving Navier–Stokes equations with $Re > 10^6$ [19]. The Navier–Stokes equations are also employed to predict tsunami

☆ This article is part of a Special issue entitled: 'ICCS 2025' published in Journal of Computational Science.

* Corresponding author.
E-mail address: paszynsk@agh.edu.pl (M. Paszyński).

https://doi.org/10.1016/j.jocs.2026.102817
Received 12 November 2025; Received in revised form 15 February 2026; Accepted 21 February 2026
Available online 23 February 2026

propagation [20], often using the simplified shallow-water equations model [21]. If we take the tsunami speed of 50 km/h (14 m/s), the kinematic viscosity coefficient of water is 10^{-6} $\frac{m^2}{s}$, then for a characteristic size of 1 m the Reynolds number is about 10^7. The extension of the Navier–Stokes equations into the Navier–Stokes–Boussinesq model, which includes temperature, enables weather prediction simulations. The Reynolds number during these simulations can vary depending on the modeled phenomena [22]. Thus, to answer the question of how well can we trust AI-generated simulations using PINN methods, we will investigate this on a model lid-driven cavity problem with different Reynolds numbers.

We solved this problem using Physics Informed Neural Networks [1], augmented with a robust loss function [23], and sped up using the collocation method [24]. We compared the numerical solution with a state-of-the-art finite element method solver in FeniCS [25]. For a fair comparison, we compared the PINNs method [2], our recent version of PINNs augmented with a robust loss function and a collocation-based accelerator, with the standard finite element method formulation using the traditional GMRES solver [26] and a direct solver [27,28].

The goal of this paper is to collect the conclusions from model Navier–Stokes simulations, verify their accuracy, and compare them with the common understanding among computer science students. It is very important for them to have a clear understanding of the accuracy of AI simulators to prevent future disasters resulting from the misuse of AI simulations. We conducted surveys among computer science students at two Krakow universities: AGH University of Krakow and Jagiellonian University. The aim of our surveys was to examine students' opinions on the reliability of simulations performed using artificial intelligence tools.

The structure of the paper is as follows. In Section 2, we introduce the lid-driven cavity flow model problem. In Section 3, we introduce the PINN formulation for the lid-driven cavity flow problem. The next Section introduces the Collocation-based Robust Variational PINN method. For the reference solution obtained using finite element method formulation with GMRES iterative solver and MUMPS direct solver we refer to Section 5. In the next section we discuss the reliability of the AI simulations by comparing FEM, PINN and CRVPINN results. Finally, Section 6 discusses our surveys. Section 7 provides the links to the computational codes. We conclude the paper in Section 8.

2. The lid-driven cavity flow model problem

We investigate a benchmark Navier–Stokes problem using three distinct numerical frameworks: Physics-Informed Neural Networks (PINNs), Collocation-based Robust Variational Physics-Informed Neural Networks (CRVPINNs), and the Finite Element Method (FEM) stabilized with a Streamline Upwind Petrov–Galerkin (SUPG) formulation. The objective is to evaluate the accuracy and robustness of the AI-based formulations (PINNs and CRVPINNs) relative to the established FEM–SUPG approach in solving incompressible fluid flow problems governed by the Navier–Stokes equations. These equations form the cornerstone of computational analyzes in aerospace and automotive engineering, among other fields.

Let $\Omega \subset \mathbb{R}^2$ be a bounded spatial domain. For the lid-driven cavity flow, we seek the velocity field $\mathbf{u} = (u_1, u_2)$ and the pressure scalar field p that satisfy

$$\begin{cases} \mathbf{u}\cdot\nabla\mathbf{u} - \frac{1}{Re}\Delta\mathbf{u} + \nabla p = & 0 & \text{in} & \Omega, \\ \nabla\cdot\mathbf{u} = & 0 & \text{in} & \Omega, \\ \mathbf{u} = & \mathbf{g} & \text{in} & \partial\Omega. \end{cases} \tag{1}$$

In our simulations, we consider a range of Reynolds numbers spanning different orders of magnitude: $Re \in \{1, 10, 100, 1000\}$.

The cavity flow problem describes the motion of an incompressible fluid confined within a square cavity driven by a lid along the top boundary. This configuration serves as a standard benchmark for evaluating the accuracy and stability of numerical solvers for incompressible flows.

No-slip boundary conditions are imposed along the left, right, and bottom walls, where the velocity is set to zero. Along the top boundary, the fluid is driven by a prescribed tangential velocity, introducing a continuous transition region between the stationary side walls and the moving lid. Within this region, the velocity varies linearly from zero to the lid velocity. The thickness of the transition zone, denoted by ϵ, controls the smoothness of the imposed boundary data and affects the intensity of the velocity singularities at the upper corners. The Dirichlet boundary conditions are thus specified as

$$\mathbf{g}(x_1, x_2) = \begin{cases} (0,0), & x_1 \in (0,1),\ x_2 = 0, \\ (0,0), & x_1 \in \{0,1\},\ x_2 \in (0, 1-\epsilon), \\ (1,0), & x_1 \in (0,1),\ x_2 = 1, \\ \left(1 - \frac{1-x_2}{\epsilon}, 0\right), & x_1 \in \{0,1\},\ x_2 \in (1-\epsilon, 1), \end{cases} \tag{2}$$

In PINN and CRVPINN codes, the boundary conditions are enforced by using hard constraints. Namely `logits = logits*20*x*(1-x)*y*(1-y) + exp(-1000*(y-1)**2)`. There is no specific ϵ there. The `exp(-1000*(y-1)**2)` quickly goes from 1 to 0. For the finite element Navier–Stokes solver in FEniCSx, we indeed enforce this condition by taking $\epsilon = 0.00001$ (which corresponds to `np.isclose` default tolerance).

3. The PINN framework

In our PINN framework, we consider the following feed-forward fully-connected neural network:

$$\mathcal{NN}_\theta(x_1, x_2) := A_n \ldots \sigma\Big(A_2\,\sigma(A_1 \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b_1) + b_2\Big) + \cdots + b_n, \tag{3}$$

where $\theta = \{A_k, b_k\}_{k=1}^n$ denotes the set of trainable parameters of the neural network. Each A_k is the weight matrix connecting layer $k-1$ to layer k, and b_k is the corresponding bias vector. The network takes the spatial coordinates $(x_1, x_2) \in \Omega$ as input and outputs a vector of size seven,

$$\mathcal{NN}_\theta(x_1, x_2) = \Big(\mathcal{NN}_{\theta,j}(x_1, x_2)\Big)_{j=1}^7. \tag{4}$$

The first three output components of $\mathcal{NN}_\theta$ are used to approximate the velocity and pressure fields of the model problem solution (u_1, u_2, p), while the remaining four components represent auxiliary functions that we will specify shortly.

To impose the boundary conditions, we multiply the first three output components of $\mathcal{NN}_\theta$ by the following scaling and shift functions

$$\begin{aligned} u_{\theta,1}(x_1, x_2) &:= \mathcal{NN}_{\theta,1}(x_1, x_2)\cdot\phi_{Dirichlet}(x_1, x_2) + \phi_{up}(x_1, x_2), \\ u_{\theta,2}(x_1, x_2) &:= \mathcal{NN}_{\theta,2}(x_1, x_2)\cdot\phi_{Dirichlet}(x_1, x_2), \\ p_\theta(x_1, x_2) &:= \mathcal{NN}_{\theta,3}(x_1, x_2)\cdot\phi_{center}(x_1, x_2), \end{aligned} \tag{5}$$

where $\phi_{Dirichlet} = 20x(1-x)y(1-y)$ enforces zero velocity on the domain boundaries, $\phi_{up} = \exp(-1000(1-y)^2)$ allows for the imposition of the lid velocity, and $\phi_{center} = 1.0 - \exp(-1000(x-0.5)^2)\exp(-1000(y-0.5)^2)$ fixes the pressure at the central point to ensure uniqueness (see Fig. 1). As a result, $u_{\theta,1}$, $u_{\theta,2}$, and p_θ now represent feasible candidates for the approximation of u_1, u_2, and p, respectively, at the expense of a loss function that no longer needs to incorporate boundary conditions.

Our loss function is now inspired by a first-order system least-squares (FOSLS) formulation of the Navier–Stokes model problem (1),

which introduces four auxiliary variables as follows:

$$\begin{aligned} w_1 &:= \frac{\partial u_1}{\partial x_1}, \quad w_2 := \frac{\partial u_1}{\partial x_2}, \quad z_1 := \frac{\partial u_2}{\partial x_1}, \quad z_2 := \frac{\partial u_2}{\partial x_2}, \\ u_1 w_1 + u_2 w_2 - \frac{1}{Re}\frac{\partial w_1}{\partial x_1} - \frac{1}{Re}\frac{\partial w_2}{\partial x_2} + \frac{\partial p}{\partial x_1} &= 0, \\ u_1 z_1 + u_2 z_2 - \frac{1}{Re}\frac{\partial z_1}{\partial x_1} - \frac{1}{Re}\frac{\partial z_2}{\partial x_2} + \frac{\partial p}{\partial x_2} &= 0, \\ \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} &= 0. \end{aligned} \tag{6}$$

These new variables will be represented by the last four components of $\mathcal{NN}_\theta$. Thus, it results sensible to define the following residual functions in the spirit of FOSLS:

$$\begin{aligned} RES_a(\theta)(x_1,x_2) &:= \frac{\partial u_{\theta,1}}{\partial x_1}(x_1,x_2) - \mathcal{NN}_{\theta,4}(x_1,x_2), \\ RES_b(\theta)(x_1,x_2) &:= \frac{\partial u_{\theta,1}}{\partial x_2}(x_1,x_2) - \mathcal{NN}_{\theta,5}(x_1,x_2), \\ RES_c(\theta)(x_1,x_2) &:= \frac{\partial u_{\theta,2}}{\partial x_1}(x_1,x_2) - \mathcal{NN}_{\theta,6}(x_1,x_2), \\ RES_d(\theta)(x_1,x_2) &:= \frac{\partial u_{\theta,2}}{\partial x_2}(x_1,x_2) - \mathcal{NN}_{\theta,7}(x_1,x_2), \\ RES_e(\theta)(x_1,x_2) &:= u_{\theta,1}(x_1,x_2)\,\mathcal{NN}_{\theta,4}(x_1,x_2) + u_{\theta,2}(x_1,x_2)\,\mathcal{NN}_{\theta,5}(x_1,x_2) \\ &\quad - \frac{1}{Re}\frac{\partial \mathcal{NN}_{\theta,4}}{\partial x_1}(x_1,x_2) - \frac{1}{Re}\frac{\partial \mathcal{NN}_{\theta,5}}{\partial x_2}(x_1,x_2) + \frac{\partial p_\theta}{\partial x_1}(x_1,x_2), \\ RES_f(\theta)(x_1,x_2) &:= u_{\theta,1}(x_1,x_2)\,\mathcal{NN}_{\theta,6}(x_1,x_2) + u_{\theta,2}(x_1,x_2)\,\mathcal{NN}_{\theta,7}(x_1,x_2) \\ &\quad - \frac{1}{Re}\frac{\partial \mathcal{NN}_{\theta,6}}{\partial x_1}(x_1,x_2) - \frac{1}{Re}\frac{\partial \mathcal{NN}_{\theta,7}}{\partial x_2}(x_1,x_2) + \frac{\partial p_\theta}{\partial x_2}(x_1,x_2), \\ RES_g(\theta)(x_1,x_2) &:= \frac{\partial u_{\theta,1}}{\partial x_1}(x_1,x_2) + \frac{\partial u_{\theta,2}}{\partial x_2}(x_1,x_2). \end{aligned} \tag{7}$$

We thus define the PINN loss function as the sum of squared residuals evaluated at a set of provided collocation points $C = \{(x_1^k, x_2^k)\}_{k=1}^N$ in the computational domain:

$$LOSS_{\text{PINN}}(\theta)(C) := \sum_{k=1}^{N} \sum_{i\in\{a,b,\ldots,g\}} \Big(RES_i(\theta)(x_1^k, x_2^k)\Big)^2. \tag{8}$$

Minimization of $LOSS_{\text{PINN}}(\theta)$, combined with random resampling of the collocation set C at each training iteration, drives the neural network to approximate a solution consistent with the governing Navier–Stokes equations. The training procedure can be summarized as follows:

1. Sample a set of collocation points $C_p = \{(x_1^k, x_2^k)\}_{k=1}^n$ within the computational domain.
2. Update the network parameters $\{A_k, b_k\}_{k=1}^n$ using a gradient-descent-based optimization scheme:
$$A_k^{i,j} \leftarrow A_k^{i,j} - \eta \frac{\partial LOSS_{\text{PINN}}(\theta)}{\partial A_k^{i,j}}, \qquad b_k^i \leftarrow b_k^i - \eta \frac{\partial LOSS_{\text{PINN}}(\theta)}{\partial b_k^i},$$
where η denotes the learning rate, which should be understood as absorbing momentum and adaptive scaling effects present in memory-based schemes such as Adam [29].
3. Repeat the parameter updates over multiple epochs until convergence, e.g., when $LOSS_{\text{PINN}}(\theta)$ stabilizes below a tolerance or a maximum number of iterations is reached.

4. The CRVPINN framework

Above, we introduced PINNs based on a FOSLS formulation the produces the residual vector $RES(u_\theta)$ that is evaluated at collocation points in $\Omega = [0,1]^2$ to define the loss function. While (8) enforces physical consistency, it is unrobust: a small loss value does not necessarily imply a small numerical error $\|u_\theta - u_{\text{exact}}\|$. To address this limitation, we introduce the *Collocation-based Robust Variational PINN* (CRVPINN) framework [24].

The CRVPINN method employs the same FOSLS formulation as in (6), but instead of operating over a continuous domain, it considers a discrete collocation grid

$$\Omega_h = \{(ih_x, jh_y) \in (0,1)^2 : 0 \le i \le N_x,\ 0 \le j \le N_y\}, \tag{9}$$

where $h_x = 1/N_x$ and $h_y = 1/N_y$. Each scalar field u is represented by its nodal values $u_{i,j} = u(ih_x, jh_y)$. Accordingly, we define the discrete inner product and the induced norm as follows:

$$(u,v)_h := h_x h_y \sum_{(i,j)\in\Omega_h} u_{i,j} v_{i,j}, \qquad \|u\|_h^2 := (u,u)_h. \tag{10}$$

Spatial derivatives are thus approximated using forward and backward finite differences:

$$\nabla_+ u_{ij} = \left(\frac{u_{i+1,j} - u_{i,j}}{h_x}, \frac{u_{i,j+1} - u_{i,j}}{h_y}\right), \tag{11}$$

$$\nabla_- u_{ij} = \left(\frac{u_{i,j} - u_{i-1,j}}{h_x}, \frac{u_{i,j} - u_{i,j-1}}{h_y}\right). \tag{12}$$

These discrete gradient operators define a discrete weak form of the first-order Stokes system as follows:

$$\begin{cases} \mathbf{u}\cdot\boldsymbol{\sigma} - \nabla_+\cdot\boldsymbol{\sigma} + \nabla_+ p = 0, \\ \nabla_-\cdot\mathbf{u} = 0, \\ \boldsymbol{\sigma} - \nabla_-\mathbf{u} = 0. \end{cases} \tag{13}$$

where

$$\boldsymbol{\sigma} = \begin{bmatrix} w_1 & w_2 \\ z_1 & z_2 \end{bmatrix}, \quad \nabla\cdot\boldsymbol{\sigma} = \begin{bmatrix} \frac{\partial w_1}{\partial x} + \frac{\partial w_2}{\partial y} \\ \frac{\partial z_1}{\partial x} + \frac{\partial z_2}{\partial y} \end{bmatrix}, \quad \nabla\mathbf{u} = \begin{bmatrix} \frac{\partial u_1}{\partial x} & \frac{\partial u_1}{\partial y} \\ \frac{\partial u_2}{\partial x} & \frac{\partial u_2}{\partial y} \end{bmatrix}. \tag{14}$$

Following [24], we define the *graph inner product* on the discrete space:

$$\begin{aligned} (u,v)_{\text{graph}} &= \big(\nabla_+\cdot\boldsymbol{\sigma} - \nabla_+ p, \nabla_+\cdot\boldsymbol{\tau} - \nabla_+ q\big)_h + \big(\nabla_-\cdot\mathbf{u}, \nabla_-\cdot\mathbf{v}\big)_h \\ &\quad + \big(\boldsymbol{\sigma} + \nabla_-\mathbf{u}, \boldsymbol{\tau} + \nabla_-\mathbf{v}\big)_h + (\boldsymbol{\sigma},\boldsymbol{\tau})_h + (\mathbf{u},\mathbf{v})_h + (p,q)_h \\ &= \big(\nabla_+\cdot\boldsymbol{\sigma}, \nabla_+\cdot\boldsymbol{\tau}\big)_h + 2\,(\boldsymbol{\sigma},\boldsymbol{\tau})_h \\ &\quad + \big(\nabla_-\cdot\mathbf{u}, \nabla_-\cdot\mathbf{v}\big)_h + \big(\nabla_-\mathbf{u}, \nabla_-\mathbf{v}\big)_h + (\mathbf{u},\mathbf{v})_h \\ &\quad + \big(\nabla_+ p, \nabla_+ q\big)_h + (p,q)_h \\ &\quad + \big(\nabla_-\mathbf{u}, \boldsymbol{\tau}\big)_h + \big(\boldsymbol{\sigma}, \nabla_-\mathbf{v}\big)_h \\ &\quad + \big(-\nabla_+ p, \nabla_+\cdot\boldsymbol{\tau}\big)_h + \big(\nabla_+\cdot\boldsymbol{\sigma}, -\nabla_+ q\big)_h \end{aligned} \tag{15}$$

This scalar product induces a symmetric positive-definite *Gram matrix* $\mathbf{G}$, assembled from bilinear forms of the type

$$g_{\boldsymbol{\sigma}}(\boldsymbol{\sigma},\boldsymbol{\tau}) = \big(\nabla_+\cdot\boldsymbol{\sigma}, \nabla_+\cdot\boldsymbol{\tau}\big)_h + 2\,(\boldsymbol{\sigma},\boldsymbol{\tau})_h\,, \tag{16}$$

$$g_{\mathbf{u}}(\mathbf{u},\mathbf{v}) = \big(\nabla_-\cdot\mathbf{u}, \nabla_-\cdot\mathbf{v}\big)_h + \big(\nabla_-\mathbf{u}, \nabla_-\mathbf{v}\big)_h + (\mathbf{u},\mathbf{v})_h\,, \tag{17}$$

$$g_p(p,q) = \big(\nabla_+ p, \nabla_+ q\big)_h + (p,q)_h\,, \tag{18}$$

together with appropriate coupling blocks $g_{\boldsymbol{\sigma}\mathbf{u}}$ and $g_{\boldsymbol{\sigma}p}$. Hence, $\mathbf{G}$ has a block structure of the form

$$\mathbf{G} = \begin{bmatrix} G_{\boldsymbol{\sigma}} & G_{\boldsymbol{\sigma}\mathbf{u}} & G_{\boldsymbol{\sigma}p} \\ G_{\boldsymbol{\sigma}\mathbf{u}}^T & G_{\mathbf{u}} & 0 \\ G_{\boldsymbol{\sigma}p}^T & 0 & G_p \end{bmatrix}. \tag{19}$$

The CRVPINN loss replaces the standard least-squares loss (8) with a *robust, norm-equivalent* formulation:

$$\mathcal{L}_{\text{CRVPINN}}(u_\theta) = \text{RES}(u_\theta)^T\, \mathbf{G}^{-1}\, \text{RES}(u_\theta), \tag{20}$$

where $\text{RES}(u_\theta)$ is the residual vector already defined in the PINN section. This modification guaranties that the loss value provides a reliable indicator of the true numerical error:

$$\frac{1}{\mu}\sqrt{\mathcal{L}_{\text{CRVPINN}}} \le \|u_\theta - u_{\text{exact}}\| \le \frac{1}{\alpha}\sqrt{\mathcal{L}_{\text{CRVPINN}}}, \tag{21}$$

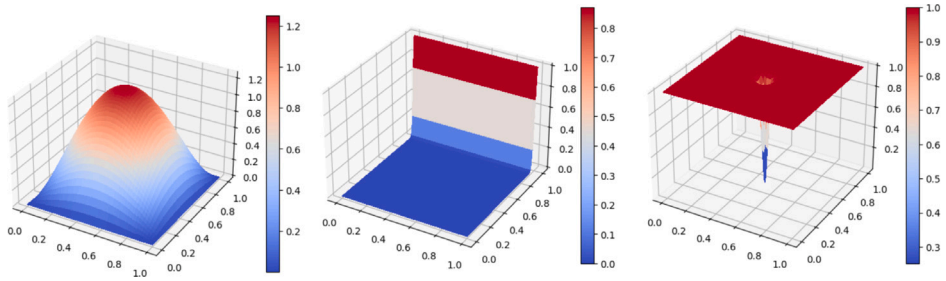


Fig. 1. From left to right: $\phi_{Dirichlet}$ is employed to enforce null Dirichlet boundary conditions, ϕ_{up} is employed to enforce the boundary condition at the top of the domain, and ϕ_{center} is employed to enforce zero pressure condition at the center of the domain.

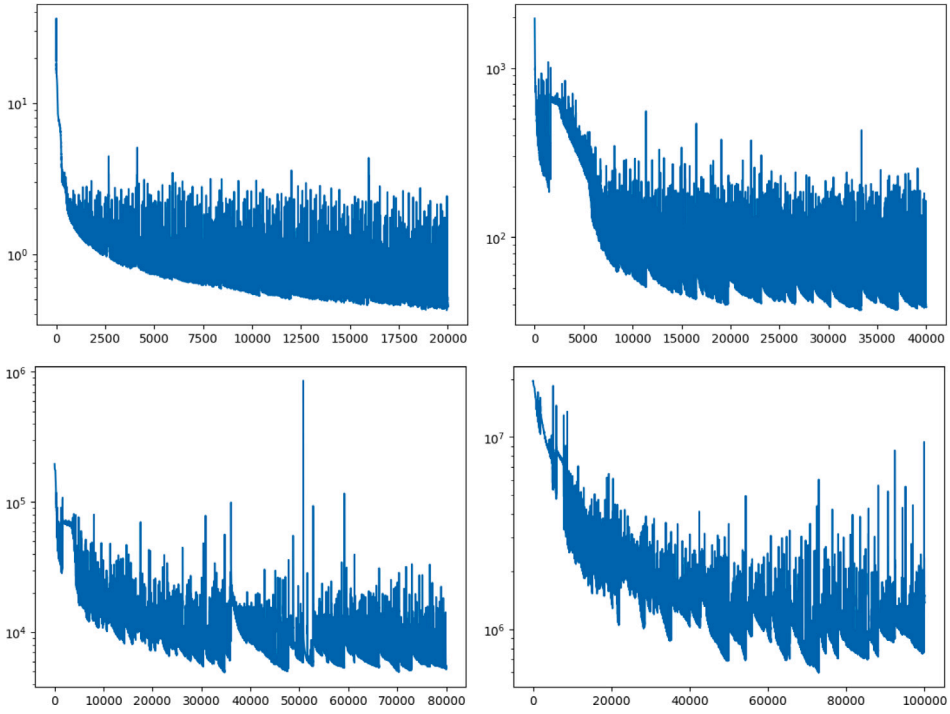


Fig. 2. The convergence of the training of the loss function for the Navier–Stokes problem with top boundary $Re = 1$ (left top panel), $Re = 10$ (right top panel), $Re = 100$ (left bottom panel), $Re = 1000$ (right bottom panel).

where μ and α are the inf–sup stability and continuity constants, respectively. Thus, the convergence of $\mathcal{L}_{\text{CRVPINN}}$ reflects the true accuracy of the neural approximation.

In practice, $\mathbf{G}^{-1}$ is not computed explicitly; instead, an LU factorization $\mathbf{G} = \mathbf{LU}$ is used for the efficient evaluation of (20). The construction of $\mathbf{G}$ relies on Kronecker-delta test functions in the discrete weak form, ensuring boundedness and inf–sup stability of the resulting discrete norm.

5. Finite element simulations

To solve Navier–Stokes equations using the Finite Element Method, we first derive the variational formulation of (1). Let $V = H_0^1(\Omega)$ and $Q = L_0^2(\Omega)$ where

$$L_0^2(\Omega) = \left\{ f \in L^2(\Omega) : \int_\Omega f \, dx = 0 \right\} \tag{22}$$

Let $\mathbf{u}_0 \in H^1(\Omega)$ be a fixed vector field on Ω satisfying the boundary conditions (2). In the weak formulation of the Navier–Stokes equations, we ask for $(\mathbf{u}, p) \in V \times Q$ such that

$$b((\mathbf{u}, p), (\mathbf{v}, q)) = l((\mathbf{v}, q)) \tag{23}$$

where

$$\begin{aligned} b((\mathbf{u}, p), (\mathbf{v}, q)) &= \int_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} + (\mathbf{u} \cdot \nabla \mathbf{u}, \mathbf{v}) - (p, \nabla \cdot \mathbf{v}) - (\nabla \cdot \mathbf{u}, q) \\ l((\mathbf{v}, q)) &= -b((\mathbf{u}_0, 0)) \end{aligned} \tag{24}$$

Form b is non-linear in the first variable, but linear in the second, so l is a linear functional on $V \times P$.

For the discretization, we use Taylor–Hood elements with quadratic polynomials for the velocity approximation and linear polynomials for the pressure, and a uniform 200×200 computational mesh. The discrete pressure space is constructed by fixing the pressure value at $(0, 0)$ to be 0.

To solve the resulting non-linear problem, we employ Newton solver from PETSc via the FEniCSx framework. To obtain convergence to the correct solution, configuration of the solver needs some care. The Newton solver with a configuration based on the one presented in the FEniCSx tutorial [25,30], which uses GMRES iterative solver:

```
petsc_options = {
  "ksp_type": "gmres",
  "ksp_rtol": 1e-12,
  "ksp_monitor": None,
  "pc_type": "hypre",
```

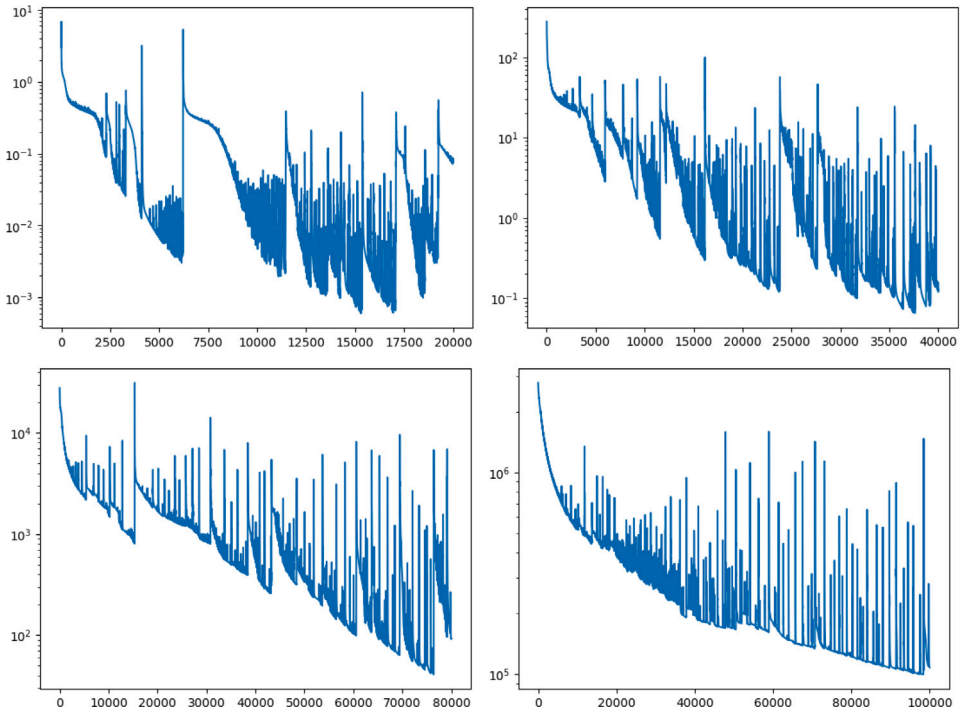


Fig. 3. CRVPINN simulations. The convergence of the training of the loss function for the Navier–Stokes problem with top boundary $Re = 1$ (left top panel), $Re = 10$ (right top panel), $Re = 100$ (left bottom panel), $Re = 1000$ (right bottom panel).

```
  "pc_hypre_type": "boomeramg",
  "pc_hypre_boomeramg_max_iter": 1,
  "pc_hypre_boomeramg_cycle_type": "v",
}
```

converges, but to the wrong value – x component of the velocity is close to the correct solution, but the y component and the pressure are 0.

This is potentially highly misleading, since the solver reports no errors and the residual norms converge to 0, despite the result being incorrect. This happens even with lid velocity 1, corresponding to the Reynolds number of 1, which indicates a numerically easy problem.

This issue can be avoided by using a direct solver, but not all direct solvers work correctly. The default direct solver provided by PETSc is unable to solve the linear systems resulting from linearization in the Newton solver and produces infinite values. Using an external solver like MUMPS:

```
petsc_options = {
  "ksp_type": "preonly",
  "pc_type": "lu",
  "pc_factor_mat_solver_type": "mumps",
}
```

works out of the box for Reynolds numbers $Re \in \{1, 10, 100\}$. Numerical results are presented in Fig. 4.

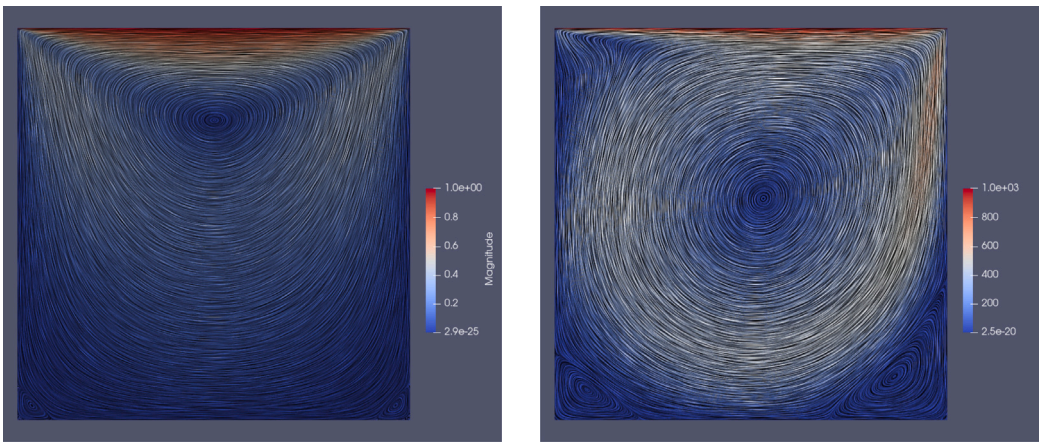
For $Re = 1000$ and higher, the Newton solver with the default starting point $(\mathbf{u}_h, p_h) = 0$ does not converge. However, we can achieve convergence by choosing a starting point that corresponds to the solution of the problem with a lower lid velocity. To solve the problem for $Re = 1000$, first we solve the problem with $Re = 100$, then with $Re = 500$, using the $Re = 100$ solution as the starting point. Finally, we solve the problem with $Re = 1000$, using the solution of $Re = 500$ as the starting point.

6. How reliable are AI CFD simulations - a comparison of PINN, CRVPINN, and FEM simulations

We have compared the numerical results for the PINN, CRVPINN, and the FEM solution. To compare the velocity and pressure we plot the profiles along specially selected lines. Namely, for the v_x velocity component we plot the profile along $\{(x, y) : x = 0.5, y \in (0, 1)\}$. For the v_y velocity profile we plot the profile along $\{(x, y) : x \in (0, 1), y = 0.5\}$. For the pressure profile, we plot the profile along $\{(x, y) : x \in (0, 1), y = 1\}$.

The comparison of the profiles is presented in Figs. 5–8. We assume that the FEM solution can serve as the reference solution, as it uses a high-resolution 200×200 Taylor–Hood elements computational mesh. Another motivation for treating FEM as the exact solution is that FEM solvers serve as the baseline for serious engineering simulations. Unfortunately, from Figs. 7–8 we can see significant differences between the reference FEM solution and the PINN/CRVPINN methods in the velocity and pressure profiles. Analyzing the convergence of the training of the PINN method, presented in Fig. 2, we conclude that the loss values during the training of PINN for $Re = 1, 10, 100, 1000$ reached the levels of 0.1, 10, 1000, and 1 000 000, respectively. Note that the PINN loss is not robust, and it is not directly related to the true error measured in the H^1 norm. For this reason, we have implemented and executed the CRVPINN method that has a robust loss. We can read from Fig. 3 that the robust loss has values of the order of 10^{-3}, 10^{-1}, 100, and 100 000 for $Re = 1, 10, 100$, and $Re = 1000$, respectively. This robust loss is an indicator of the true error, as shown in [23].

What does it mean? Such a large numerical error for high values of the Reynolds number, true error of 10^2 for $Re = 100$, and the true error of 10^5 for $Re = 1000$, is unacceptable in engineering computations. This is why we checked PINN/CRVPINN computations for the model Navier–Stokes equations. They are employed for aerospace engineering simulations (airplane safety), tsunami modeling simulations (in the simplified form of the shallow water equations), or even the weather forecast simulations (using Navier–Stokes–Boussinesq equations). If PINNs cannot accurately solve a simple model lid-driven



(a) velocity ($Re = 1$) (b) velocity ($Re = 1000$)

Fig. 4. Velocity solutions of the Navier–Stokes problem obtained using FEM.

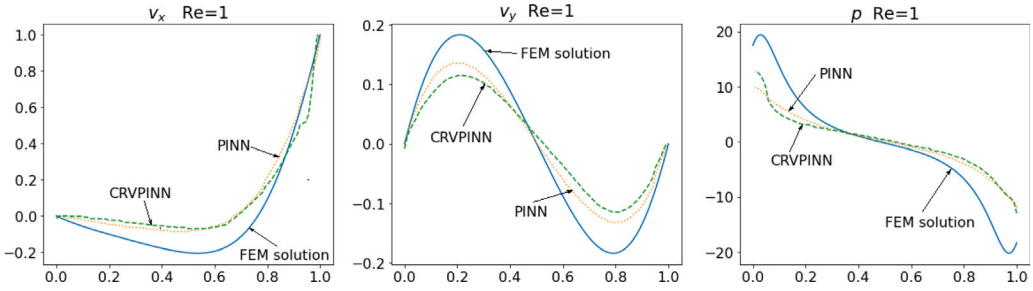


Fig. 5. Comparison of PINN, CRVPINN and FEM results for $Re = 1$. The first column represents v_x velocity component. The second column represents v_y velocity component. The third column represents pressure p.

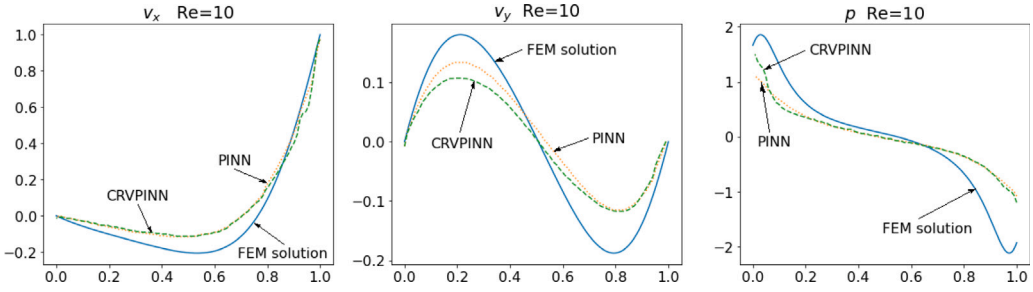


Fig. 6. Comparison of PINN, CRVPINN and FEM results for $Re = 10$. The first column represents v_x velocity component. The second column represents v_y velocity component. The third column represents pressure p.

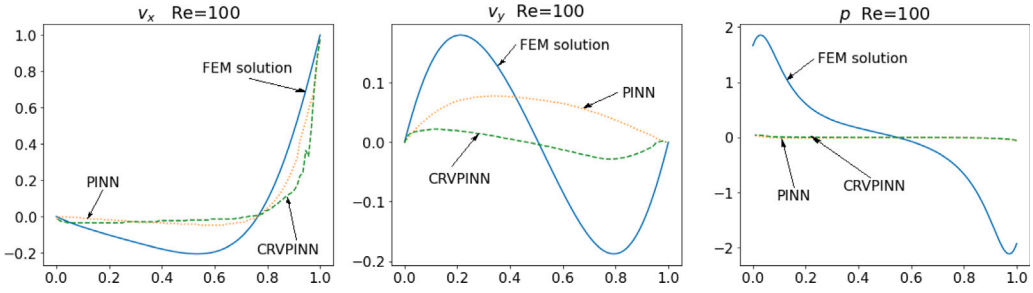


Fig. 7. Comparison of PINN, CRVPINN and FEM results for $Re = 100$. The first column represents v_x velocity component. The second column represents v_y velocity component. The third column represents pressure p.

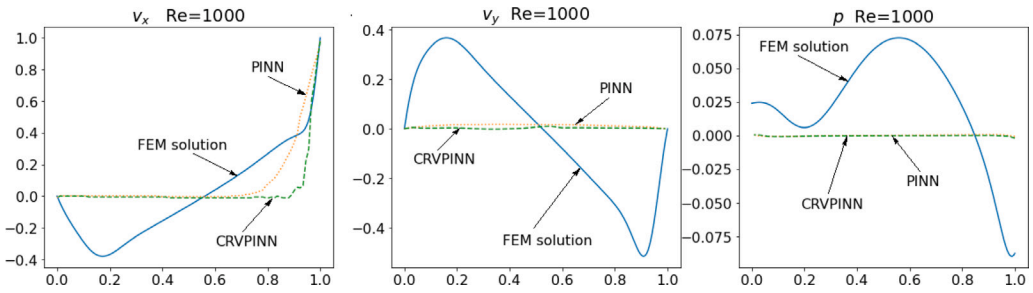


Fig. 8. Comparison of PINN, CRVPINN and FEM results for $Re = 1000$. The first column represents v_x velocity component. The second column represents v_y velocity component. The third column represents pressure p.

cavity problem, how can we trust AI-generated numerical simulations of challenging engineering problems?

7. Common opinion on reliable AI

In this chapter, we check how the quality of the PINN numerical simulations coincides with the societal trust in AI-generated numerical simulations.

The study presented in this article was conducted in two universities in Kraków, Poland. The students were selected primarily due to the nature of their studies (i.e., IT and AI orientation) and the assumption that they plan to become future professionals performing various scientific and engineering calculations in areas related, among others, to aerospace engineering, health monitoring, civil engineering, automotive engineering, or even tsunami prediction. The convenience sample of undergraduate and graduate students in computer science and applied computer science was used to learn how their current education on AI and individual experiences with AI in educational and professional dimensions shape their attitudes towards using engineering and scientific simulations and calculations performed by AI.

To collect data, an online survey was administered. The survey consisted of open-ended and closed-ended questions structured around the following topics on AI calculations and simulations: spontaneous associations, general attitudes, areas of application, contextual scenarios, AI self-sufficiency, advantages, and disadvantages. The survey was created and hosted in MS Forms. The survey link was sent to the instructors of the selected courses in both universities, who were asked to forward it to the course participants they were teaching. The research was conducted between October 10 and 24, 2025.

The sample consists of 209 undergraduate computer science students from AGH University of Kraków (N = 141) and applied computer science students from Jagiellonian University in Kraków (N = 68). Males represented the majority of the respondents (82%); however, the sex structure of the sample does not differ significantly from the general structure of science students in Poland, especially in computer science.[1] The average age was 22 years. More than half of the students (55%) did not participate in any courses on how to use AI in scientific calculations or simulations during their studies. They also declare that 53% can only partially indicate or apply methods for verifying calculations performed by AI. Nonetheless, some of them (17%) use engineering software that applies AI methods to scientific or engineering calculations, and more than one third (34%) are aware of such software but do not use any.

In general, from the performed surveys it implies that the attitude towards AI development and its applications that the students present can be characterized as cautious optimism. They recognize AI possibilities in increasing efficiency and innovation; however, this is accompanied by the claim that humans are the only verifiers and decision-makers. Their attitude is, therefore, balanced, as it is based on knowledge, reflection, and emotional distance from the risks associated with autonomous AI operations.

When asked whether they would replace human experts performing calculations and simulations with AI, the vast majority disagree with this idea (85%), which suggests that the students are inclined to trust AI as an analytical tool but do not trust it with full responsibility for decision making.

8. Source codes

The python code for running Physics Informed Neural Networks computations is available at

https://colab.research.google.com/drive/1LH0wLAda-19vCc_0FzlVeQvCZyiFECa3

The python code for running Collocation-based Physics Informed Neural Networks is available at

https://colab.research.google.com/drive/1pAFT4lfVVeNByQenrlKV_trmjyvZNFpf

The python code for running the Finite Element Navier–Stokes simulation is available at

`https://github.com/marcinlos/ns-fenicsx`

9. Conclusions

In this paper, we selected the lid-driven cavity model and verified the accuracy of AI-generated numerical simulations using the PINN method with a robust loss function. The motivation behind selecting the lid-driven cavity problem was the importance of the Navier–Stokes equations. They are employed in aerospace engineering for airplane safety verification; they are also used to predict tsunami propagation and to perform weather predictions.

We solved the lid-driven cavity problem for Reynolds numbers $Re = 1, 10, 100, 1000$ using a Finite Element Method code in FeniCs, and we used it as a reference solution. We compare the AI-generated numerical results from the PINN method. To estimate the quality of the numerical solution, we introduced the robust loss function that relates the value of the loss to the true error expressed in H^1 norm. To speed up the computations, we introduced the collocation method and discrete weak formulations.

The experiments presented in this paper show that AI-generated numerical simulations do not resolve the high-Reynolds-number flow well, even in the model lid-driven cavity problem for $Re = 100$ or higher. This questions the ability of AI-generated numerical simulations to model high-Reynolds-number flows accurately, which is crucial for simulations in aerospace engineering, deep-water tsunami modeling, and rapid atmospheric phenomena. Cautions and human expert verification of these numerical results are needed.

The optimistic conclusion from this paper is that there is good agreement between the cautious optimism of computer science students

[1] See for instance: https://op.europa.eu/webpub/eac/education-and-training-monitor/pl/country-reports/poland.html#notes-ref-25.

towards AI-generated numerical results and their actual accuracy. The cautious optimism mentioned multiple times in the paper is a consistent attitude represented by the computer science students. They notice the potential of AI in increasing efficiency, innovation, and precision in calculations; however, they emphasize the need to maintain human control and critical thinking. They consider AI a supporting tool, not a replacement for humans; they would trust AI with technical and repetitive tasks but not with decisions requiring responsibility and context. They identified speed, scalability, and error reduction as advantages, while threats include a lack of reliability and transparency, and the risk of turning a blind eye when trusting the technology. Their trust in AI depends significantly on context and risk; it is highest in areas such as weather forecasting, financial data analysis, and energy management, and lowest in situations involving physical security. They treat AI as an intelligent coworker whose effective and ethical activity depends on data quality and prudent human control.

CRediT authorship contribution statement

Tomasz Służalec: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Daria Wójcik:** Writing – review & editing, Writing – original draft, Visualization, Validation, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Carlos Uriarte:** Writing – review & editing, Writing – original draft, Validation, Methodology, Investigation, Formal analysis. **Marcin Łoś:** Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Anna Paszyńska:** Writing – review & editing, Writing – original draft, Investigation, Formal analysis, Data curation, Conceptualization. **Maciej Paszyński:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

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

Acknowledgments

The work is supported by the "Excellence initiative - research university" for AGH University of Krakow. The authors are grateful for the support from the funds that the Polish Ministry of Science and Higher Education assigned to AGH University of Krakow. The work has been supported by National Science Centre, Poland grant no. 2025/57/B/ST6/00058.

Data availability

Data will be made available on request.

References

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

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

[3] Ehsan Kharazmi, Zhongqiang Zhang, George Em Karniadakis, Hp-VPINNs: Variational physics-informed neural networks with domain decomposition, Comput. Methods Appl. Mech. Engrg. 374 (2021) 113547.

[4] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, George Em Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: A review, Acta Mech. Sin. 37 (12) (2021) 1727–1738.

[5] Zhiping Mao, Ameya D. Jagtap, George Em Karniadakis, Physics-informed neural networks for high-speed flows, Comput. Methods Appl. Mech. Engrg. 360 (2020) 112789.

[6] Yuyao Chen, Lu Lu, George Em Karniadakis, Luca Dal Negro, Physics-informed neural networks for inverse problems in nano-optics and metamaterials, Opt. Express 28 (8) (2020) 11618–11633.

[7] Chuang Liu, Heng An Wu, A variational formulation of physics-informed neural network for the applications of homogeneous and heterogeneous material properties identification, Int. J. Appl. Mech. 15 (08) (2023).

[8] Majid Rasht-Behesht, Christian Huber, Khemraj Shukla, George Em Karniadakis, Physics-informed neural networks (PINNs) for wave propagation and full waveform inversions, J. Geophys. Res.: Solid Earth 127 (5) (2022) e2021JB023120.

[9] Yuyao Chen, Lu Lu, George Em Karniadakis, Luca Dal Negro, Physics-informed neural networks for inverse problems in nano-optics and metamaterials, Opt. Express 28 (8) (2020) 11618–11633.

[10] Younghyeon Kim, Hyungyeol Kwak, Jaewook Nam, Physics-informed neural networks for learning fluid flows with symmetry, Korean J. Chem. Eng. 40 (9) (2023) 2119–2127.

[11] Chuang Liu, HengAn Wu, Cv-PINN: Efficient learning of variational physics-informed neural network with domain decomposition, Extrem. Mech. Lett. 63 (2023).

[12] Han Gao, Matthew J. Zahr, Jian-Xun Wang, Physics-informed graph neural Galerkin networks: A unified framework for solving PDE-governed forward and inverse problems, Comput. Methods Appl. Mech. Engrg. 390 (2022) 114502.

[13] Siddhartha Mishra, Roberto Molinaro, Estimates on the generalization error of physics-informed neural networks for approximating a class of inverse problems for PDEs, IMA J. Numer. Anal. 42 (2) (2022) 981–1022.

[14] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, Michael W Mahoney, Characterizing possible failure modes in physics-informed neural networks, Adv. Neural Inf. Process. Syst. 34 (2021) 26548–26560.

[15] Odus R. Burggraf, Analytical and numerical studies of the structure of steady separated flows, J. Fluid Mech. 24 (1) (1966) 113–151.

[16] James D. Bozeman, Charles Dalton, Numerical study of viscous flow in a cavity, J. Comput. Phys. 12 (3) (1973) 348–363.

[17] U. Ghia, K.N Ghia, C.T Shin, High-re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method, J. Comput. Phys. 48 (3) (1982) 387–411.

[18] E. Erturk, T.C. Corke, C. Gökçöl, Numerical solutions of 2-D steady incompressible driven cavity flow at high Reynolds numbers, Internat. J. Numer. Methods Fluids 48 (7) (2005) 747–774.

[19] Eric J. Nielsen, Boris Diskin, High-performance aerodynamic computations for aerospace applications, Parallel Comput. 64 (2017) 20–32, High-End Computing for Next-Generation Scientific Discovery.

[20] National Oceanic and Atmospheric Administration, Tsunami propagation, , 2023,.

[21] Peiwei Xie, Yan Du, Tsunami wave generation in Navier–Stokes solver and the effect of leading trough on wave run-up, Coast. Eng. 182 (2023) 104293.

[22] Maciej Paszyński, Leszek Siwik, Krzysztof Podsiadło, Peter Minev, A massively parallel algorithm for the three-dimensional Navier-Stokes-Boussinesq simulations of the atmospheric phenomena, in: Valeria V. Krzhizhanovskaya, Gábor Závodszky, Michael H. Lees, Jack J. Dongarra, Peter M.A. Sloot, Sérgio Brissos, João Teixeira (Eds.), Computational Science – ICCS 2020, Springer International Publishing, Cham, 2020, pp. 102–117.

[23] Sergio Rojas, Paweł Maczuga, Judit Muñoz-Matute, David Pardo, Maciej Paszyński, Robust variational physics-informed neural networks, Comput. Methods Appl. Mech. Engrg. 425 (2024) 116904.

[24] Marcin Los, Tomasz Sł użalec, Paweł Maczuga, Askold Vilkha, Carlos Uriarte, Maciej Paszyński, Collocation-based robust variational physics-informed neural networks (CRVPINNs), Comput. Struct. 316 (2025) 107839.

[25] Martin S. Alnaes, Jan Blechta, Johan Hake, August Johansson, Benjamin Kehlet, Anders Logg, Chris N. Richardson, Johannes Ring, Marie E. Rognes, Garth N. Wells, The FEniCS project version 1.5, Arch. Numer. Softw. 3 (2015).

[26] Youcef Saad, Martin H. Schultz, GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (3) (1986) 856–869.

[27] Patrick R. Amestoy, Alfredo Buttari, Jean-Yves L'Excellent, Theo Mary, Performance and scalability of the block low-rank multifrontal factorization on multicore architectures, ACM Trans. Math. Software 45 (1) (2019).

[28] Patrick R. Amestoy, Iain S. Duff, Jean-Yves L'Excellent, Jacko Koster, A fully asynchronous multifrontal solver using distributed dynamic scheduling, SIAM J. Matrix Anal. Appl. 23 (1) (2001) 15–41.

[29] Diederik P. Kingma, Jimmy Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[30] Igor A. Baratta, Joseph P. Dean, Jørgen S. Dokken, Michal Habera, Jack S. Hale, Chris N. Richardson, Marie E. Rognes, Matthew W. Scroggs, Nathan Sime, Garth N. Wells, DOLFINx: the next generation FEniCS problem solving environment, 2023, preprint.