Journal Pre-proof

On the choice of physical constraints in artificial neural networks for predicting flow fields

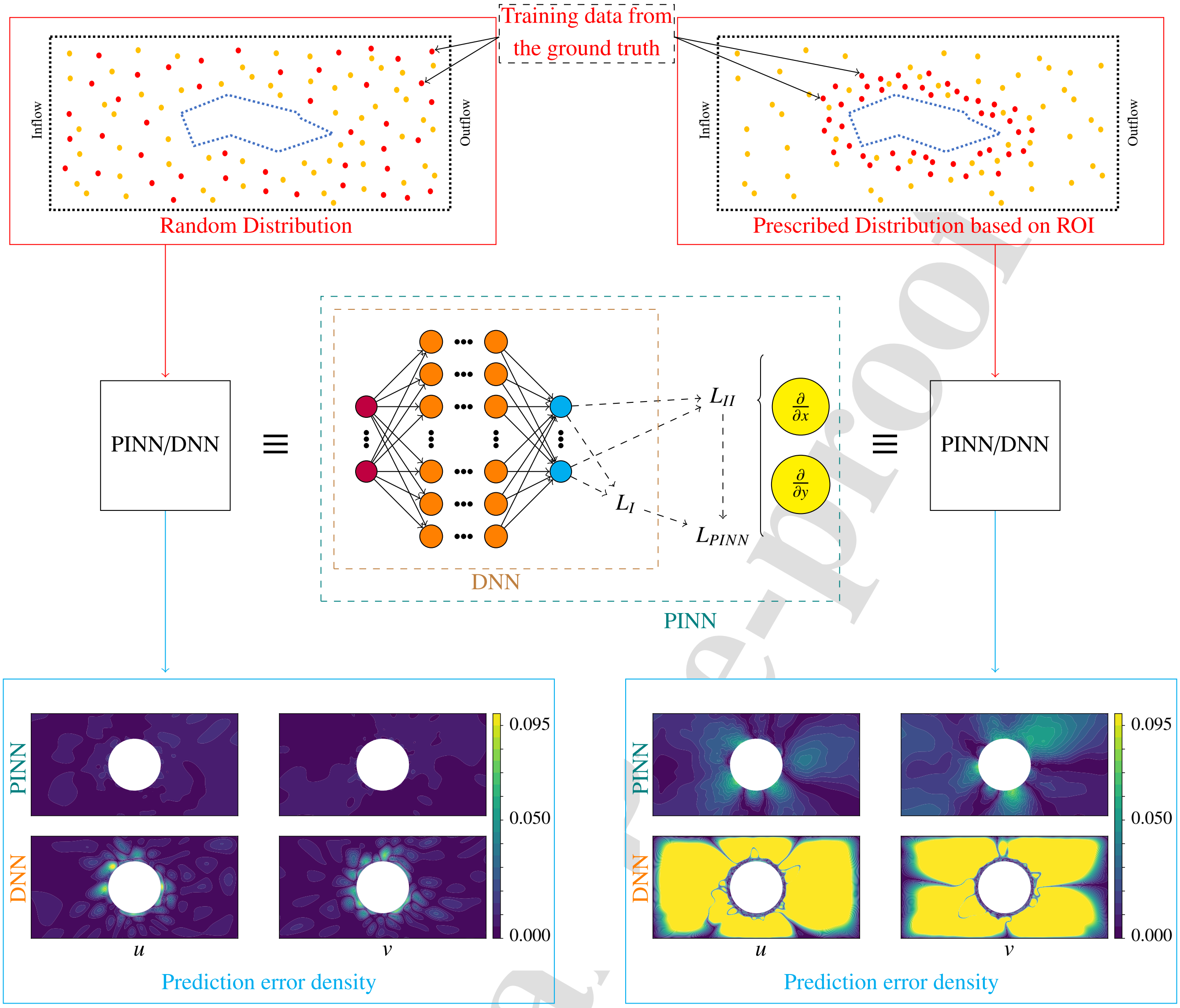
Rishabh Puri, Junya Onishi, Mario Rüttgers, Rakesh Sarma, Makoto Tsubokura, Andreas Lintermann

PII: S0167-739X(24)00372-8
DOI: https://doi.org/10.1016/j.future.2024.07.009
Reference: FUTURE 7431

To appear in: *Future Generation Computer Systems*

Received date : 31 December 2023
Revised date : 26 May 2024
Accepted date : 4 July 2024

Please cite this article as: R. Puri, J. Onishi, M. Rüttgers et al., On the choice of physical constraints in artificial neural networks for predicting flow fields, *Future Generation Computer Systems* (2024), doi: https://doi.org/10.1016/j.future.2024.07.009.



1 # Graphical Abstract

2 **On the Choice of Physical Constraints**
3 **in Artificial Neural Networks for Predicting Flow Fields**

4 Rishabh Puri, Junya Onishi, Mario Rüttgers, Rakesh Sarma, Makoto Tsubokura, Andreas Lintermann

Highlights

On the Choice of Physical Constraints in Artificial Neural Networks for Predicting Flow Fields

Rishabh Puri, Junya Onishi, Mario Rüttgers, Rakesh Sarma, Makoto Tsubokura, Andreas Lintermann

- Integration of governing physics significantly improved the prediction accuracy of the data-driven and data-free Artificial Neural Network for the potential flow cases investigated in this work

- Prediction accuracy of data-driven Artificial Neural Networks depends on the distribution of the ground truth in training and random distribution of training data has best performance amongst the different distributions studied in this work

- For an unsteady two-dimensional Taylor-Green vortex, which was trained using the Sequence-to-Sequence method, data-driven Artificial Neural Networks were able to interpolate in the temporal range and reconstruct the vortex structures for untrained time steps

On the Choice of Physical Constraints in Artificial Neural Networks for Predicting Flow Fields

Rishabh Puri[a,c,*], Junya Onishi[b], Mario Rüttgers[a], Rakesh Sarma[a], Makoto Tsubokura[b], Andreas Lintermann[a]

[a] *Jülich Supercomputing Centre, Forschungszentrum Jülich GmbH, Wilhelm-Johnen-Straße, Jülich, 52425, Germany*
[b] *RIKEN Center for Computational Science, 7-1-26, Minatojima-minami-machi, Chuo-ku, Kobe, 650-0047, Hyogo, Japan*
[c] *Engler-Bunte Institute, Combustion Technology, Karlsruhe Institute for Technology, Engler-Bunte Ring 7, Karlsruhe, 76131, Germany*

Abstract

The application of Artificial Neural Networks (NNs) has been extensively investigated for fluid dynamic problems. A specific form of ANNs are Physics-Informed Neural Networks (PINNs). They incorporate physical laws in the training and have increasingly been explored in the last few years. In this work, prediction accuracy of PINNs is compared with that of conventional Deep Neural Networks (DNNs). The accuracy of a DNN depends on the amount of data provided for training. The change in prediction accuracy of PINNs and DNNs is assessed using a varying amount of training data. To ensure the correctness of the training data, they are obtained from analytical and numerical solutions of classical problems in fluid mechanics. The objective of this work is to quantify the fraction of training data relative to the maximum number of data points available in the computational domain, such that the accuracy gained with PINNs justifies the increased computational cost. Furthermore, the effects of the location of sampling points in the computational domain and noise in training data are analyzed. In the considered problems, it is found that PINNs outperform DNNs when the sampling points are positioned in the Regions of Interest. The PINNs for the potential flow around the Rankine oval have shown better robustness against noise in training data compared to DNN. Both models show higher prediction accuracy when sampling points are randomly positioned in the flow domain as compared to a prescribed distribution of sampling points. The findings reveal new insights on the strategies to massively improve the prediction capabilities of PINNs with respect to DNNs.

Keywords: Physics-informed Neural Networks, Simplified Navier-Stokes equations, Partial differential equations, Fluid Dynamics, Unsteady flow

1. Introduction

Since the scientific revolution in the 16th and 17th centuries, scientists try to express nature in terms of equations. The dynamics of fluid flow is described through a set of Partial Differential Equations (PDEs), known as the Navier-Stokes equations [1, 2]. Although some simplified problems in fluid mechanics have analytical solutions, the solution to the Navier-Stokes equations can only be approximated using numerical methods that are solved in a discretized domain. The resolution required for these discretizations in space and time to sufficiently resolve the flow features increases with the complexity and parameters of the underlying flow, for instance with high Reynolds numbers.

In the second half of the 20th century, the advent of supercomputers provided a boost to the development of numerical methods and computational models to approximate fluid flow behaviour allowing large scale computations for real-world problems. Since then, the complexity of the Computational Fluid Dynamics (CFD) models and the capacity of the High-Performance Computing (HPC) systems has increased many times over. Depending on the order of accuracy of these CFD models, the solutions obtained by solving the temporally- and/or spatially-discretized governing equations, lead to varying errors in the computed flow fields. The desired accuracy determines the computational costs and hence highly-resolved simulations are expensive.

Artificial Neural Networks (ANNs) have the potential to complement, improve and even replace conventional CFD methods [3]. These deep learning-based NNs can further be categorised as data-driven or physics-informed. Data-driven Deep Neural Networks (DNNs) can be trained with spatial coordinates or temporal data of a domain as input to the network, where flow quantities such as the velocity or pressure fields, derived from analytical solutions, experimental results or CFD simulations, are used as ground truth [4]. Once trained, such purely data-driven DNNs can be employed to predict the velocity or pressure fields of the complete domain, while delivering results close to the reference data. These DNNs have to learn an approximation of the underlying physics while training on ground truth generated from flow solutions. Compared to the numerical solvers, these NNs can predict solutions faster [5]. For certain problems, they may, however, suffer from physical inconsistencies or violate the governing equations [6].

Different NN architectures can be employed for DNNs used in fluid mechanics. Convolutional Neural Networks (CNNs) are commonly used for data-driven solutions of problems in fluid

*Corresponding author
Email address: rishabh.puri@kit.edu (Rishabh Puri)

mechanics, which are solved mainly on cartesian grids [5, 7]. Owing to the filters, CNNs are able extract important multi-scale features from a large dataset. CNNs with encoder-decoder architechtures are used for evaluating steady state flow fields [5]. Matsuo et al. [8] used a combination of 2D and 3D CNNs to reconstruct a 3D flow field of flow around a square cylinder while training on sparse 2D data. Sekar et al. [9] proposed to train an encoder-decoder CNN to extract the geometric parameters of an airfoil while taking an image of a two-dimensional airfoil contour as input. The sequential model shows good prediction results while training on large CFD datasets. U-nets are also encoder-decoder based fully connected CNNs, where encoder and decoder layers are connected using skip connections [10]. By introducing skip connections in the fully connected layers, U-Nets are able to reproduce both high- and low-level features [11]. Generative models have enabled improved predictions of results not previously used for DNN training. Jolaade et al. [12] evaluated both a Generative Adversarial Network (GAN) and an Adversarial Auto-Encoder (AAE) for predicting the evolution in time of highly nonlinear fluid flow. The authors find that both models were able to predict the Gaussian vortices forward in time with AAE showing better results than GAN. To predict unsteady flow fields, Reduced Order Models (ROMs) have been commonly used with a Recurring Neural Network (RNN) or a Long Short-Term Memory (LSTM) as the propagator. Two Hybrid Reduced Order Models (ROMs) were presented by Bukka et al. [13] to predict unsteady flows. The first model uses the Proper Orthogonal Decomposition (POD) to project the high fidelity simulation data to a low dimension. The second model, referred to as the convolution recurrent autoencoder network (CRAN), employs convolutional neural networks with nonlinear activations, to extract the low-dimensional features. However, Fotiadis et al. [14] found that CNN based models have better performance than RNNs and LSTMs for predicting results for shallow water problems. Deep learning models with noisy trainng data can be used as an alternative to repetitive experiments. Sofos et al. [15] developed a CNN based deep learning model for reconstructing turbulent flow images from low-resolution counterparts encompassing noise.

CNNs suffer from a major drawback, that they can only be trained on data from uniform cartesian grids. This makes their application to most real world flow problems inefficient. For solving flow problems for complex geometries with irregular boundaries and unstructured grids, Graph Convolutional Neural Networks (GCNNs) can be implemented. Chen et al. [16] tested a GCNN as a surrogate model to predict flow around complex two-dimensional shapes on triangular unstructured grids. In comparison with U-Nets, the GCNN achieved better results, but it required more computation resources. For extrapolating the time-evolution of the flow in advection and incompressible fluid dynamics, Lino et al. [17] propossed two GCNN-based model architechures - multi-scale (MuS)-GNN and rotation-equivariant (RE) MuS-GNN. On complex flow domains, both models generalized high-gradient fields from uniform advection fields. The multi-scale approach provided a better approximation to the Navier-Stokes equations over a range of Reynolds numbers and design parameters as compared to the single-scale GCNNs.

The above discussed purely data-driven deep learning models require significant training data to predict results with good accuracy. Such large datasets are not always available. An alternative approach to potentially allow accurate training with sparse measurements is to integrate physical laws in the loss function of a DNN. In the case of fluid mechanics, these losses are based on the governing equations and include constraints given by initial and boundary conditions. This approach has the potential to drastically improve the predictive capability of the network [6]. Such learning models are referred to as Physics-Informed Neural Networks (PINNs).

Together with recent developments in automatic differentiation [18] and the availability of scattered partial spatio-temporal data for training, PINNs are capable of accurately and efficiently predicting solutions for fluid mechanics problems [4, 19]. Recently, PINNs have demonstrated their potential compared to conventional CFD methods with respect to computational efficiency and accuracy in solving certain PDEs [20–23]. The application of data-driven PINNs to the problems in fluid mechanics can be distinguished based on the implementation of constraints for initial/boundary conditions and on the collection of residuals from different spatial/temporal points in the flow domain. Using Graphics Processing Units (GPUs) and parallelizing the computation, the application of PINNs can be further expanded to more computationally demanding problems. For example, near-wall blood flows using only sparse data [24] or high-speed flows [25] can be predicted with this approach. Embedding the Navier-Stokes equations into an ANN allows the extraction of the pressure or velocity fields from experimental data. Raissi et al. [26] developed a Hidden Fluid Mechanics (HFM) model using a Physics-Informed deep learning approach to extract qualitative data from experimental results. The method is agnostic to the geometry, and to the initial and boundary conditions. Based on the complexity of the problem and the desired accuracy of the solution, hybrid models combining CFD solvers and PINNs have been developed, e.g., in [27]. Here, the flow solver Mantaflow [28] is coupled to a Convolutional Neural Network (CNN) for buoyant plume simulations at different Richardson numbers *Ri*. Ma et al. [29] implemented the Navier-Stokes equations and the boundary conditions in a U-Net architechture to predict steady flow fields. It was found that different flow regimes for flow around a cylinder could be learned and the adhered "twin-vortices" were predicted correctly. To predict solutions for a steady state natural convection problem for variable and complex geomtries, Peng et al. [30] proposed a Physics-Informed Graph Convolution Network (PIGN). The authors also compared the performance of the PIGN with a purely data-driven GNN model and found that PIGN had superior performance. The results demonstrated that the excellent geometric adaptability and prediction capability of a PIGN can be achieved with only limited training data and once fully trained, the model could solve natural convection problems with a lower computation time. Recently, deep neural networks with domain decomposition also have shown potential in solving differential equations efficiently [31, 32].

Jagtap et al. [33] proposed a conservative PINN (cPINN) based on the domain decomposition method for solving forward and inverse problems.

PINNs can also be trained in a data-free manner, i.e., the training data does not contain any ground-truth data from analytical solutions or CFD simulations, except for the data from initial or boundary conditions [34]. Grimm et al. [35] implemented the governing physics in a U-Net using the discretization approaches of a Finite Difference Method. The authors found that a physics aware data-free model generalised better than a data-driven model, while predicting steady flow fields around random geometries for low inlet velocities. However, Chuang et al. [36] observed that such data-free PINNs can be difficult to train and lack temporal information, i.e., yielding solely steady state solutions.

The previously mentioned studies focus solely on the capability of predicting flow fields with deep learning methods, without considering the sparsity of data for different flow applications. Investigating the training data-dependency of deep learning models can be useful for real world problems, where large training datasets are not available. For example, the development of a car body in the automobile industry is usually supported by CFD simulations and wind tunnel experiments [37, 38]. However, although these techniques are capable of correctly predicting force coefficients or regions of flow separation [39], they are limited in reproducing real conditions like weather, driving style, or the road surface.

In contrast to wind tunnel experiments, collecting on-road data enables to reveal more complex flow structures related to such real conditions, e.g., increased flow unsteadiness in the region of the A-pillar vortex implying noise generation [40] can be analyzed for varying on-road conditions. Real-time surface sensors could capture the performance of a driving prototype vehicle [41], and DNNs could be trained with these measurements to predict the surrounding flow fields. Such surface sensors can only be installed sparsely and hence their number and strategic placement is of great importance. Furthermore, the integration of the governing physics with loss constraints could be essential for improving such predictions. Notably, the calculation of additional physical losses in PINNs may result in higher computational demands. Depending on the complexity of the flow problem, the application of PINNs may not be justified over employing in general cheaper-to-train DNNs that may provide similarly accurate and physically plausible solutions.

In this regard, the number and placement of the following types of data sources are discussed in this investigation:

(i) domain points with a corresponding ground truth (data-driven) and

(ii) domain points without ground truth (data-free).

The study assesses the performance of PINNs and conventional DNNs with respect to variations in the number of the these types of data sources. The goal is to demonstrate and quantify the amount and location of training data that justifies the use of PINNs over conventional DNNs in terms of prediction accuracy for different flow configurations. For this purpose, the following flow configurations are considered.

- Potential flow,
- a boundary layer flow based on the Blasius equation, and
- a Taylor-Green Vortex.

The ground truth data for the different flow configurations in this study are obtained using analytical and numerical methods. The ground truth is also used to validate the ANN-predicted flow fields. Throughout the manuscript, ANN nomenclature is used to refer to both PINN and DNN.

Given that the objective of this study is to analyze the effect of physical constraints, training data concentration in the spatial domain and noisy training data for individual flow scenarios, fully-connected feed forward neural network architectures are used to compare the performance of PINNs with that of DNNs.

The findings are expected to contribute to a more efficient use of PINNs in fluid dynamics and potentially extend its application to real-world flow problems such as in vehicle aerodynamics. The manuscript is structured as follows. In Section 2, the flow configurations are described and details about the training and test data are provided. The DNNs and PINNs are introduced. Subsequently, the network-predicted flow fields are compared to the analytic solutions in Section 3. Finally, the findings are summarized, conclusions are drawn, and an outlook is given in Section 4.

2. Methods

In this section, the theoretical backgrounds of the computations are described. Section 2.1 provides information about the flow configurations considered in this work. This includes the governing equations as well as the boundary and initial conditions used for solving the equations. In Section 2.2, the architecture, parameters, and basic loss functions of the DNNs are described, and the physical loss functions that extend the DNNs to PINNs are explained.

2.1. Flow configurations

The governing equations, spatial domains, and boundary conditions of the two-dimensional flow problems investigated in this study are described in what follows.

Potential flow

A potential flow is defined as a steady, incompressible, inviscid, and irrotational flow around a body. The velocity field $\vec{u} = (u, v)^T$ is described by the gradient of a scalar function called the potential function ϕ, given by

$$\vec{u} = \nabla\phi. \tag{1}$$

Here, u represents the velocity component in the x-direction, and v in the y-direction. The orientation of the directions are illustrated in Figure 1. The condition for irrotational flow, i.e., $\nabla \times \vec{u} = 0$, is satisfied by $\nabla \times \nabla\phi = 0$. The continuity equation for incompressible flows $\nabla \cdot \vec{u} = 0$ yields the first governing equation for potential flows, given by

$$\nabla \cdot \nabla\phi = \Delta\phi = 0. \tag{2}$$

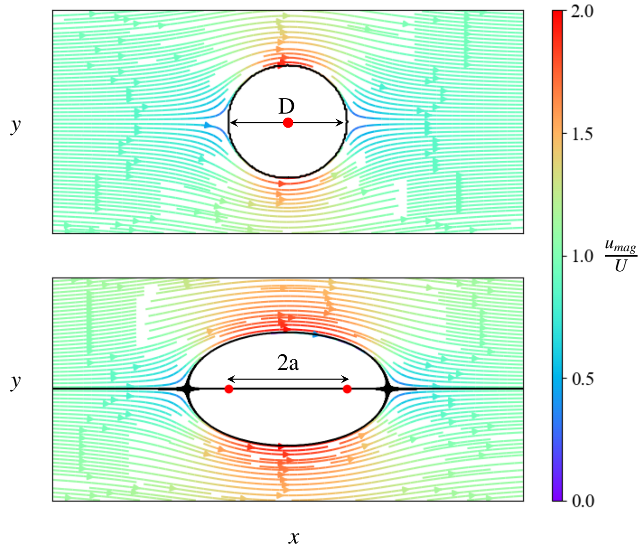


Figure 1: Streamlines of potential flow around a cylinder (top) and Rankine oval (bottom), colored by the normalized velocity magnitude u_{mag}/U.

Further governing equations based on the stream function ψ are

$$u = \frac{\partial \psi}{\partial y}, \qquad v = -\frac{\partial \psi}{\partial x}. \tag{3}$$

These equations fulfill the continuity equation and the condition for irrotational flows yields the second governing equation for potential flows, i.e.,

$$\Delta \psi = 0. \tag{4}$$

Figure 1 shows the two potential flow configurations investigated in this study, i.e., the potential flow around a circular cylinder with diameter D and around a Rankine oval. Both domains are characterized by a uniform inflow with velocity $\vec{u} = (U, 0)^T$, a source, and a sink. The length of the fluid domain in case of the circular cylinder is $4D$ and $2D$ in the x- and y-direction, and the source and sink have the same center. In case of the Rankine oval, they are separated by a distance of $2a$. Here, the length of the fluid domain is $8a$ and $5a$ in the x- and y-direction. The velocity fields in Figure 1 show the velocity magnitude u_{mag}, normalized by U. The potential and stream functions read

$$\phi = Ux + \frac{Q}{\pi} \cdot \frac{x}{x^2 + y^2}, \tag{5}$$

$$\psi = Uy - \frac{Q}{\pi} \cdot \frac{y}{x^2 + y^2} \tag{6}$$

for the circular cylinder, and

$$\phi = Ux + \frac{m}{4\pi} \cdot \log\left[\frac{(x+a)^2 + y^2}{(x-a)^2 + y^2}\right], \tag{7}$$

$$\psi = Uy - \frac{m}{2\pi} \cdot \tan^{-1}\left(\frac{2ay}{x^2 + y^2 - a^2}\right) \tag{8}$$

for the Rankine oval. The strength of the source and sink are given by $Q = \pi(D/2)^2 U$ for the cylinder, and m for the Rankine oval.

To calculate the flow field, the fluid domain is discretized using a structured grid with cell spacing $\Delta_{pot,c} = D/80$ for the circular cylinder and $\Delta_{pot,R} = a/100$ for the Rankine oval.

Blasius boundary layer flow

The boundary layer equations for a flat plate of length L_b are derived from the Navier-Stokes equations by using Prandtl's boundary layer approximation [42]. The important assumptions are a high Reynolds number $Re \gg 1$ and attached flow, i.e., there is no flow separation. The effects of viscosity are only limited to a thin layer of width δ near the surface of the body, which is oriented normal to the plate. Considering a zero pressure gradient, the boundary layer equations are given by

$$\frac{\partial \rho u}{\partial x} + \frac{\partial \rho v}{\partial y} = 0, \tag{9}$$

$$\rho u \frac{\partial u}{\partial x} + \rho v \frac{\partial u}{\partial y} = \frac{\partial}{\partial y}\left(\mu \frac{\partial u}{\partial y}\right) \tag{10}$$

$$\frac{\partial p}{\partial x} = 0, \frac{\partial p}{\partial y} = 0, \tag{11}$$

where ρ is the density of the fluid and μ is the dynamic viscosity, with x and y being oriented parallel and orthogonal to the plate respectively.

In the scope of this study, the velocity field of the flat plate boundary layer equations is predicted using ANNs. The basic criteria for Blasius' solution was to transform the above system of PDEs to a single ODE by using coordinate transformation [43]. To find a self-similar solution, where the solution should not change if an independent and dependent variable are scaled appropriately, the dependent variable f is defined. The quantity f is related to the stream function ψ and a function of the independent variable η.

Based on the boundary layer thickness δx, η is defined as:

$$\eta \sim \frac{y}{\delta(x)} = \frac{y}{(\nu x/U_0)^{1/2}}. \tag{12}$$

This is known as the scaled form of the stream function, where $\nu = \mu/\rho$ is the kinematic viscosity. The velocity components in the x- and y-direction are scaled by U_0 by

$$\tilde{u} = \frac{u}{U_0}, \qquad \tilde{v} = \frac{v}{(\nu U_0/x)^{1/2}}. \tag{13}$$

From the above equations, a scaled stream function is obtained by

$$f(\eta) = \frac{\psi}{(\nu x U_0)^{1/2}}. \tag{14}$$

The velocity components can now be expressed in terms of the scaled stream function as

$$u = U_0 \frac{df}{d\eta}, \tag{15}$$

$$v = \frac{1}{2}\sqrt{\frac{\nu U_0}{x}}\left(\eta \frac{df}{d\eta} - f\right). \tag{16}$$

Inserting these values in the governing Eqs. (9), (10), and (11), and after some simplifications, the following ODE is obtained

$$\frac{d^3 f}{d\eta^3} + \frac{1}{2} f \frac{d^2 f}{d\eta^2} = 0, \tag{17}$$

which is the final form of the Blasius boundary layer equation for flows over a flat plate. At the wall, no-slip boundary conditions are prescribed by setting $u(y = 0) = v(y = 0) = 0$, and at $y \geq \delta$ the velocity becomes the free stream velocity,

$$f(\eta = 0) = 0, \tag{18}$$
$$f'(\eta = 0) = 0, \tag{19}$$
$$f'(\eta \to \infty) = 1. \tag{20}$$

In this equation, $f' = df/d\eta$.

Taylor-Green Vortex

The Taylor–Green vortex is an unsteady flow of a decaying vortex, for which a complete solution of the incompressible Navier-Stokes equations will suffice to illustrate the process of dissipation of large eddies into smaller ones. An attempt was made by Taylor et al. [44] to obtain a solution for the subsequent motion of the viscous incompressible fluid, when the initial solution in Cartesian coordinates is given by

$$u = A(\cos ax)(\sin by)(\sin cz), \tag{21}$$
$$v = B(\sin ax)(\cos by)(\sin cz), \tag{22}$$
$$w = C(\sin ax)(\sin by)(\cos cz), \tag{23}$$

where w is the velocity component in the z-direction. The equations described above are consistent if

$$Aa + Bb + Cc = 0. \tag{24}$$

The governing equations for a two-dimensional Taylor-Green vortex are given by

$$\nabla \cdot \vec{u} = 0, \tag{25}$$
$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = \frac{1}{\rho} \nabla \cdot \bar{\bar{\sigma}}, \tag{26}$$

where Eq. (25) is the continuity equation and Eq. (26) defines the Cauchy momentum equation. Here, the quantity $\bar{\bar{\sigma}}$ is the the viscous stress tensor for incompressible flow given by

$$\bar{\bar{\sigma}} = -p\bar{\bar{I}} + \mu(\nabla\vec{u} + (\nabla\vec{u})^T), \tag{27}$$

where p stands for the pressure and $\bar{\bar{I}}$ for the identity tensor. According to Taylor's analysis and for the condition:

$$A = a = b = 1, \tag{28}$$

the analytical solution for a two-dimensional vortex is given by

$$u = \cos x \sin y F(t), \tag{29}$$
$$v = -\sin x \cos y F(t), \tag{30}$$
$$p = -\frac{\rho}{4}(\cos 2x + \sin 2y) F^2(t), \tag{31}$$

where $F(t) = e^{-2\nu t}$ and t represents the time. Figure 2 gives an example of the analytical initial solution. The analytical solutions from Eqs. (29) to (31) are used for generating training data.

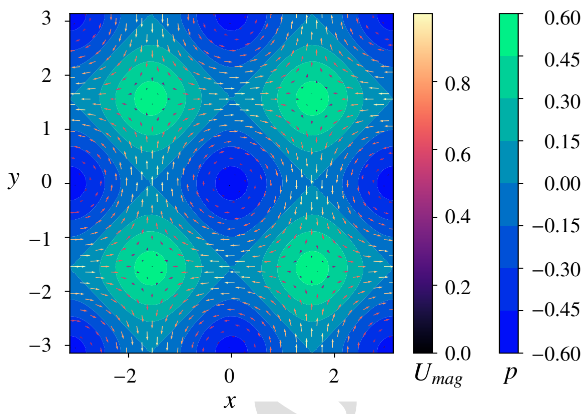


Figure 2: 2D Taylor-Green Vortex at $t = 0$.

2.2. Architecture of the ANNs

A fully-connected feed forward network architecture is used for every problem in this work and the hyperbolic tangent (tanh) activation function [45] is used for the hidden and output layers. The random search method is used forhyperparameter tuning. Figure 3 provides a general example of network architectures and loss functions for DNNs and PINNs. The neurons of the input layer and the output neurons are colored in red and blue. The DNN has only one loss function L_I, which is the

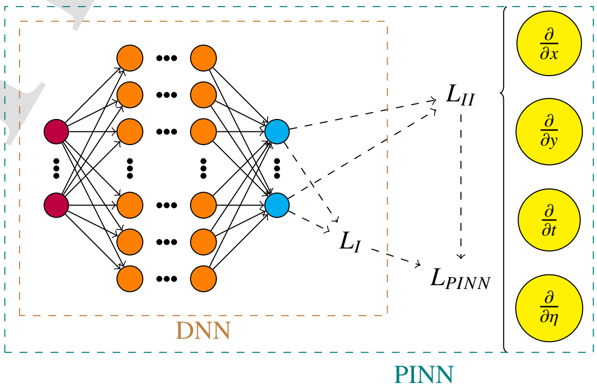


Figure 3: Architecture of a generic DNN and PINN.

Mean-Squared Error (MSE) between the DNN predictions and the ground truth. In the PINN case, further losses L_{II} for the governing equations are also included. For L_{II}, the differentials with respect to the input variables, as shown by the yellow circles in Figure 3, are calculated using the automatic differentiation functionalities of PyTorch [1]. That is, `autograd` methods like `grad` and `jacobian` are used in the loss functions for residuals of the governing equations.

The flow-specific inputs and outputs are shown in Table 1. For the two potential flow cases, the inputs are the Cartesian coordinates (x, y). The outputs are the 2D velocity field in the

[1] Torch version 2.0.1+cu117

Flow case	Input	Output
Potential flow	x, y	u, v
Blasius equation	η	f, f'
Taylor-Green Vortex	x, y, t	$\sigma_{xx}, \sigma_{xy}, \sigma_{yy}$, u, v, p

Table 1: Input and output of the ANNs for each flow configuration.

in x- and y-directions. The input for the Blasius boundary layer flow is the independent variable η given by Eq. (12), instead of the 2D Cartesian coordinates that are used in the other cases. The reason for this is the fact that the scaled stream function f depends only on η, cf. Section 2.1. The output of the network is the scaled stream function and its first derivative. To obtain a predicted velocity field, the output values are derived from the equations in Eq. (13). For the unsteady flow case of the two-dimensional Taylor-Green vortex, time t along with the Cartesian coordinates are the inputs to the ANN. The outputs are defined by the velocity and pressure fields as well as by the components of the viscous stress tensor $\bar{\bar{\sigma}}$.

Each data point defines an input-output pair and solutions are generated for N_{total} data points. The losses L_I and L_{II} depend on the types of data points of each flow configuration. All data points are a subset of N_{total} defined for each problem. Figure 4 provides a general example of the different types of data points for a two-dimensional flow around an arbitrary shape. Points extracted from domain boundaries N_b are expressed by black dots. If there is flow around an object, e.g., the blue obstacle in Figure 4, the losses include wall points N_w, which are represented by the blue dots at the shape's contour. The yellow data points in the flow domain away from the boundaries are denoted as N_d. The domain points N_d together with N_b and N_w (if there is an object) are used to calculate the residual loss. They are kept fixed for each training run. A subset of N_d, i.e., $N_{d,1}$, represented by the red dots in Figure 4, and its corresponding ground truth data from analytical solutions is varied for each training run. These variations are defined by the fraction $\mathcal{F}$

$$\mathcal{F} = \frac{N_{d,1}}{N_d}. \tag{32}$$

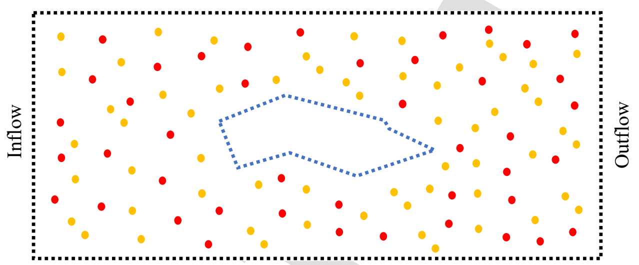


Figure 4: Distribution of data points for a general example of a two-dimensional flow around an arbitrary shape. The boundary and wall points N_b and N_w are shown with black and blue dots, and the domain points N_d with yellow dots. All these points are kept fixed for each training run. The variable data points with existing ground truth data, $N_{d,1} \subseteq N_d$ are denoted by the red dots.

For the potential flow problems, the residual loss from the governing equations is embedded into the total loss for all boundary subdomains and for a set of random points in the fluid domain. The physical loss function used in training the potential flow PINNs is defined by

$$L_{II,pot} = \frac{1}{N_{pot}}\left[\sum_{n=1}^{N_{pot}} \mid \nabla \cdot \vec{u}_n \mid^2 + \sum_{n=i}^{N_{pot}} \mid \nabla \times \vec{u}_n \mid^2\right], \tag{33}$$

where $N_{pot} = N_b + N_w + N_d$. The prediction loss against the exact solution is given as

$$L_{I,pot} = \frac{1}{N_{pot,1}} \sum_{n=1}^{N_{pot,1}} \left|\vec{u}_n - \vec{u}_n^*\right|^2, \tag{34}$$

where $N_{pot,1} = N_b + N_w + N_{d,1}$. For each point n, $\vec{u}_n^*$ is the exact velocity vector and $\vec{u}_n$ is the predicted velocity vector.

For the Blasius boundary layer flow, the physical loss of Eq. (17) is defined by

$$L_{II,bl} = \frac{1}{N_{bl}} \sum_{n=i}^{N_{bl}} \left|f_n''' + \frac{1}{2} f_n f_n''\right|^2, \tag{35}$$

where the total number of data points $N_{bl} = N_b + N_d$, with N_d including η away from the boundaries, and N_b represents η at the boundaries. The quantity N_{bl} is kept fixed for each training run. The prediction loss for the Blasius flow problem is given by

$$L_{I,bl} = \frac{1}{N_{bl,1}} \sum_{n=i}^{N_{bl,1}} \mid f_n - f_n^* \mid^2, \tag{36}$$

where f_n^* is the exact value of the scaled stream function from the numerical solution and $N_{bl,1} = N_b + N_{d,1}$.

In case of the two-dimensional Taylor-Green vortex, Sequence-to-Sequence (S2S) training is implemented. The schematic for the S2S training is shown in Figure 5, which is based on the backward-compatible sequence training model implemented by Mattey et al. [46] for time-dependent PDEs. The training data is calculated for specific time-steps defined by the time step size Δt. The size of the spatial domain for all time steps is the same. Starting from $t = 0$, the ANN is sequentially trained for each time step, and training is restarted when a stopping criteria is met. The stopping criteria is defined by either the maximum number of epochs or a specified training loss value. This process is continued until the final time step $t = T$ is reached, where the physical loss in training is augmented by the prediction loss from all time steps between $t = 0$ and $t = T_{N-1}$. The physical loss for a certain time step, $t = t_i$ is

$$L_{II,tgv} = \frac{1}{N_{tgv}}\left[\sum_{n=1}^{n=N_{tgv}} \left|\nabla \cdot \vec{u}_n\right|^2 + \right. \tag{37}$$

$$\sum_{n=1}^{n=N_{tgv}} \left|\frac{\partial \vec{u}_n}{\partial t} + \vec{u}_n \cdot \nabla \vec{u}_n - \frac{1}{\rho} \nabla \cdot \bar{\bar{\sigma}}_n\right|^2 +$$

$$\left. \sum_{n=1}^{n=N_{tgv}} \left|\bar{\bar{\sigma}}_n + p_n \bar{\bar{I}} - \mu(\nabla \vec{u}_n + (\nabla \vec{u}_n)^T)\right|^2\right], \tag{38}$$

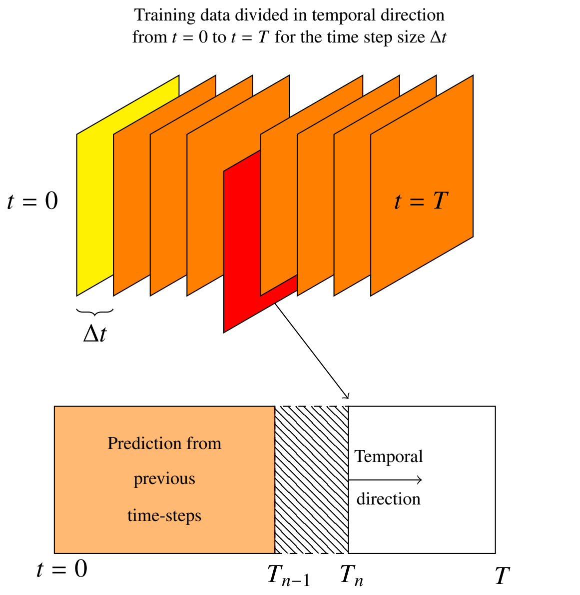


Figure 5: Sequence to Sequence (S2S) training for unsteady flow problems.

where $N_{tgv} = N_b + N_d$ defines the spatial data points. The prediction loss for the data-driven training is given by

$$L_{I,tgv} = L_{tgv,u} + L_{tgv,p} + L_{tgv,u'} + L_{tgv,p'} \tag{39}$$

with

$$L_{tgv,\vec{u}} = \frac{1}{N_{tgv,1}} \sum_{n=1}^{n=N_{tgv,1}} \mid \vec{u}_n - (\vec{u}_n)^* \mid^2, \tag{40}$$

$$L_{tgv,p} = \frac{1}{N_{tgv,1}} \sum_{n=1}^{n=N_{tgv,1}} \mid p_n - p_n^* \mid^2, \tag{41}$$

$$L_{tgv,\vec{u}^t} = \frac{1}{N_{tgv,2}} \sum_{n=1}^{n=N_{tgv,2}} \mid \vec{u}_n^t - (\vec{u}_n^t)^* \mid^2, \tag{42}$$

$$L_{tgv,p^t} = \frac{1}{N_{tgv,2}} \sum_{n=1}^{n=N_{tgv,2}} \mid p_n^t - (p_n^t)^* \mid^2, \tag{43}$$

where $N_{tgv,1} = N_b + N_{d,1}$ defines the data points in space at t_i. Similarly, $N_{tgv,2}$ are the training points from previously trained time steps. For each training point in the prediction loss, the superscript $(*)$ defines the exact solution and the superscript (t) defines the solution from the previous time steps.

Weights and biases of the models are updated by an Adaptive Moment Estimation (ADAM) [47] or Stochastic Gradient Decent (SGD) optimizer [48]. For all investigated flow cases, both, the input to the ANN and the ground truth, are used without any normalization.

3. Results

In this section, the computation cost is analysed using the training time of PINNs and DNNs. Additionaly, the performance of PINNs and DNNs is analyzed in terms of their prediction accuracy for variations of $\mathcal{F}$. The qualitative results for each case are shown for certain selected values of $\mathcal{F}$. The basis for this selection is the difference in the performance of PINNs and DNNs for each problem at the training data points defined by $\mathcal{F}$. While training the PINNs, 50% of the domain points are used for the physical loss, which is kept constant along with the boundary points. The location for these points remains unchanged during training the multiple cases. For all cases, a 80 : 20% data split is used to distribute between training and testing datasets. Hyperparameter tuning is performed for the PINN models and the selected hyperparameters are also used for the respective DNN models.

The models are trained on the GPU partition of the JURECA-DC cluster [49] installed at the Jülich Supercomputing Centre (JSC), Forschungszentrum Jülich. Each node is equipped with four NVIDIA A100 GPUs and two AMD EPYC 7742 CPUs with 64 cores clocked at 2.25 GHz. The results presented in the following are for deterministic training with the same parameters for both the PINN and DNN. Additionally, these results are verified by randomly initializing the PINN and DNN individually and checking their performance.

The model performance is evaluated using the prediction accuracy for the complete flow field. Errors are quantified by juxtaposing the results of the ANNs to the exact solutions, which are obtained analytically or numerically. The parameter chosen for evaluating the prediction accuracy is calculated as a relative Euclidean norm (L_2) error given by

$$\epsilon_\phi = \frac{\sqrt{\sum_{x,y,\eta} \mid \phi(x,y,\eta) - \phi_e(x,y,\eta) \mid^2}}{\sqrt{\sum_{x,y,\eta} \mid \phi_e(x,y,\eta) \mid^2}}. \tag{44}$$

Here, $\phi = u, v, p, \sigma_{xx}, \sigma_{xy}, \sigma_{yy}, f, f'$, and ϕ_e is the exact value of the corresponding output variable. Both ϕ and ϕ_e are calculated for N_{total} grid points for every flow case.The performance of the NNs during the training is evaluted with the L_2 error for the testing dataset and the prediction error of a trained model is calculated as the L_2 error for the complete flow domain.

3.1. Potential flow: Cylinder

A two-dimensional uniform grid is generated using the `meshgrid` function in the `NumPy`[2] module of Python. The cell size is set to 0.0125D. The grid has N_{total} = 46,600 data points of which N_b = 964 are located at the domain boundary and N_w = 235 are located on the cylinder wall. The rest of the data points are uniformly distributed within the flow domain. The domain points that have corresponding ground-truth data are varied from $\mathcal{F}$ = 0.05 to $\mathcal{F}$ = 0.8. Both PINNs and DNNs are trained with 6 hidden layers and each hidden layer has 60 neurons. The ADAM optimizer is used with a learning rate of LR = 0.0005. The models are trained on a single GPU for 20,000 epochs.

As shown in Figure 6, a computational cost analysis between the PINNs and DNNs for different values of $\mathcal{F}$ is performed using the L_2 error curve from the testing data against training

[2]NumPy version 1.25.2

time required to train 20,000 epochs. The training time required for PINN is six times more than that of the DNN. The progression of the training error shows similar trends for the two tested values of $\mathcal{F}$ for PINN as well as for DNN. At 20,000 epochs, both DNNs have a similar L_2 testing error in comparison to the respective PINNs, but the following qualitative analysis highlights the higher accuracy of the PINN. The change in the prediction error with variation in $\mathcal{F}$ is shown in Figure 7. For all $\mathcal{F}$, the PINN performs better than the DNN.

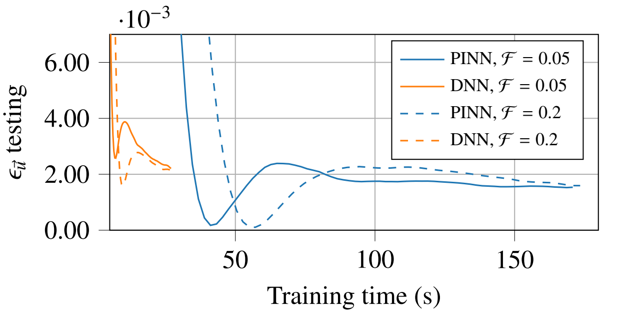


Figure 6: L_2 testing error versus training time for potential flow around a cylinder.

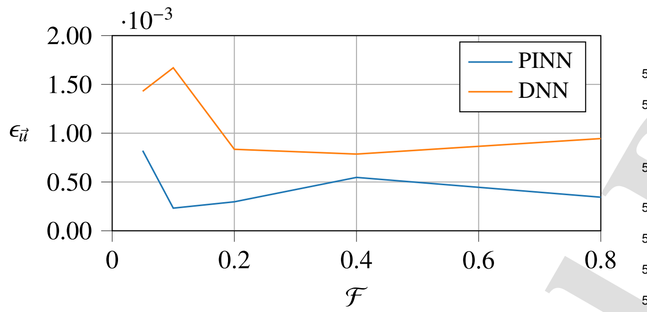


Figure 7: L_2 prediction error for a varying $\mathcal{F}$ for the potential flow around a cylinder.

The absolute error density in prediction of the velocity field for both models is shown for $\mathcal{F} = 0.05$ in Figure 8(a), and for $\mathcal{F} = 0.2$ in Figure 8(b). Comparing the results of Figure 8(a), it can be deduced that for an equal number of training epochs and the same hyperparameters, the DNN fails to accurately predict both x- and y-components of the velocity field in the vicinity of the cylinder wall. In contrast, the PINN-based predictions show improved predictions for the overall flow fields.When the number of training data with ground truth is increased from $\mathcal{F} = 0.05$ to $\mathcal{F} = 0.2$, both ANNs predict the flow around the cylinder better, as it is visible in Figure 8(b). However, this improvement is reflected differently for the PINN and DNN. The DNN, missing associated physics in the loss function, cannot accurately predict the velocity field near the cylinder wall, whereas the PINN outputs show higher accuracy. These results underline the clear superiority of data-driven PINN models for predicting the potential flow around a cylinder. However, this gain in the prediction accuracy with the PINN is achieved with a comparatively higher training time. For instance, with

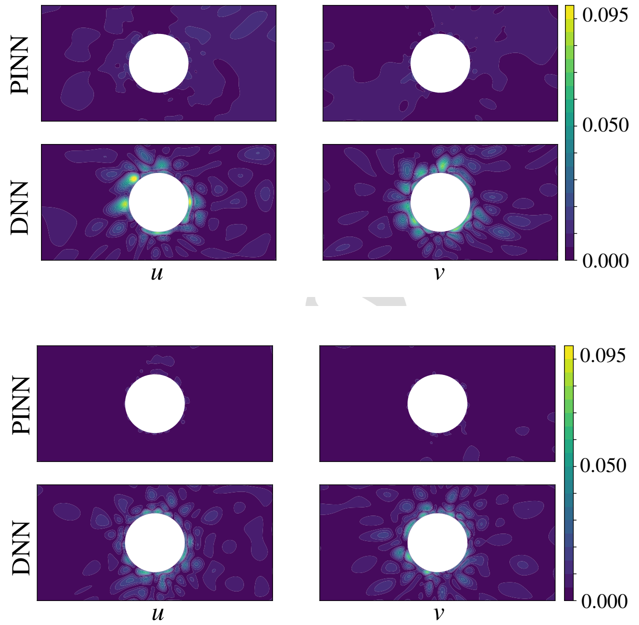


Figure 8: Error density of the predicted velocity field for a potential flow around a cylinder with (a) $\mathcal{F} = 0.05$ and (b) $\mathcal{F} = 0.2$.

$\mathcal{F} = 0.05$, the prediction error of PINN is almost half to that of DNN, but PINN has six times longer training time.

3.2. Potential flow: Rankine oval

To resolve the Rankine oval flow, a total number of N_{total} = 332,616 uniformly distributed spatial data points are used, of which N_b = 2,600 are on the boundaries of the domain and N_w = 1,592 are on the Rankine oval boundary. Similar to the previous case, the training data points on the boundary are kept fixed. The included ground truth for the data-driven training is varied with $\mathcal{F}$ as a percentage of the domain data points used for the physical loss. Eqs. (33) and (34) define the loss functions for the training models with and without integrated physics. Both models have 5 hidden layers and each hidden layer has 60 neurons. The ADAM optimizer with a learning rate $LR = 0.0005$ is used for all training runs. All models are trained on a single GPU for 20,000 epochs.

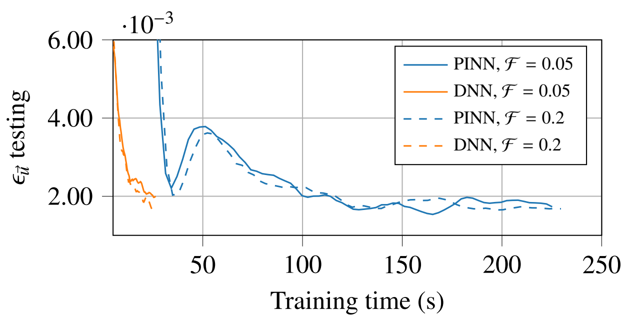


Figure 9: L_2 testing error versus training time for potential flow around a Rankine oval.

It can be observed from the L_2 testing error plot in Figure 9 that the DNN is able to achieve a similar performance as the PINN. An increase in the training data from $\mathcal{F} = 0.05$ to $\mathcal{F} = 0.2$ results in an increase of 5 s and 1 s in training time of the PINN and DNN respectively. For both $\mathcal{F}$ values, the training time of the PINN is almost nine times larger than the training time of the DNN.

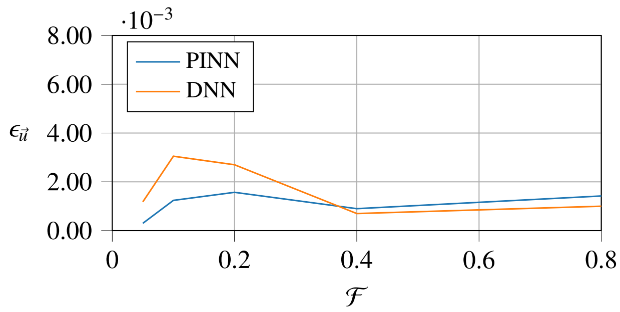


Figure 10: L_2 prediction error for a varying $\mathcal{F}$ for the potential flow around a Rankine oval.

The change in the prediction error under variation of $\mathcal{F}$ is shown in Figure 10. Predictions from PINNs provide higher accuracy up to $\mathcal{F} = 0.38$ compared to DNNs, while the latter performs slightly better for higher values of $\mathcal{F}$. However, the L_2 error for both Rankine models first increases with $\mathcal{F}$ and then drops until $\mathcal{F} = 0.4$ is reached. The largest gap between the two types of ANNs is observed for $\mathcal{F} = 0.1$. Given the higher number of N_{total} data points, both models already have more training data available than the cylinder case for each $\mathcal{F}$ value.

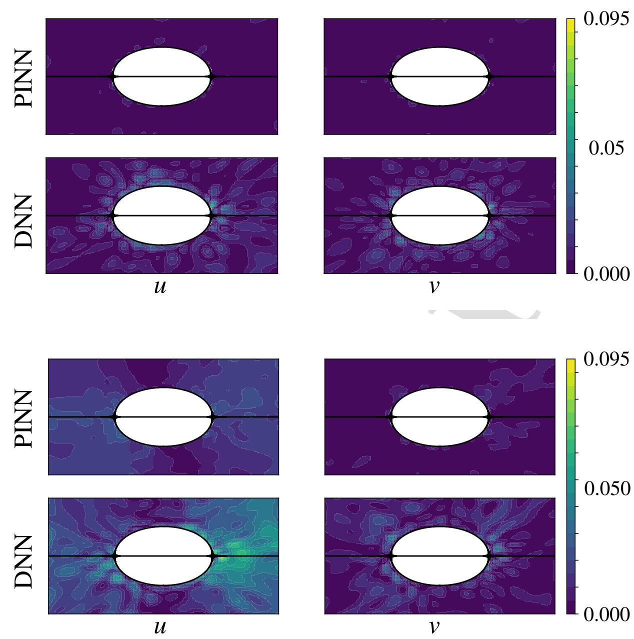


Figure 11: Error density of the predicted velocity field for a potential flow around a Rankine oval using (a) $\mathcal{F} = 0.05$ and (b) $\mathcal{F} = 0.2$ for the training.

The density plots for the absolute prediction error are shown for $\mathcal{F} = 0.05$ and $\mathcal{F} = 0.2$ in Figure 11(a) and Figure 11(b). For $\mathcal{F} = 0.05$, both models are able to predict the velocity fields with reasonable accuracy, although the PINN shows qualitatively better results than the DNN. In comparison to the cylinder case, the L_2 error of the DNN is much lower for $\mathcal{F} = 0.05$, which can be attributed to a larger number of training data points for the Rankine oval compared to the cylinder. For $\mathcal{F} = 0.2$, the DNN struggles to predict the x-velocity components near the stagnation point and downstream of the oval. Once again, PINNs show a higher prediction accuracy for potential flow with lower $\mathcal{F}$ values and can be used to predict the flow around a Rankine oval when minimal ground truth data is available.

3.3. Blasius boundary layer flow

For the Blasius boundary layer flow case, hyperparameter tuning yields best results when using the SGD optimizer with a learning rate of $LR = 0.002$, 6 hidden layers and 60 neurons per hidden layer. The loss to be minimized is calculated using Eqs. (35) and (36). As ground truth, $N_{total} = 10{,}000$ data points are extracted from the numerical solution and are randomly distributed for data-driven training, keeping the boundary points fixed for each training run. Both models are trained on a single GPU for 20,000 epochs and the epochs are kept constant for each training run.

The L_2 testing error progressions against compute time required by both PINNs and DNNs are shown in Figure 12. For both $\mathcal{F} = 0.2$ and $\mathcal{F} = 0.4$, the training times of PINNs are almost ten times higher than DNNs. Figure 13 shows, that

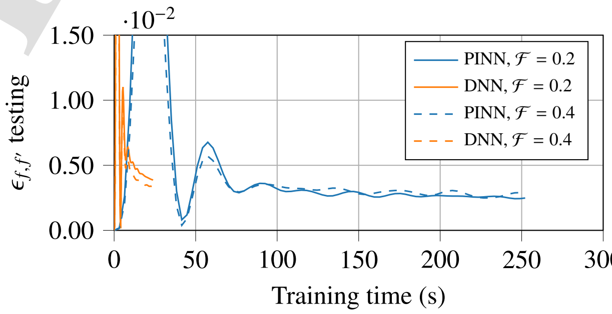


Figure 12: L_2 testing error vs training time for the Blasius boundary layer flow.

except for $\mathcal{F} = 0.4$, the PINN-based predictions have a lower L_2 error compared to the DNN-based predictions. At $\mathcal{F} = 0.4$, both types of ANN have a similar accuracy with an L_2 error of 7.0×10^{-4}.

Figures 14(a) and 14(c) show the predicted velocity profiles obtained from the models with $\mathcal{F} = 0.2$, and Figures 14(b) and 14(d) for $\mathcal{F} = 0.4$. When trained with $\mathcal{F} = 0.2$, both models predict the velocity profile of the streamwise component (u/U_0) well with minimal deviation from the ground truth between $\eta = 5.0$ and $\eta = 8.0$. Predictions of both models for the normal velocity component $(v\sqrt{x/(\nu U_0)})$ are in good agreement with the ground truth away from the wall. However, near the wall, the PINN has a better prediction than the DNN, which

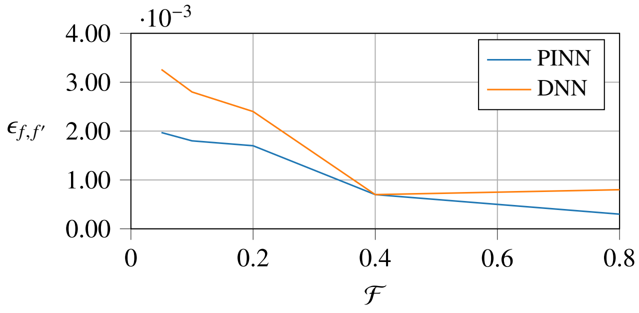


Figure 13: L_2 prediction error for a varying $\mathcal{F}$ value for the Blasius boundary layer flow case.

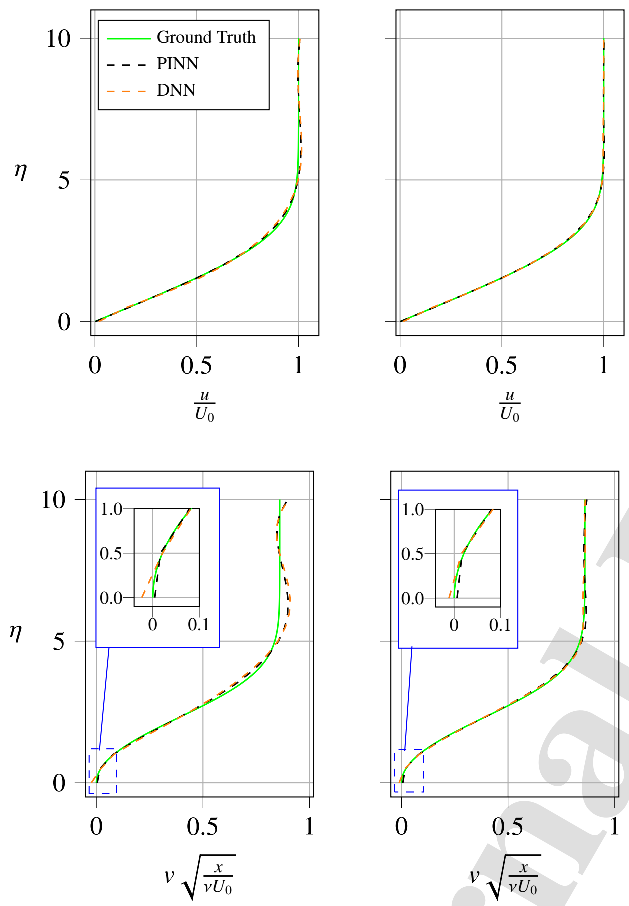


Figure 14: Comparison of predictions of velocity profiles in the Blasius boundary layer by a PINN and a DNN against the exact solution with $\mathcal{F} = 0.2$ (a, c) and $\mathcal{F} = 0.4$ (b, d).

can be observed in the zoomed inset in Figure 14(c). In the region between $\eta = 5.0$ and $\eta = 10.0$, which correspond to the free stream conditions, predictions from both models show deviation from the ground truth. While considering the predictions with $\mathcal{F} = 0.4$, both models predict the velocity profiles in good agreement with the ground truth both in the boundary layer and free stream regions. In this case, the PINN provides again a better prediction of the normal velocity component near the wall, as can be seen in Figure 14(d). It can be concluded that including a sufficient amount of ground truth data in the training can help in accurately predicting the velocity profiles for the boundary layer flow problem simplified by Blasisus. However, it has to be noted that this is achieved with a higher computation cost for PINN, and hence this gain in accuracy has to be justified for higher computational efforts.

3.4. 2D Taylor-Green Vortex

The spatial grid for the two-dimensional Taylor-Green vortex is generated using the `meshgrid` function in `NumPy`. The grid spacing is uniform with a cell size of 0.02, and $(x, y) \in [-\pi, \pi]$. Data for training is extracted from the complete spatio-temporal grid for six time snapshots with a temporal step size of 5 sec, where each time snapshot has the same spatial grid. This time step size is selected such that the velocity and pressure fields have varied enough to train the ANNs on the temporal range. For each time step, a total of $N_{total} = 99{,}860$ spatial points are generated of which 1,264 points are located at the domain boundary. Again, the number and location of the boundary points are kept constant for the training of all models.

The percentage of the domain data points with an exact solution is varied during the training of PINNs and DNNs. When using the S2S method, the number of training data points in the spatial domain for each time step is kept constant and the domain data points are randomly chosen. The PINN and DNN models are trained for a time range of $[0, \ldots, 30]$ s. The SGD optimizer with a learning rate of $LR = 0.003$ is used for training the PINNs and DNNs, and each hidden layer has 300 neurons. The stopping criteria for training of each time step is set to 30,000 epochs. Models on a coarse grid with a cell size of 0.05 and $N_{total} = 16{,}000$ points are also trained for each time step. The training for each time step is run for 20,000 epochs. The objective is to investigate the model performance under different grid sizes. These are referred as reduced models in this text. All models are trained on 10 nodes, using in total 40 GPUs.

To compare the training time of PINNs and DNNs, the L_2 testing error progressions are plotted in Figure 15 for $\mathcal{F} = 0.05$. Each peak signifies the start of sequence training for the next time step. As observed, the DNN achieves a relatively lower training error at the end of the second sequence, but the error does not decrease further in following training sequences. Although the PINN has a higher training error for the second sequence, the error decreases consistently in the following sequences.

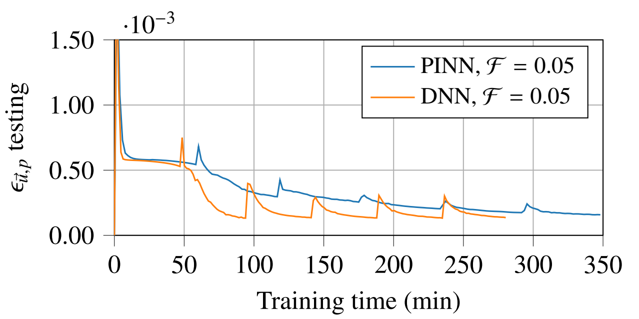


Figure 15: L_2 testing error vs. training time for the 2D Taylor-Green vortex with a grid cell size of 0.02.

The advantage of S2S training for PINNs is reflected in the prediction accuracy of temporal interpolation. The L_2 error for different $\mathcal{F}$ values are plotted for $t = 17s$ in Figure 16 for both fine and coarse grids. Note that flow fields from $t = 17s$ did not belong to the training data. There is no intersection point

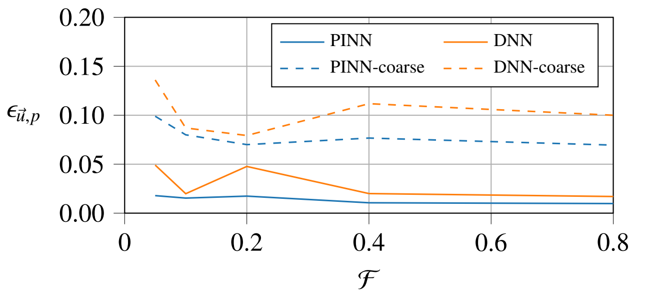


Figure 16: L_2 prediction error for a varying $\mathcal{F}$ for the 2D-Taylor-Green vortex at $t = 17$.

found for the training setups. The PINN models consistently show better performance than the DNN models for all variations in $\mathcal{F}$. For the reduced models, the lowest prediction error of 6.9×10^{-2} is achieved by the PINN using $\mathcal{F} = 0.8$. In case of the finer mesh, the PINN achieves the lowest L_2 error of 9.8×10^{-3} at $\mathcal{F} = 0.8$. For increasing $\mathcal{F}$ values, PINNs have a consistently improving performance, whereas DNN-based predictions are characterized by a fluctuating L_2 error, similar to the potential flow cases. That is, the inclusion of governing physics and increased ground truth data in training can improve ANN predictions for a two-dimensional Taylor-Green vortex trained using the S2S method.

Variable	$PINN$	DNN	$PINN_{reduced}$	$DNN_{reduced}$
u	0.0175	0.0084	0.054	0.0539
v	0.0078	0.0104	0.057	0.0576
p	0.0267	0.1239	0.0944	0.1264

Table 2: L_2 error in the output variables of the two-dimensional Taylor-Green Vortex for $t = 17s$ using $\mathcal{F} = 0.05$. Reduced models are trained on a dataset with $N_{total} = 16{,}000$ spatial grid points.

The L_2 errors for the different models are summarized for $\mathcal{F} = 0.05$ and $t = 17s$ in Table 2. It can be observed that models trained on a coarse grid have higher L_2 errors.

A qualitative comparison of predicted variables with the exact solution at $t = 17$ sec is shown in Figure 17 for the models trained on a finer grid. The large blank regions in the pressure field of the DNN predictions highlight the model's inability to predict fields with different min-max ranges when no physical loss is used in the training. The velocity fields are predicted well by both models. A similar comparison is shown for the reduced models in Figure 18 and a similar trend for the predictions of the pressure field is observed. Given the unsteady nature of this problem, all models are trained in time with S2S learning, see Section 2. The results shown in Figures 17 and 18 highlight the interpolation capability of the S2S-trained models. Despite having no data from $t = 17s$ in the training, the models are still able to predict the flow variables at this point in time.

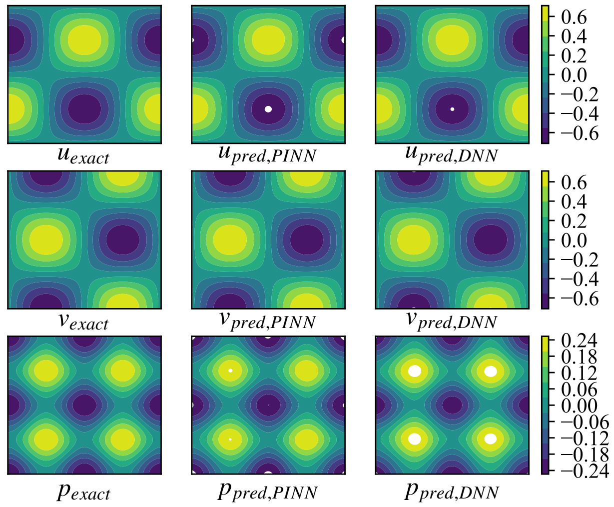


Figure 17: Comparison of the prediction performance of PINN and DNN models for the two-dimensional Taylor-Green vortex at $t = 17s$ and $\mathcal{F} = 0.05$. The blank regions are predictions outside the range of the ground truth.

3.5. *Effect of spatial distribution on prediction*

In the investigations above, the data points for each $\mathcal{F}$ value are randomly distributed in the flow domain and the training data at the boundaries and walls are kept fixed. In this section, the variation in performance of data driven ANNs with a change in spatial distribution of the data points for a given $\mathcal{F}$ value is analyzed. That is, a Region of Interest (ROI) is specified and the data points are distributed in this ROI. This space-specific distribution of data in the ROI is termed as prescribed distribution in this work. An example is illustrated in Figure 19, where the ROI is the near-wall region of an arbitrarily shaped body. Additionally, data-free training is investigated, where only data points at boundaries are used as ground truth data. In such a case, the red dots in Figure 19 disappear. The ANN models for each case are trained with the same hyperparameters as defined in the above discussed results.

For potential flow problems, the ROI is the near-wall region and the data points for the L_I loss at $\mathcal{F} = 0.05$ are distributed near the wall of the cylinder and the boundary of the Rankine oval. The L_{II} loss for the PINN is calculated using randomly distributed points as described in Section 2.2.

As shown in Figure 20(a), for the cylinder case, the prediction accuracy of the PINN is with an L_2 error of 2.64×10^{-3} far better than the DNN with an L_2 error of 6.1×10^{-2}. However, it can also be seen for the DNN that the flow field near the wall of the cylinder and domain boundaries is predicted with comparatively lower error than the rest of the flow field. This explains the dependence of data-driven ANNs on the spatial distribution of the training data. A similar performance is shown in Figure 20(b) for the data-free PINNs with only boundary conditions as constraints. For the data-free models, the L_2 error with the PINN is 7.7×10^{-4}, whereas the DNN prediction has an L_2 error of 6.45×10^{-2}.

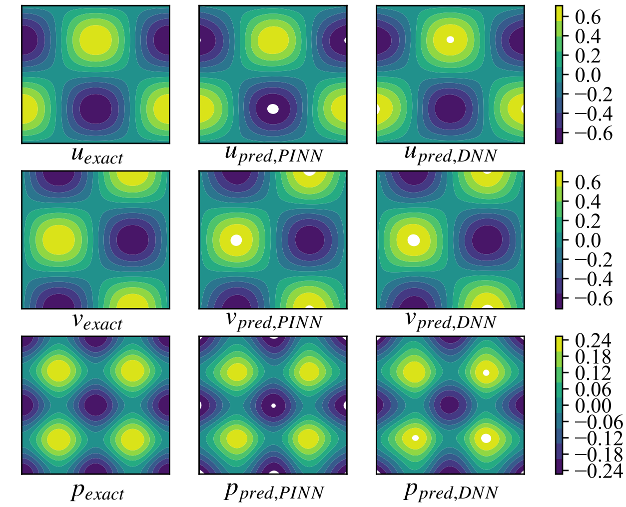


Figure 18: Comparison of the prediction performance of the reduced PINN and DNN models for the two-dimensional Taylor-Green vortex at $t = 17s$ and $\mathcal{F} = 0.05$. The blank regions are predictions outside the range of the ground truth.

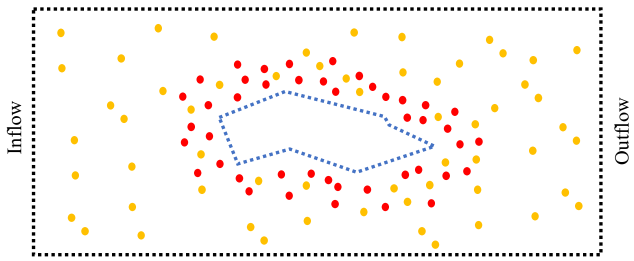


Figure 19: Distribution of data points for a general example of a two-dimensional flow around an arbitrary shape with a region of interest (ROI) near the wall. The boundary points N_b and N_w are shown with black and blue dots, and the domain points N_d are shown with yellow dots. All these points are kept fixed for each training run. The variable data points with existing ground truth data, $N_{d,1} \subseteq N_d$ are denoted by the red dots.

Similar results are obtained for the potential flow around the Rankine oval. Both ANN models have a reduced prediction accuracy when trained on ground truth data concentrated near the boundary of the Rankine oval. As shown in Figure 21(a), even the PINN struggles to predict the flow field near the domain boundaries when training data from ground truth is concentrated near the boundaries. However, the flow field near the wall, which has higher velocity gradients and is critical to the flow development, is still well reconstructed. The L_2 error for the PINN is 4.26×10^{-3} in comparison to 5.6×10^{-1} for the DNN. In case of the DNN, in areas in the vicinity of the wall (10 cell lengths), the prediction seems to be marginally better. While considering the data-free case in Figure 21(b), both the PINN and DNN show improvements in prediction accuracy. In this case, the PINN has an L_2 error of 2.4×10^{-3}, while the error in case of the DNN is 4.98×10^{-2}.

For the Blasius flow case, the input to the ANN is defined by the variable η. The ROIs are not randomly selected, but they are

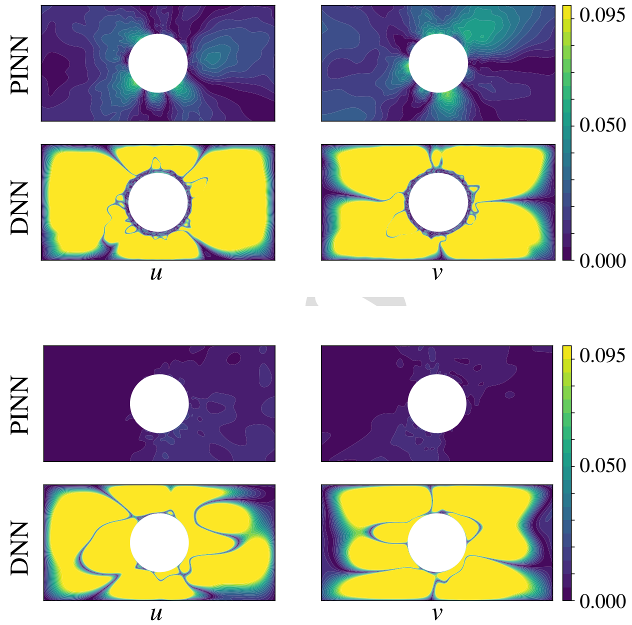


Figure 20: Error density for the potential flow around a cylinder when trained with a concentrated spatial distribution of data points. (a) $\mathcal{F} = 0.05$ with a near-wall ROI and, (b) data-free prediction.

defined based on the boundary conditions given by Eqs. (18)-(20). It can be observed from the prediction results shown in Figure 22 that the model accuracy is highly dependent on the distribution of data. Both the PINN and DNN have a decreased prediction accuracy when trained on data at $\eta = 0$ and $5 \geq \eta \leq 10$ having the same amount of ground truth data as at $\mathcal{F} = 0.2$. The PINN has an L_2 error of 2.1×10^{-2} and the DNN has an L_2 error of 4.1×10^{-1}, which are one order of magnitude higher than the L_2 error from training with randomly distributed ground truth data.

A similar analysis is conducted for the two-dimensional Taylor-Green vortex, such that the training data from the ground truth is concentrated near the domain boundaries and corresponds to $\mathcal{F} = 0.05$. As observed in Figure 23, both the PINN and DNN fail to predict the velocity and pressure fields. Although both models have reduced accuracy as compared to models trained on randomly distributed data, velocity predictions from the PINN are able to capture vortex structures, while the DNN completely fails to reconstruct the velocity field. The L_2 error for both velocity components predicted by the PINN is 7.1×10^{-10} and for predictions by the DNN 1.52. Both models achieve a comparable accuracy in the prediction of the pressure field with L_2 errors of 1.54 and 1.22 for PINN and DNN respectively.

The effect of the distribution of training data can be observed in Figure 23, where the models are able to reconstruct the fields near domain boundaries with more accuracy as compared to rest of the domain. When the distribution of ground truth data is concentrated around the regions of high pressure gradients with $\mathcal{F} = 0.05$, the prediction accuracy of both PINNs and DNNs

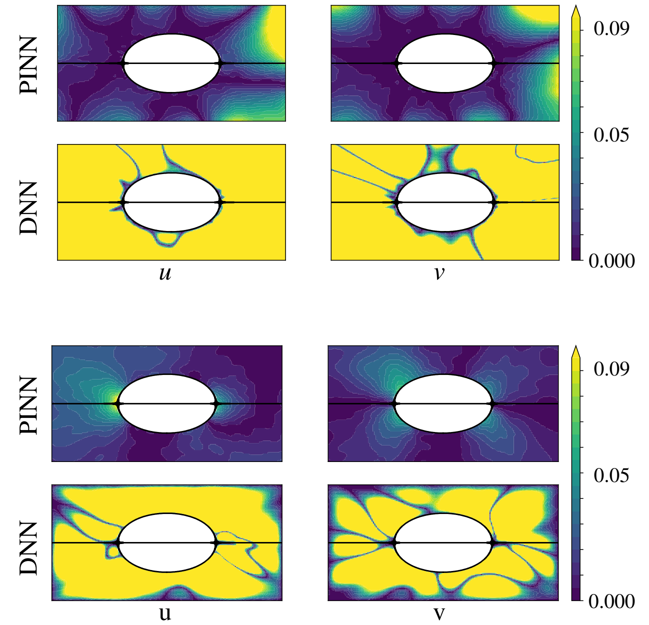


Figure 21: Error density for the potential flow around the Rankine oval when trained with a concentrated spatial distribution of data points. (a) Prediction corresponding to $\mathcal{F} = 0.05$ with a near-wall ROI, and (b) data-free prediction.

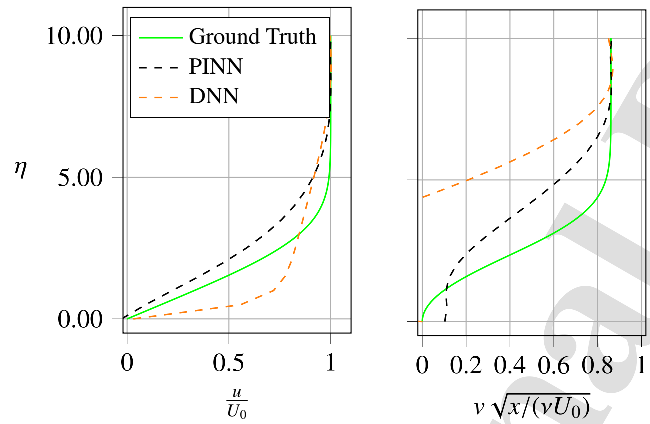


Figure 22: Comparison of the predicted velocity profiles in the Blasius boundary layer by PINN, DNN, and the exact solution. The ground truth is defined by only the boundary conditions and the number of data points corresponding to $\mathcal{F} = 0.2$.

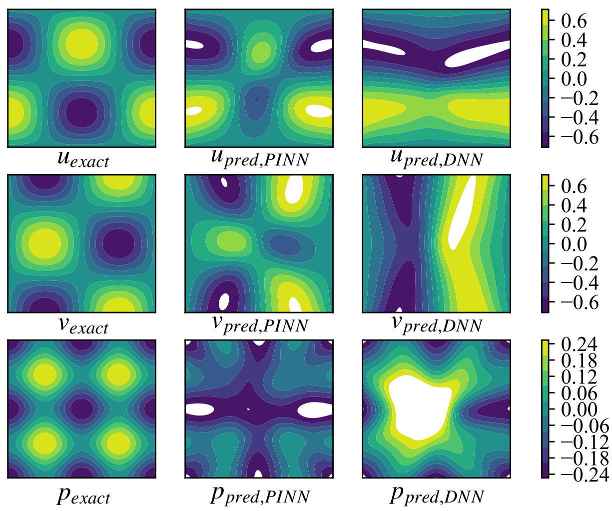


Figure 23: Comparison of the prediction performance of PINN and DNN models for the two-dimensional Taylor-Green vortex at $t = 17s$ and $\mathcal{F} = 0.05$. Training data from ground truth is prescribed near the boundaries and the blank regions are predictions outside the range of the ground truth.

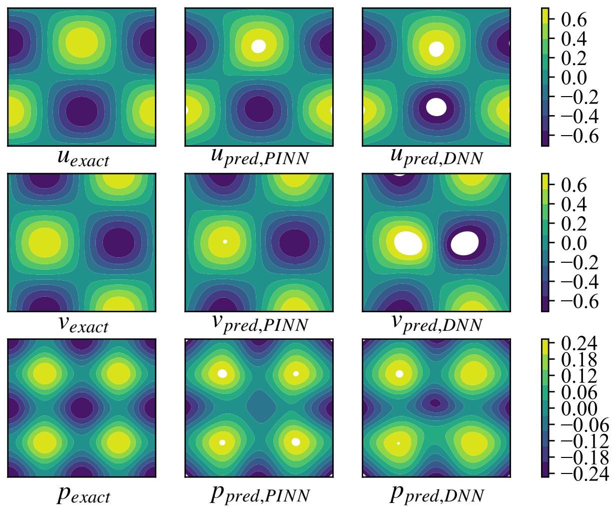


Figure 24: Comparison of the prediction performance of PINN and DNN models for the two-dimensional Taylor-Green vortex at $t = 17s$ and $\mathcal{F} = 0.05$. Training data from ground truth is prescribed near the high pressure regions and the blank regions are predictions outside the range of the ground truth.

improves as shown in Figure 24. The vortical structures are captured and also the DNN is able to reconstruct the pressure field with an L_2 error of 3.5×10^{-1}. The prediction of the x-component of the velocity field improves the most with an L_2 error of 1.1×10^{-1} and 1.3×10^{-1} from the PINN and DNN respectively. The above results highlight the importance of integrating the governing physics in the loss function of ANNs and the effect of distribution of training data from the ground truth on the predictive performance of the two-dimensional Taylor-Green vortex. The PINNs show better performance than DNNs for all data distributions. Both models perform best when training data is randomly distributed.

A summary of the prediction results for the above discussed flow problems is shown in Table 3. For each flow problem investigated in this work, the PINNs outperform the DNNs. The largest difference in performance of both models is observed for potential flow, while both models have comparable performance for the two-dimensional Taylor-Green vortex. For the potential flow and Blasius case, it is also observed that the

Flow case	$\mathcal{F}$	Ground Truth distribution	PINN L_2 Error	DNN L_2 Error	DNN-to-PINN L_2 error ratio
Potential flow: cylinder	0.05	Random	8.2×10^{-4}	1.4×10^{-3}	1.7
	0.05	Prescribed	2.64×10^{-3}	6.1×10^{-2}	23.10
	0.0	Boundary conditions	7.7×10^{-4}	6.45×10^{-2}	83.70
Potential flow: Rankine oval	0.05	Random	3.0×10^{-4}	1.2×10^{-3}	4.00
	0.05	Prescribed	4.26×10^{-3}	5.6×10^{-1}	131.40
	0.0	Boundary conditions	2.4×10^{-3}	4.98×10^{-2}	20.75
Blasius boundary layer flow	0.2	Random	1.7×10^{-3}	2.4×10^{-3}	1.41
	0.2	Boundary conditions	2.1×10^{-2}	4.1×10^{-1}	19.50
2D Taylor-Green vortex	0.05	Random	1.7×10^{-2}	4.9×10^{-2}	2.88
	0.05	Random-reduced	9.9×10^{-2}	1.34×10^{-1}	1.35
	0.05	Prescribed-BC	7.193×10^{-1}	1.52	2.11
	0.05	Prescribed-PC	1.48×10^{-1}	2.1×10^{-1}	1.41

Table 3: Prediction accuracy of flow problems for different ANN configurations, amount of ground truth data used in training, and distribution of training data on the grid. BC: Boundary condition, PC: Centers of high pressure.

DNN-to-PINN L_2 error ratio is significantly higher when the ground truth data is prescribed in the ROI defined by boundaries or high gradients or the data-free case, compared to the randomly distributed case.

3.6. *Effect of noise in training data*

After evaluating the performance of DNNs and PINNs against variations in training data distribution, the effect of noise in training data is investigated. This noise scaling represents the Signal to Noise Ratio (SNR) metric commonly used for measuring devices used for experiments. As discussed in Section 1, deep learning based PINNs can be used to extrapolate flow information from sensors on vehicles under on-road conditions. To replicate noisy sensor data, training data is embedded with Gaussian noise. The noise is scaled to be between 10% and 20% of the standard deviation inherent in the velocity data across the domain. The impact of noise on flow structures of potential flow around a Rankine oval is shown in Figure 25.

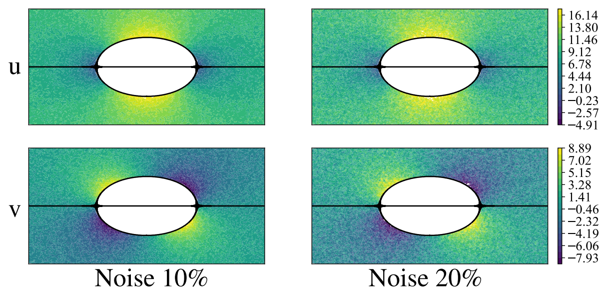


Figure 25: Noise in training data for potential flow around a Rankine oval.

Both PINN and DNN are trained on training data with varying SNR and $\mathcal{F} = 0.2$. The training hyperparameters are kept similar to the models used in Section 3.2 and both models are trained for 20,000 epochs. The errors in predicted flow fields for flow around a Rankine oval are shown in Figure 26. As observed, the inclusion of physical constraints helps the reconstruction in the presence of noise in the training data. The prediction error from PINN is 1.704×10^{-3} and 3.08×10^{-3} for 10% and 20% noise. In comparison, the prediction error for $\mathcal{F} = 0.2$ from a PINN trained without noise in data is 1.57×10^{-3}. Thus, the prediction error of PINN increases by 8% and 96% for training data with 10% and 20% noise respectively. On the other hand, the performance of DNN degrades heavily with noisy training data. When compared with the DNN trained on data without noise, the prediction error increases by 100% and 600% for 10% and 20% noise.

4. Conclusion and Outlook

In this work, the performance of data-driven ANNs is investigated for four classical flow problems. The ANNs are based on two network configurations: a classical DNN architecture and a PINN, the latter enforcing physical constraints in the loss function. The amount and location of ground truth data employed in training are varied for both architectures, and the effect on the prediction accuracy is compared.

For the potential flow configurations of a cylinder and Rankine oval, the results show lower errors using PINNs when less ground truth data is available for training. For the cylinder case, PINNs performed better for all $\mathcal{F}$ values. Different results for ANNs are obtained for potential flow around a Rankine oval, where DNNs perform better for $\mathcal{F} > 0.38$. Additionally, an analysis on the location of the ground truth data used in the training was performed. In contrast to the data-driven training using randomly distributed ground truth data, training with prescribed sampling of data points for potential flow cases have comparatively higher L_2 errors. Thus, the distribution of ground truth data for data-driven cases is an important factor for improving prediction accuracy. The data-free training has better prediction accuracy than the data-driven training with prescribed sampling of data points. The results are, however, still worse than the case with the random distribution of training data. However it was observed that the PINNs significantly outperformed DNNs, when the training data was prescribed. This is especially important for real-world applications, for instance when limited sensor measurements are available based on location constraints. In this case, the PINN would be an obvious choice over DNN.

Summarizing the observations from the Blasius boundary layer flow, PINNs have a better prediction accuracy for all $\mathcal{F}$ values except at $\mathcal{F} = 0.4$, where both the PINN and DNN have similar accuracy. Data-driven models with ground truth data

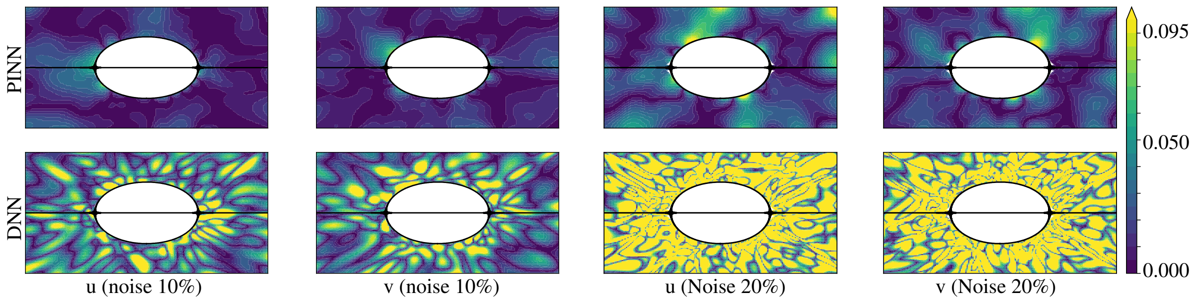


Figure 26: Error density plots for flow around a Rankine oval when trained on noisy data with $\mathcal{F} = 0.2$.

concentrated near the boundaries have a higher L_2 error in velocity profiles compared to the case, when ground truth data is randomly distributed. Even for the prescribed data distribution, the PINN achieves an L_2 error one order lower than that of the DNN. Given the availability of ground truth data corresponding to $\mathcal{F} \geq 0.4$ and a random distribution of ground truth data, velocity fields can be predicted with higher accuracy using PINNs.

The unsteady flow problem of the two-dimensional Taylor-Green vortex is solved using the S2S method, where each time-step is individually trained and solutions from previous time-steps are used as additional constraints. Both PINN and DNN data-driven models when trained on randomly distributed ground truth, are able to capture flow structures and reconstruct velocity and pressure fields. For all values of $\mathcal{F}$ investigated in this work, PINNs have better prediction accuracy than DNNs. Additionally, model performance is compared for different cell size in grid and also for prescribed distribution of ground truth in training. It is observed that the PINN is able to outperform the DNN even when trained for larger cell sizes. However, performance of both models improved when the grid cell size is reduced from 0.05 to 0.02.

When trained with ground truth data distributed only near the domain boundaries, PINNs have a better prediction of the velocity field compared to DNNs. Both models have a comparable prediction accuracy for the pressure field. When compared with the results from the randomly distributed data-driven training, both models have poor predictions and fail to reconstruct the velocity and pressure fields. The prediction accuracy of both PINNs and DNNs improved when ground truth data is distributed around the regions of high pressure gradients, but is still lower than the randomly distributed data-driven training. Based on the above results, it can be concluded that S2S data-driven models implemented for the unsteady flow problem in this work have a strong dependence on spatial distribution of ground truth in training and the prediction accuracy can be improved by using a smaller cell size. Further improvement of the predictive capabilty of PINNs for unsteady flow problems may be possible with normalization of training data to a common range and application of weighing functions for L_{II} loss terms.

Furthermore, an analysis to compare the training costs for both the PINN and DNN was performed. As expected, it is found that PINNs have higher training cost compared to DNNs, even by a factor of ten in some cases. But it is observed that PINNs consistently perform better than DNNs, especially when the data is sparse and they are located in critical locations such as near the wall. Furthermore, under noisy training data, PINNs perform significantly better than DNNs, which had a loss in accuracy of 100% compared to 8% for PINN under 10% noise in training data. And in many practical problems of interest, data is generally sparse and also noisy. Hence, the compromise with the higher training costs provides an ANN with higher accuracy, which is robust to noise and data sparsity. This is observed to be a significant advantage offered by PINNs, albeit the higher computational costs.

To the knowledge of the authors, the investigation in this manuscript is one of the first attempts to quantify the amount and location of training data when comparing the performance of PINNs and DNNs, along with inclusion of the effect of noise. In this case, the investigations are limited to classical flow problems, where it is observed that this choice significantly affects the prediction accuracy. This finding could potentially be exploited to utilize the superior performance of PINNs in cases, where limited and concentrated sensor measurements are available for real-world applications. For a fixed geometry of a car body, a version of the PINN with constraints based on the Navier-Stokes equations can be trained on the sparse and noisy surface sensor data, to predict flow fields for different on-road conditions. S2S learning can be used to constantly feed new data to the model at successive time intervals, while preserving the information learned from the previous time intervals. The findings in this work serve as a benchmark for such physics-based machine learning methods to be extended to realistic flow cases in the future, to complement traditional solvers and reduce computation costs.

Conflict of Interest statement

This statement is to declare that the authors of this manuscript, Rishabh Puri, Junya Onishi, Mario Rüttgers, Rakesh Sarma, Makoto Tsubokura, and Andreas Lintermann do not possess any financial dependence that might bias this work. The authors hereby declare that no conflict of interest exists in this work.

Acknowledgement

The research leading to these results has been conducted in the Joint Laboratory for Extreme Scale Computing (JLESC) project: Deep Neural Networks for CFD Simulations. Furthermore, the research has been performed in the CoE RAISE project, which receives funding from the European Union's *Horizon 2020 – Research and Innovation Framework Programme* H2020-INFRAEDI-2019-1 under grant agreement no. 951733. The authors gratefully acknowledge the computing time granted by the JARA Vergabegremium and provided on the JARA Partition part of the supercomputer JURECA [49] at *Forschungszentrum Jülich*.

Author contributions

Rishabh Puri and Junya Onishi developed the PINNs and DNNs and analyzed their predictive capabilities. Mario Rüttgers and Rakesh Sarma had the conceptual idea of analyzing the choice of physical constraints for PINN- and DNN-based predictions of flow fields and supervised the project. Makoto Tsubokura and Andreas Lintermann provided advise from fluid mechanical and machine learning perspectives, and directed the project. Rishabh Puri wrote the manuscript with contributions by all co-authors.

References

[1] L. Euler, Principes généraux du mouvement des fluides, Mémoires de l'Academie des Sciences de Berlin 11 (1757) 274–315.

[2] G. G. Stokes, On the theories of the internal friction of fluids in motion and of the equilibrium and motion of elastic solids, Trans. Cambridge Philos. Soc. 8 (1845) 287–319.

[3] G. Calzolari, W. Liu, Deep learning to replace, improve, or aid cfd analysis in built environment applications: A review, Building and Environment 206 (2021) 108315. doi:https://doi.org/10.1016/j.buildenv.2021.108315.

[4] 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, Journal of Computational Physics 378 (2019) 686–707. doi:10.1016/j.jcp.2018.10.045.

[5] X. Guo, W. Li, F. Iorio, Convolutional neural networks for steady flow approximation, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 481–490. doi:10.1145/2939672.2939738.

[6] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nature Reviews Physics 3 (6) (2021) 422–440. doi:10.1038/s42254-021-00314-5.

[7] N. Thuerey, K. Weißenow, L. Prantl, X. Hu, Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows, AIAA Journal 58 (1) (2020) 25–36. arXiv:https://doi.org/10.2514/1.J058291, doi:10.2514/1.J058291.

[8] M. Matsuo, K. Fukami, T. Nakamura, M. Morimoto, K. Fukagata, Reconstructing three-dimensional bluff body wake from sectional flow fields with convolutional neural networks, SN Computer Science 5 (3) (2024) 306. doi:10.1007/s42979-024-02602-0.

[9] V. Sekar, Q. Jiang, C. Shu, B. C. Khoo, Fast flow field prediction over airfoils using deep learning approach, Physics of Fluids 31 (5) (2019) 057103. doi:10.1063/1.5094943.

[10] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: N. Navab, J. Hornegger, W. M. Wells, A. F. Frangi (Eds.), Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015, Springer International Publishing, Cham, 2015, pp. 234–241.

[11] J. Chen, J. Viquerat, E. Hachem, U-net architectures for fast prediction of incompressible laminar flows (2019). arXiv:1910.13532.

[12] M. Jolaade, V. L. S. Silva, C. E. Heaney, C. C. Pain, Generative networks applied to model fluid flows, in: D. Groen, C. de Mulatier, M. Paszynski, V. V. Krzhizhanovskaya, J. J. Dongarra, P. M. A. Sloot (Eds.), Computational Science – ICCS 2022, Springer International Publishing, Cham, 2022, pp. 742–755.

[13] S. R. Bukka, R. Gupta, A. R. Magee, R. K. Jaiman, Assessment of unsteady flow predictions using hybrid deep learning based reduced-order models, Physics of Fluids 33 (1) (2021) 013601. doi:10.1063/5.0030137.

[14] S. Fotiadis, E. Pignatelli, M. L. Valencia, C. D. Cantwell, A. J. Storkey, A. A. Bharath, Comparing recurrent and convolutional neural networks for predicting wave propagation, ArXiv abs/2002.08981 (2020).

[15] F. Sofos, D. Drikakis, I. W. Kokkinakis, Deep learning architecture for sparse and noisy turbulent flow data, Physics of Fluids 36 (3) (2024) 035155. doi:10.1063/5.0200167.

[16] J. Chen, E. Hachem, J. Viquerat, Graph neural networks for laminar flow prediction around random two-dimensional shapes, Physics of Fluids 33 (12) (2021) 123607. doi:10.1063/5.0064108.

[17] M. Lino, S. Fotiadis, A. A. Bharath, C. D. Cantwell, Multi-scale rotation-equivariant graph neural networks for unsteady Eulerian fluid dynamics, Physics of Fluids 34 (8) (2022) 087110. doi:10.1063/5.0097679.

[18] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: A survey, Journal of Machine Learning Research 18 (1) (2017) 1–43.

[19] S. Cai, Z. Mao, Z. Wang, M. Yin, G. E. Karniadakis, Physics-informed neural networks (pinns) for fluid mechanics: a review, Acta Mechanica Sinica 37 (12) (2021) 1727–1738. doi:10.1007/s10409-021-01148-1.

[20] K. Um, R. Brand, Y. R. Fei, P. Holl, N. Thuerey, Solver-in-the-loop: Learning from differentiable physics to interact with iterative pde-solvers, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20, Curran Associates Inc., Red Hook, NY, USA, 2020. doi:https://doi.org/10.48550/arXiv.2007.00016.

[21] S. L. Brunton, B. R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annual Review of Fluid Mechanics 52 (1) (2020) 477–508. doi:10.1146/annurev-fluid-010719-060214.

[22] Z. Long, Y. Lu, X. Ma, B. Dong, PDE-net: Learning PDEs from data, in: J. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, Vol. 80 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 3208–3216. doi:https://doi.org/10.48550/arXiv.1710.09668.

[23] J. Han, A. Jentzen, W. E, Solving high-dimensional partial differential equations using deep learning, Proceedings of the National Academy of Sciences 115 (34) (2018) 8505–8510. doi:10.1073/pnas.1718942115.

[24] A. Arzani, J.-X. Wang, R. D'Souza, Uncovering near-wall blood flow from sparse data with physics-informed neural networks, Physics of Fluids 33 (2021) 071905. doi:10.1063/5.0055600.

[25] Zhiping Mao, Ameya D. Jagtap, George Em Karniadakis, Physics-informed neural networks for high-speed flows, Computer Methods in Applied Mechanics and Engineering 360 (2020) 112789. doi:10.1016/j.cma.2019.112789.

[26] M. Raissi, A. Yazdani, G. E. Karniadakis, Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations, Science 367 (6481) (2020) 1026–1030. doi:10.1126/science.aaw4741.

[27] E. Illarramendi, A. Alguacil, M. Bauerheim, A. Misdariis, C. Benedicte, E. Benazera, Towards an hybrid computational strategy based on deep learning for incompressible flows (06 2020). doi:10.2514/6.2020-3058.

[28] N. Thuerey, T. Pfaff, mantaflow: An extensible framework for fluid simulation. http://mantaflow.com/index.html, (2016), (accessed 01 July 2023).

[29] H. Ma, Y. Zhang, N. Thuerey, X. Hu, O. J. Haidn, Physics-driven learning of the steady navier-stokes equations using deep convolutional neural networks, Communications in Computational Physics 32 (3) (2022) 715–736. doi:https://doi.org/10.4208/cicp.OA-2021-0146.

[30] J.-Z. Peng, N. Aubry, Y.-B. Li, M. Mei, Z.-H. Chen, W.-T. Wu, Physics-informed graph convolutional neural network for modeling geometry-adaptive steady-state natural convection, International Journal of Heat and Mass Transfer 216 (2023) 124593. doi:https://doi.org/10.1016/j.ijheatmasstransfer.2023.124593.

[31] W. Wu, X. Feng, H. Xu, Improved deep neural networks with domain decomposition in solving partial differential equations, Journal of Scientific Computing 93 (1) (2022) 20. doi:10.1007/s10915-022-01980-y.

[32] A. Malek, A. Emami Kerdabadi, Solving differential equations by artificial neural networks and domain decomposition, Iranian Journal of Science 47 (4) (2023) 1233–1244. doi:10.1007/s40995-023-01481-z.

[33] A. D. Jagtap, E. Kharazmi, G. E. Karniadakis, Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems, Computer Methods in Applied Mechanics and Engineering 365 (2020) 113028. doi:https://doi.org/10.1016/j.cma.2020.113028.

[34] E. H. W. Ang, G. Wang, B. F. Ng, Physics-informed neural networks for low reynolds number flows over cylinder, Energies 16 (12) (2023). doi:10.3390/en16124558.

[35] V. Grimm, A. Heinlein, A. Klawonn, Learning the solution operator of two-dimensional incompressible navier-stokes equations using physics-aware convolutional neural networks (2023). arXiv:2308.02137.

[36] P.-Y. Chuang, L. A. Barba, Experience report of physics-informed neural networks in fluid simulations: pitfalls and frustration (2022). doi:10.48550/ARXIV.2205.14249.

[37] R. Palin, V. Johnston, S. Johnson, A. D'Hooge, B. Duncan, J. I. Gargoloff, The aerodynamic development of the tesla model s - part 1: Overview, in: SAE 2012 World Congress & Exhibition, SAE International, 2012. doi:https://doi.org/10.4271/2012-01-0177.

[38] A. D'Hooge, R. B. Palin, S. Johnson, B. Duncan, J. I. Gargoloff, The aerodynamic development of the tesla model s - part 2: Wheel design optimization, in: SAE 2012 World Congress & Exhibition, SAE International, 2012. doi:https://doi.org/10.4271/2012-01-0178.

[39] M. Rüttgers, J. Park, D. You, Large-eddy simulation of turbulent flow over the drivaer fastback vehicle model, Journal of Wind Engineering and Industrial Aerodynamics 186 (2019) 123–138. doi:https://doi.org/10.1016/j.jweia.2019.01.003.

[40] A. A. Lawson, R. G. Dominy, D. B. Sims-Williams, P. Mears, A comparison between on-road and wind tunnel surface pressure measurements on a mid-sized hatchback, in: SAE World Congress & Exhibition, SAE International, 2007. doi:https://doi.org/10.4271/2007-01-0898.

[41] D. Zhang, S. Subramanian, R. Hampson, W. Jackson, K. Kontis, G. Dobie, C. Macleod, Automotive aerodynamics sensing using low-profile pressure sensor strip, IEEE Transactions on Instrumentation and Measurement 72 (2023) 1–9. doi:10.1109/TIM.2023.3292963.

[42] L. Prandtl, Über flüssigkeitsbewegung bei sehr kleiner reibung, in: A. Krazer (Ed.), Chronik des III. Internationalen Mathematiker- Kongresses in Heidelberg, ACM Press, Heidelberg, 1904, pp. 484–491.

[43] P. R. H. Blasius, Grenzschichten in flussigkeiten mit kleiner reibung, Zeitschrift für angewandte Mathematik und Physik 56 (1908) 1–37.

[44] G. I. S. Taylor, A. E. Green, Mechanism of the production of small eddies from large ones, Proceedings of The Royal Society A: Mathematical, Physical and Engineering Sciences 158 (1937) 499–521.

[45] B. L. Kalman, S. C. Kwasny, Why tanh: choosing a sigmoidal function 4 (1992) 578–581 vol.4. doi:10.1109/IJCNN.1992.227257.

[46] R. Mattey, S. Ghosh, A novel sequential method to train physics informed neural networks for allen cahn and cahn hilliard equations, Computer Methods in Applied Mechanics and Engineering 390 (2022) 114474. doi:https://doi.org/10.1016/j.cma.2021.114474.

[47] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings, 2015. doi:10.48550/ARXIV.1412.6980.

[48] L. Bottou, F. E. Curtis, J. Nocedal, Optimization methods for large-scale machine learning, SIAM Review 60 (2) (2018) 223–311. doi:10.1137/16M1080173.

[49] D. Krause, P. Thörnig, Jureca: Modular supercomputer at jülich supercomputing centre, Journal of large-scale research facilities 4 (2018). doi:10.17815/jlsrf-4-121-1.

Highlights

- Integration of governing physics significantly improved the prediction accuracy of the data-driven and data-free Artificial Neural Network for the potential flow cases investigated in this work
- Prediction accuracy of data-driven Artificial Neural Networks depends on the distribution of the ground truth in training and random distribution of training data has best performance amongst the different distributions studied in this work
- For an unsteady two-dimensional Taylor-Green vortex, which was trained using the Sequence-to-Sequence method, data-driven Artificial Neural Networks were able to interpolate in the temporal range and reconstruct the vortex structures for untrained time steps

Author Biography

Rishabh Puri

Rishabh Puri is currently pursuing his doctoral degree at the Engler-Bunte Institute of the Karlsruhe Institute for Technology and he is working on numerical simulations of non-premixed ammonia combustion in porous media. Previously, he worked at the Simulation and Data Laboratory "Highly Scalable Fluid & Solids Engineering" (SDL FSE) of the Jülich Supercomputing Centre (JSC) in the domain of ML/AI in Fluid Dynamics. His research focus includes Direct Numerical Simulations (DNS) of turbulent reactive flows and development of Machine Learning (ML) methods for Computational Fluid Dynamics (CFD) and combustion.

Junya Onishi

Junya Onishi received the Ph.D. degree in Engineering from the University of Tokyo, Japan in 2004. He is currently a Research Scientist at the RIKEN Center for Computational Science, Japan. His research interests include the development and application of computational methods for solving fluid mechanical problems in industrial fields.

Mario Rüttgers

Mario Rüttgers is working as a postdoctoral researcher at the Simulation and Data Laboratory "Highly Scalable Fluid & Solids Engineering" (SDL FSE) of the Jülich Supercomputing Centre (JSC), Forschungszentrum Jülich. He received his doctoral degree at the Institute of Aerodynamics (AIA), RWTH Aachen University, in 2023. He is associated with the Helmholtz School for Data Science in Life, Earth and Energy (HDS-LEE). His research focuses on combining computational fluid dynamics and machine learning techniques to improve diagnoses and treatments of respiratory diseases.

Rakesh Sarma

Rakesh Sarma obtained his PhD from Delft University of Technology, Netherlands, in 2018. His doctoral thesis was on the development of Bayesian inference and reduced order modelling methods for prediction of instabilities in aeroelastic structures. Thereafter, he worked at the Dutch National Center for Mathematics and Computer Science in Amsterdam in the domain of ML/AI in space weather and stratified turbulence applications. Currently, he works at Forschungszentrum Jülich on development of parallel and scalable AI methods and workflows for HPC applications.

Makoto Tsubokura

Makoto Tsubokura is a professor in the department of computational science at the Kobe University, and also a team leader of complex phenomena unified simulation research team at RIKEN Center of Computational Science. He got his PhD in engineering from the university of Tokyo in 1998. His research interests include HPC-CFD for industrial applications, applied aerodynamics, and unified continuum mechanics simulation. He is a fellow of the Japan Society of Fluid Mechanics.

Andreas Lintermann

Andreas Lintermann is a postdoctoral researcher and group leader of the SDL FSE at JSC, Forschungszentrum Jülich. He is coordinating the European Center of Excellence in Exascale Computing "Research on AI- and Simulation-Based Engineering at Exascale" (CoE RAISE), leads the activities in the EuroCC/EuroCC2 and interTwin projects from Jülich's side, and is involved in the Industry Relations Team of the institute. His research focuses on high-performance computing, heterogeneous hardware, modular supercomputing, artificial intelligence, bio-fluidmechanical analyses of respiratory diseases, lattice-Boltzmann methods, high-scaling meshing methods, and efficient multi-physics coupling strategies.

Author credit statement

Rishabh Puri: Writing – Original Draft, Methodology, Validation, Software, Visualization **Junya Onishi:** Methodology, Validation, Investigation **Mario Rüttgers:** Conceptualization, Supervision, Writing – Review and Editing **Rakesh Sarma:** Conceptualization, Supervision, Writing – Review and Editing **Makoto Tsubokura:** Project administration **Andreas Lintermann:** Project administration, Resources, Writing – Review and Editing

Rishabh Puri

Junya Onishi

Mario Rüttgers

Rakesh Sarma

Makoto Tsubokura

Declaration of interests

☒ 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.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: