



Application of machine learning to model the pressure poisson equation for fluid flow on generic geometries

Paulo Sousa¹ · Alexandre Afonso^{2,3} · Carlos Veiga Rodrigues¹

Received: 9 October 2023 / Accepted: 28 April 2024 / Published online: 30 May 2024
© The Author(s) 2024

Abstract

This study addresses the importance of enhancing traditional fluid-flow solvers by introducing a Machine Learning procedure to model pressure fields computed by standard fluid-flow solvers. The conventional approach involves enforcing pressure–velocity coupling through a Poisson equation, combining the Navier–Stokes and continuity equations. The solution to this Poisson equation constitutes a substantial percentage of the overall computational cost in fluid flow simulations, therefore improving its efficiency can yield significant gains in computational speed. The study aims to create a versatile method applicable to any geometry, ultimately providing a more efficient alternative to the conventional pressure solver. Machine Learning models were trained with flow fields generated by a Computational Fluid Dynamics solver applied to the confined flow over multiple geometries, namely wall-bounded cylinders with circular, rectangular, triangular, and plate cross-sections. To achieve applicability to any geometry, a method was developed to estimate pressure fields in fixed-shape blocks sampled from the flow domain and subsequently assemble them to reconstruct the entire physical domain. The model relies on multilayer perceptron neural networks combined with Principal Component Analysis transformations. The developed Machine Learning models achieved acceptable accuracy with errors of around 3%. Furthermore, the model demonstrated enhanced computational efficiency, outperforming the classical PISO algorithm by up to 30 times.

Keywords Machine learning · Deep learning · Computational fluid dynamics · Incompressible flows · Poisson Eq

1 Introduction

Artificial Intelligence (AI) algorithms have been expanding toward scientific computation [1], predicting protein structures in biology [2], optimizing drug discovery [3, 4], accelerating material discovery [5], aiding climate modeling [6–8] and enhancing Computational Fluid Dynamics (CFD) solvers [9–11]. Broadly, there are three groups of models: (i) purely data-driven models working blindly to physical laws and requiring large amounts of data; (ii) physics-informed models that respect governing equations and boundary conditions [12]; and (iii) hybrid approaches of those two groups. Commonly the application of AI to physical problems aims to create an ambitious model that outputs a solution to the problem. Alternatively, the aim may focus on a surrogate model replacing a component of the physical solver to accelerate computations, e.g. turbulence closures in CFD models [13], or enhance the solution [10].

✉ Alexandre Afonso
aafonso@fe.up.pt

Paulo Sousa
pauloacunhasousa@hotmail.com

Carlos Veiga Rodrigues
calvr@vestas.com

¹ Modelling & Analytics, Vestas Wind Systems A/S,
Technology Centre Porto, Rua Lionesa,
4465-671 Matosinhos, Porto, Portugal

² Transport Phenomena Research Center, Faculdade de
Engenharia, Universidade do Porto, Rua Dr. Roberto Frias,
4200-465 Porto, Porto, Portugal

³ ALiCE, Associate Laboratory in Chemical Engineering,
Faculdade de Engenharia, Universidade do Porto, Rua Dr.
Roberto Frias s/n, 4200-465 Porto, Portugal

Fluid flow is a highly complex phenomenon for which analytical solutions are frequently unattainable in many engineering applications. Instead, numerical simulation alongside experimental studies plays an essential role in modeling. CFD is the process of employing numerical methods to solve mathematical models that describe fluid flow. Generally, these are based on the discretization of the Navier–Stokes (NS) equations over a computational grid that characterizes the volume occupied by the fluid. The computation of a pressure field that is consistent with the flow velocity is governed by a Poisson equation whose solution comprehends a significant percentage of the computational cost.

Deep learning (DL) is a subset of Machine Learning (ML) based on artificial neural networks (ANN), typically claimed to mimic the human brain allowing it to learn from a large dataset. It is inspired by information processing of biological systems and has been successfully applied to problems such as computer vision [14], speech recognition [15], drug design [16], medical image analysis [17], material inspection [18], board and video game [19] and in some of these even surpassing the human expert performance [20]. Algorithms and DL networks can be of immense complexity to tackle a specific problem. Some examples of supervised learning algorithms are generally distinguished as i) Multilayer perceptrons (MLP): Consisting of a collection of connected units (neurons) which act together as a network to map an input to an output; ii) Convolutional neural networks (CNNs): Commonly used to analyze visual imagery and are a regularized version of MLP. CNNs take advantage of hierarchical patterns in data, and are a specialized type of neural networks that uses the convolution mathematical operation instead of matrix multiplication [21]; iii) Recurrent neural networks (RNNs): Typically used for sequential data or time series analysis, commonly used in natural language processing and speech recognition. iv) Graphical neural networks (GNNs): Incorporating graphical models to capture intricate relationships between variables, particularly useful when complex dependencies exist among data elements organized in a graph structure. One application of ML models in CFD is in assisting flow solvers by replacing computations with surrogate models that improve execution time whilst decreasing resource consumption. One candidate is the Poisson equation used to compute the pressure field as it requires significant computational effort. Additionally, the ability to find solutions to the Poisson equation is not merely useful for fluid dynamics problems, but can be applied to other physics problems such as heat transfer, electromagnetism, and astronomy. Recent studies solved Poisson equations with ML techniques employing CNNs [22, 23] or even GNNs [24]. Examples of applications to CFD include Özbay et al [25] where a CNN

architecture was used to compute the pressure field for 2D cartesian meshes, Illarramendi et al [26] which coupled a CNN model with a traditional interactive solver to ensure user-defined accuracy and Weymouth [27] where a technique to accelerate classic Multi-Grid methods was proposed. Nonetheless, there are open challenges such as the generalization to different geometries and the ability of ML-based surrogate models to cope with laminar and turbulent flow regimes.

In this work, several fluid flow simulations were computed using the open-source CFD software OpenFOAM v6 [28] to construct multiple datasets. An algorithm capable of dealing with any geometry was designed and used to construct ML surrogate models capable of solving the pressure Poisson equation in CFD solvers. The novelty of the proposed approach is in its flexibility to tackle any kind of geometry by ensuring the neural network (NN) is decoupled from the CFD mesh. This enables the application of a trained surrogate model to any flow domain. Furthermore, application to different flow regimes was attempted by taking advantage of the the structure of the Poisson system of equations, as it was independent of assuming laminar or turbulent regime when solving the flow, the latter employing eddy-viscosity turbulence models.

Section 2 introduces the methodology, and Sect. 3 describes the numerical setup. The attained results are displayed in Sect. 4. Section 5 finishes by presenting the Conclusions.

2 Methodology | mathematical models

2.1 Fluid-flow solver

The fluid-flow simulation was performed with the OpenFOAM v6 [28] CFD toolbox, whose computer model is based on the NS equations discretized over a numerical grid using the Finite Volume Method (FVM) [29]. Both laminar and turbulent flow regimes were encompassed in a general set of equations with further changes associated with the respective regime: while for laminar flow the stress tensor is the Stokes law of friction [30], for turbulent flow a Boussinesq eddy-viscosity is employed together with further equations to model turbulence quantities.

All flow simulations were performed in a time-dependent approach by integrating the equations with the backward Euler scheme. The Pressure Implicit with Split Operator (PISO) algorithm [31] was used to ensure consistency between the velocity and pressure fields. The following text addresses a minimal description of the flow model sufficient to depict how ML methods can be integrated. Further details on CFD modeling may be found in

OpenFOAM [28] and Moukalled et al. [29], namely the treatment of boundary conditions and discretization for unstructured grids.

2.1.1 Laminar flow regime

Following Schlichting and Gersten [30] the conservation of momentum for a Newtonian fluid undergoing incompressible flow is governed by:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla) \mathbf{u} = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \mathbf{a}_b, \tag{1}$$

where \mathbf{u} is the flow velocity vector with each component describing the velocity along each spatial coordinate, p corresponds to the kinematic pressure,¹ $\boldsymbol{\tau}$ is the deviatoric stress tensor and the \mathbf{a}_b term is the net acceleration of all body forces acting on the fluid. Tensor $\boldsymbol{\tau}$ is related to the rate-of-strain tensor, $\mathbf{S} = \frac{1}{2}[\nabla \mathbf{u} + (\nabla \mathbf{u})^T]$, by

$$\boldsymbol{\tau} = \nu \left[2\mathbf{S} - \mathbf{I} \frac{2}{3} \nabla \cdot \mathbf{u} \right], \tag{2}$$

where \mathbf{I} is the identity tensor. Incompressible flow is characterized by negligible changes in density during fluid flow and can be assumed in a wide range of applications. This is not exclusive of liquids, rather, it applies to any fluid under specific flow conditions, highlighting its broad applicability. Considering incompressibility, the velocity field becomes solenoidal, and a continuity equation governing the conservation of mass is given by:

$$\nabla \cdot \mathbf{u} = 0. \tag{3}$$

In addition to eqs. (1–3), initial and spatial boundary conditions must be specified to fully solve the system.

2.1.2 Turbulent flow regime

Many engineering applications focus on flows under turbulent regimes. The main characteristic of this flow regime is the presence of eddies and vortices, which promote better mixing and more fluid momentum transfer in contrast to laminar flow [32]. According to Wang et al [33], turbulent flow is characterized by chaotic and irregular fluid motion.

Turbulent flows of Newtonian fluids are associated with high Reynolds number (Re) defined as $Re = UL/\nu$ with U and L being representative scales of velocity and length, while ν is the fluid kinematic viscosity [30].

Under turbulent regime, generally it is not feasible to simulate the instantaneous flow field as the required computer resources would be prohibitive. Instead, the flow may

¹ Kinematic pressure is the fluid pressure divided by the fluid density, thus given in units of $length^2 \cdot time^{-2}$ instead of $mass \cdot length^{-1} \cdot time^{-2}$.

be described in terms of the statistics of its fields. To tackle this the NS equations are averaged to establish a mean field, with any deviation from it being modeled as a fluctuation field, in a procedure known as Reynolds decomposition. Following an unsteady Reynolds-averaged Navier–Stokes (uRaNS) approach [30], the \mathbf{u} field becomes representative of the average fluid velocity. This allows to use the same formulation as in eqs. (1) and (3), with changes to the deviatoric stress tensor $\boldsymbol{\tau}$ to accommodate for extra terms, i.e. Reynolds stresses, together with further changes to the boundary conditions near walls.

The stress terms for flow under turbulent regime were modeled via a Boussinesq eddy-viscosity and the $k-\omega$ SST turbulence model, with resolved laminar sub-layer to better cope with adverse pressure gradients. A detailed description is given in Appendix A.

2.1.3 Pressure-momentum coupling

Analytical solutions for most flows are currently unattainable due to the mathematical properties of eq. (1), namely the non-linearity of the advection term and the need to simultaneously solve the continuity eq. (3), i.e. the coupling between \mathbf{u} and p . The PISO algorithm [31] was proposed to address this problem as it provides stable results and reduces computational effort when compared to iterative methods [34]. Rather than solving all the coupled equations iteratively, PISO splits the operator into an implicit momentum predictor and multiple explicit correction steps.

An algebraic system of equations was obtained by employing the FVM to discretize eq. (1) in a computational grid [29], resulting in:

$$\mathbf{M} \cdot \mathbf{U} = -\nabla p + \mathbf{a}_b, \tag{4}$$

where \mathbf{M} is a matrix of coefficients, and \mathbf{U} the velocity field. Note that eq. (4) is the same for both laminar and turbulent flow regime formulations, as these merely imply changes to values in \mathbf{M} .

A Poisson equation for pressure is derived by coupling eq. (4) with eq. (3):

$$\nabla \cdot (\mathbf{A}^{-1} \nabla p) = \nabla \cdot (\mathbf{A}^{-1} \mathbf{H} + \mathbf{A}^{-1} \mathbf{a}_b), \tag{5}$$

where \mathbf{A} is the matrix with the diagonal components of \mathbf{M} , and \mathbf{H} arises from the $\mathbf{M}\mathbf{U} - \mathbf{a}_b = \mathbf{A}\mathbf{U} - \mathbf{H}$ decomposition. Equation (5) can be solved to yield a pressure field that can be used to correct the velocity field to satisfy mass conservation.

Considering constant viscosity a similar procedure may be used to derive an analytical form of the pressure Poisson equation by taking the divergence of eq. (1), resulting in:

$$\frac{\partial}{\partial x_i} \left(\frac{\partial p}{\partial x_i} \right) = - \frac{\partial}{\partial x_i} \left[\frac{\partial (u_i u_j)}{\partial x_j} \right], \tag{6}$$

as shown by Ferziger and Perić [35] in index notation which is valid for laminar flow after subtracting the hydrostatic pressure from the p field, i.e. equivalent to removing the gravity acceleration from the \mathbf{a}_b term. Once again considering incompressibility, in vector form eq. (6) can be shown to be equivalent to

$$\nabla^2 p = -\nabla \mathbf{u} : (\nabla \mathbf{u})^T. \tag{7}$$

2.2 Surrogate model

2.2.1 Fundamentals of Deep Learning

The basic unit of the ANN is an artificial neuron and its output follows the equation

$$y = f \left(\sum_{i=1}^N x_i w_i + b_i \right), \tag{8}$$

where f represents the activation function, w_i the weight of the neuron i , and b_i the bias of the present neuron. A neural network consists of multiple connected layers which consist of multiple neurons. The output of a node i from layer l is calculated as:

$$a_i^l = f(z_i^l) = f \left(\sum_{j=1}^N x_j^l w_{ji}^l + b_i^l \right), \tag{9}$$

where f can be any chosen activation function, and w_{ji}^l the weight of the neuron j , where j represents nodes at layer $l - 1$, in neuron i from layer l . The bias b_i^l represents a zero-order coefficient associated with node i from layer l . The training process consists of sequentially computing each node output a_i^l from the input layer through all the hidden layers until the last as in eq. (9). This process is called forward propagation which ends in a computed prediction. The predicted output Y_{pred} can be compared to the known label Y , and the discrepancy between these values is evaluated through a chosen loss function as in

$$J = L(Y_{pred}, Y). \tag{10}$$

The derivatives of the loss in relation to the weights w_{ji} and bias b_i^l at every node i and layer l are also computed and used to update the weights in a process called backward propagation.

2.2.2 Model input

The surrogate model aims to match a set of input fields to the pressure field p , such that the latter can be predicted

from such inputs. The inputs consist of flow-field data and an extra argument that represents the geometry. A signed-distance function (SDF) was established so to provide information on the distance to the nearest wall at each position \mathbf{x} of the computational domain, defined as:

$$\text{SDF}(\mathbf{x}) = \begin{cases} \min(\delta(\mathbf{x}, \partial\Omega_b), \delta(\mathbf{x}, \partial\Omega_o)) & \text{if } \mathbf{x} \in \Omega, \\ 0 & \text{if } \mathbf{x} \in \Omega_o, \end{cases} \tag{11}$$

where Ω represents the computational domain where fluid exists and Ω_o the domain encompassed by the obstacle. The $\partial\Omega_b$ and $\partial\Omega_o$ represent the boundaries of Ω and Ω_o , respectively.

Two model families were established based on how the flow field is input into the model: \mathbf{M}_u and $\mathbf{M}_{f(\mathbf{u})}$. An overview of the model families is illustrated in Fig. 1. In \mathbf{M}_u models the input flow field corresponds to the components of the velocity field \mathbf{u} . An alternative family of models, $\mathbf{M}_{f(\mathbf{u})}$, is established by using the r.h.s. of eq. (7) as input, hence $f(\mathbf{u}) = -\nabla \mathbf{u} : (\nabla \mathbf{u})^T$.

2.2.3 Data sampling and normalization

The CFD model was run to provide data fields to train the ML model, and to serve as a reference to assess the error of the ML model predictions. To ensure the flow fields are referred to consistently, a characteristic time-scale was defined as $t^* = \phi / U_{\infty,c}$ where ϕ is the characteristic length and $U_{\infty,c}$ is the centerline velocity at the inlet. The CFD model was run for a total time of $300 t^*$ to fully capture the dynamic behavior of the flow. The sampling was performed with a time step of $\Delta t = 3 t^*$, resulting in 100 fields extracted from each simulation run.

The flow fields were normalized as $\mathbf{u}^* = \mathbf{u} / \max(|\mathbf{u}|)$ or $f(\mathbf{u})^* = f(\mathbf{u}) / \max(|\mathbf{u}|^2)$, depending on the model family \mathbf{M}_u or $\mathbf{M}_{f(\mathbf{u})}$, respectively. The kinematic pressure field was normalized as $p^* = p / \max(|\mathbf{u}|^2)$.

Instead of training the ML model with a monolithic dataset (as done in most applications of NN models) each temporal frame of a CFD simulation was decomposed into hundreds of thousands of different blocks, as illustrated in Fig. 2. This allowed generalization to different geometries. Using Latin Hyper-cube Sampling (LHS) [36] N blocks were sampled from the CFD domain.

Training with the real values of pressure after pre-processing and sampling would consist of an ill-posed optimization problem as the random pressure offsets in the solution are not correlated with the inputs, i.e. the same input value can have multiple different outputs depending on the boundary conditions. To tackle this, the pressure spatial mean, $\overline{p^*}$, was removed from each output block, e.g.

Fig. 1 Model families classified based on the input fields

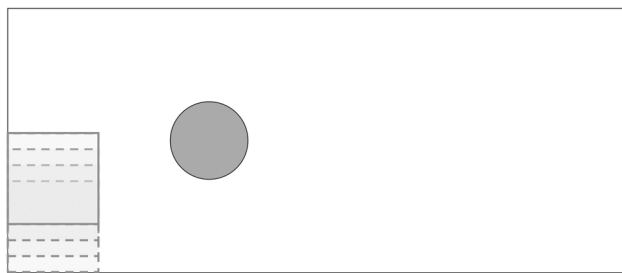
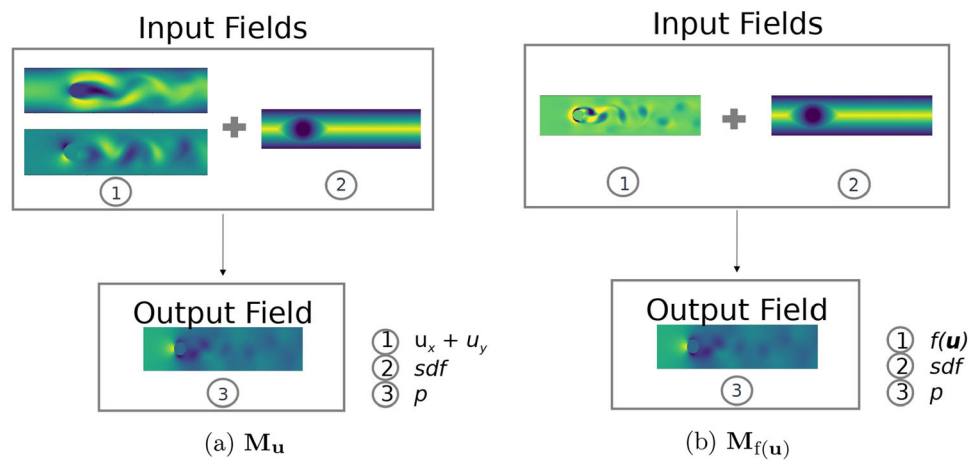


Fig. 2 Representation of the sampling method from the original domain

$p' = p^* - \bar{p}^*$. The normalization of a field quantity ϕ was $\phi^* = \phi / \max(|\phi|)$, hence bounding ϕ^* to $[-1, 1]$.

The loss function for eq. (10) was defined as the mean squared error (MSE),

$$MSE = \frac{1}{N_{points}} \sum_{i=1}^{N_{points}} [\hat{\theta} - \theta]^2, \tag{12}$$

where N_{points} was the number of grid points where an estimation was computed, θ corresponds to the reference values, i.e. the CFD results, and $\hat{\theta}$ the values computed with the surrogate model on this set of points.

The error metrics considered for evaluating the models accuracy consisted of BIAS,

$$BIAS = \frac{1}{N_{points}} \sum_{i=1}^{N_{points}} (\hat{\theta} - \theta), \tag{13}$$

standard-deviation error (STDE),

$$STDE = \left[\frac{1}{N_{points}-1} \sum_{i=1}^{N_{points}} (\hat{\theta} - \bar{\hat{\theta}})^2 \right]^{\frac{1}{2}}, \tag{14}$$

where $\bar{\hat{\theta}}$ is the mean of the predicted values, and root mean squared error (RMSE)

$$RMSE = \left[\frac{1}{N_{points}} \sum_{i=1}^{N_{points}} (\hat{\theta} - \theta)^2 \right]^{\frac{1}{2}}, \tag{15}$$

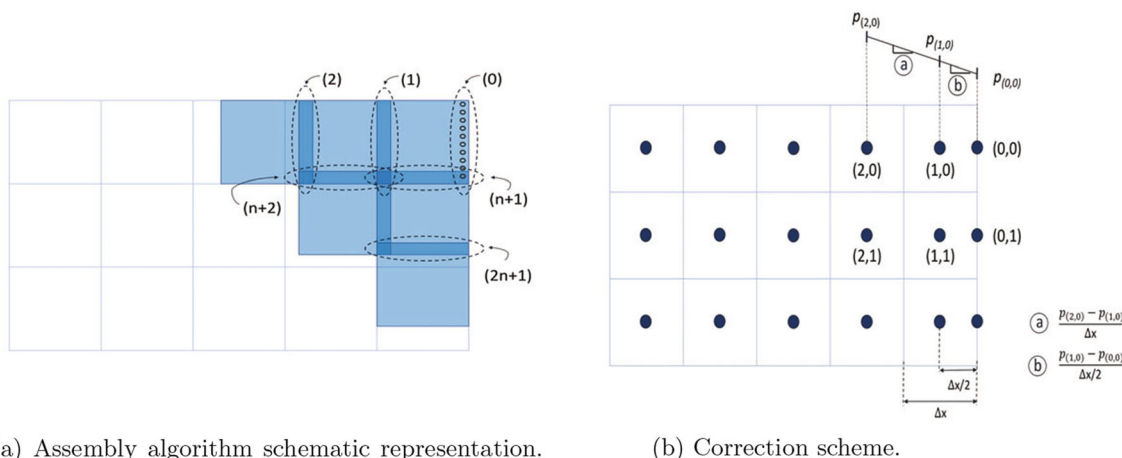
and afterward normalizing each of those by dividing them by $(\max(\theta) - \min(\theta))$.

The set of CFD results consisted of simulations characterized by different Re numbers defined as $Re = \overline{U}_{\infty} D / \nu$, where D is the cylinder diameter and \overline{U}_{∞} the mean longitudinal inlet velocity.

2.3 Blocks assembly into the flow domain

Once the solution field for each block is predicted by the ML surrogate model, an assembly algorithm was developed to reconstruct the full pressure field, as illustrated in Fig. 3a. The pressure value at location (0) is taken as the reference pressure to ensure the fulfillment of the outlet fixed value boundary condition and was set to 0 for simplicity. The assembly process begins by placing the predicted field of the first block into the domain and applying its initial correction at (0) to meet the boundary condition. Subsequently, the predicted field of the second block from the same row is placed, and a correction is applied at (1) to ensure consistency in the intersection zone, facilitating coherence between the two blocks. This incremental placement and correction process is repeated until reaching the left end of the domain for the first row. Similar reconstructions and corresponding corrections are done incrementally up to the left end of the domain. Once the first row is completed, corrections are made at the upper end of the new block in the intersection zone indicated by (n+1). This iterative process continues until the entire domain is fully defined.

Having considered the fixed value boundary condition at the furthest downstream centroid (and not in the cell face where the boundary condition is defined) a correction must be applied to the whole domain. Considering the schematic



(a) Assembly algorithm schematic representation.

(b) Correction scheme.

Fig. 3 Assembly algorithm methods

representation from Fig. 3b, enforcing the gradient in the nearby upstream cell to be equal to the gradient at the outlet boundary, i.e. $(p_{(1,0)} - p_{(0,0)}) / (\Delta x / 2) = (p_{(2,0)} - p_{(1,0)}) / \Delta x$, one can estimate the deviation caused by the incorrect assumption reaching

$$p_{\text{deviation}} = p_{(1,:)} - \frac{1}{3} p_{(2,:)} \tag{16}$$

The corrected pressure field is computed by removing the deviation, $p_{\text{corr}} = p - p_{\text{deviation}}$, leading to consistent results with respect to the reference fields.

The model architecture, size of the blocks, and NN training-related hyper-parameters such as batch size, learning ratio, and moving average parameters were properly optimized. Since some of these (such as the model architecture and the size of the blocks) are not easily decoupled from the assembly algorithm, all of the above were optimized considering the surrogate model as a whole, i.e. using the RMSE normalized of the surrogate model results as the target. The alternative would be to consider the parameters that better optimize the NN individually but not in the way it interacts with the assembly process. The only parameter whose influence could be partially decoupled from the NN training was the overlap ratio, i.e. the ratio between the overlap region size and the block size, but it was also optimized considering the whole surrogate model accuracy. Refer to Appendix B to follow the parameters selection process.

2.4 Neural Network architecture

The NN model developed can be classified as an MLP with 3 hidden layers, whose purpose is to act as a transfer function between velocity and pressure values. Following Sousa [37] a Principle Component Analysis (PCA) was

applied to the input and output layers of the NN model, hence acting as encoder and decoder stages as shown in Fig. 4. Rectified Linear Unit (RELU) and Linear activation functions were used in the hidden and output layers, respectively.

The application of PCA and its subsequent truncation to a few Principal Components (PC) that explain most of the variance found on the input data saves large amounts of computational resources since it helps the model training by reducing the dimensionality of the input/output data, and only yield the most relevant information.

In the current setup, the PCs were attained from the normalized values directly (cf. Sect. 2.2.3) without applying further operations such as standardization.

3 Numerical setup

3.1 Training and evaluation method

After proper hyperparameter tuning, the training was done with the different families of geometries listed in Table 1 and the same Re number, but its performance was also accessed for different flow conditions to study the generalization capability. The training convergence criteria consisted of stopping whenever the decrease in the loss value, computed from eq. (10), considering the test dataset (i.e. the group of blocks sampled according to Sect. 2.2.3) was lower than 0.1% over 250 epochs. Each of the following evaluations was performed based on 100 temporal frames from 4 different simulations with different obstacles (from the same family of geometries) providing a representative evaluation of the model performance over the whole family of geometries as well as predicting the dynamical behavior of each one. The notation to identify the models is defined in Table 2.

Fig. 4 Schematic representation of the multilayer perceptron (MLP) neural network. Adapted from Liang et al [38]

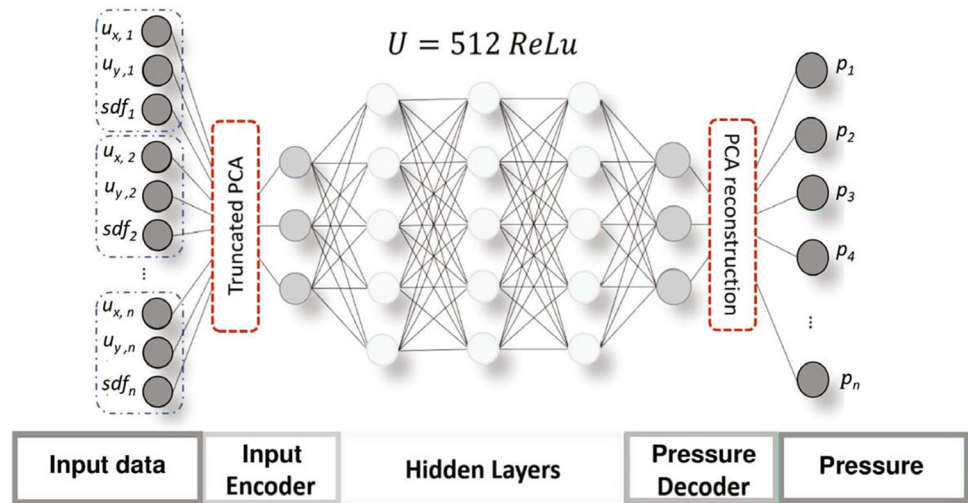


Table 1 Dataset symbology and kind of obstacle

Dataset	Symbol
Circular obstacle	◦
Square obstacle	◻
Triangular obstacle	◁
Inclined plate obstacle	/

Table 2 Description of the models and their symbology based on the training datasets

Training datasets	Model symbol
◦	\mathbf{M}_{\circ}
◦, ◻	$\mathbf{M}_{\circ, \square}$
◦, ◻, ◁	$\mathbf{M}_{\circ, \square, \triangleleft}$
◦, ◻, ◁, /	$\mathbf{M}_{\circ, \square, \triangleleft, /}$

Regarding the number of blocks to be extracted from each temporal frame, N , it was necessary to arrive at a good compromise between accuracy and training cost. Using all available blocks would be infeasible in terms of computational cost and would bring mostly redundant information, therefore two distinct N were tested: $N_1 = 500$ and $N_2 = 2500$ (resulting in a total number of blocks per simulation of $N_{total,1} = 5 \times 10^4$ and $N_{total,2} = 2.5 \times 10^5$ for N_1 and N_2 , respectively). Further details on the size of the datasets used are provided in Appendix C for models $\mathbf{M}_{\mathbf{u}}$ and $\mathbf{M}_{f(\mathbf{u})}$. It is shown that the application of PCA achieves dimensionality reduction (i.e. reduction of the total number of points attained by multiplying the input and output sizes by the number of samples) of up to 3 orders of magnitude.

The convergence curves from the training of each model can also be found in Appendix C.

3.2 Data generation - CFD setup

The training and test cases consisted of flow around a bi-dimensional obstacle in confined flow as represented in Fig. 5 for the ◦ obstacle, where H is the distance between plates, and \hat{n} the unit vector perpendicular to the domain boundary at any point. The boundaries and corresponding boundary conditions are defined in Table 3 according to Fig. 5 following OpenFOAM [28] nomenclature. The flow was induced by the inlet velocity where it to was specified to follow the analytical solution for fully developed laminar flow between parallel plates given by $u(y) = 3\overline{U_{\infty}}/2 [1 - (y/h)^2]$ where y is the distance from the centerline, h half of the distance between plates and $\overline{U_{\infty}}$ the mean longitudinal velocity.

The generated meshes had finer refinement near every wall as illustrated in Fig. 6.

The simulations performed at high Re numbers had a different numerical setup. Since for pressure and velocity the boundary conditions were maintained, only the boundary conditions for the additional variables: the turbulent kinetic energy, k , and specific turbulent dissipation rate, ω , are defined in Table 4 following OpenFOAM [28] nomenclature. The free-stream values, i.e. at the inlet, had to be estimated, hence the following relations were taken

$$k_{\infty} = \frac{3}{2}(U_{\infty}I)^2, \tag{17}$$

U_{∞} is the free stream velocity, here defined as the mean inlet velocity and I the turbulence intensity which was set to 10% and

Fig. 5 Simulation case representation with domain dimensions and boundary conditions

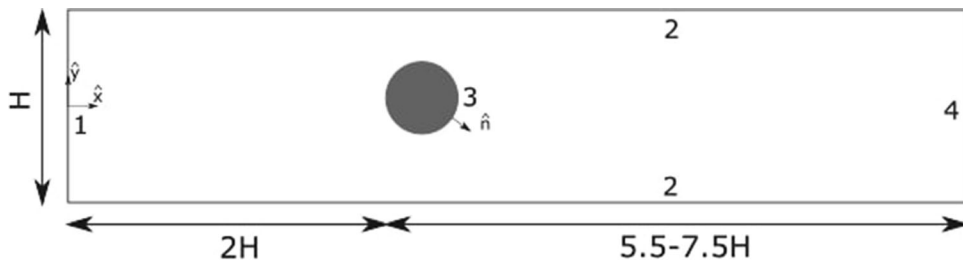


Table 3 Boundary conditions for laminar flow

Boundary	u	p
1 - Inlet	codedFixedValue	zeroGradient
2 - Top & bottom walls	noSlip	fixedFluxPressure
3 - Obstacle	noSlip	fixedFluxPressure
4 - Outlet	pressureInletOutletVelocity	fixedValue

Fig. 6 Mesh illustration for each obstacle: **a** ○, **b** □, **c** /, **d** ◁. For illustration purposes, the mesh density is $\frac{1}{100}$ of the original

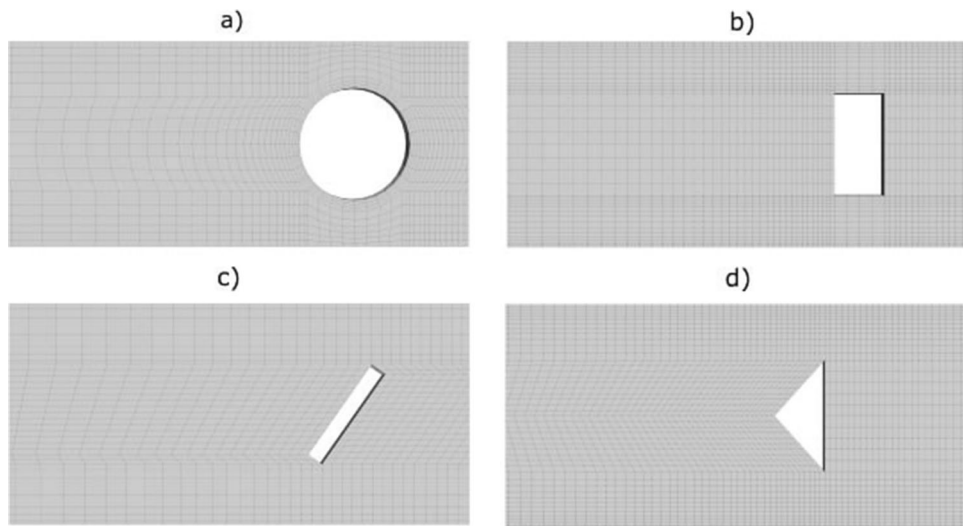


Table 4 Boundary conditions for turbulent flow quantities

Boundary	k	ω
1 - Inlet	fixedValue	fixedValue
2 - Top & bottom walls	fixedValue	omegaWallFunction
3 - Obstacle	fixedValue	omegaWallFunction
4 - Outlet	inletOutlet	inletOutlet

$$\omega_\infty = C_\mu^{-\frac{1}{4}} \frac{\sqrt{k}}{l}, \tag{18}$$

where l is the turbulent length scale which was defined based on the characteristic inlet length scale, L , as $l = 0.07L$ [39], C_μ is a turbulence model constant which was set to $C_\mu = 0.09$. These free stream conditions were taken following OpenCFD [40] guidelines in agreement with the broader recommendation on Menter [41]. The estimations used for the turbulent quantities are not precise, hence the region of interest, i.e. the obstacle, was distanced

$10L$ and $25L$ from the inlet and outlet, respectively, with L as the characteristic length. The boundary conditions are defined using the nomenclature from OpenFOAM, further information can be found either in the documentation or in the source code.

The mesh was refined to guarantee that the elements adjacent to the wall were within the viscous sub-layer, hence at $y^+ < 5$ [42, c.f.]. This practice avoids the use of logarithmic boundary layer wall functions, as that implies constant-stress conditions for those near-wall grid cells.

4 Results

4.1 Model M_u

The first results obtained consisted in the testing of M_{\circ} , $M_{\circ, \square}$, $M_{\circ, \square, \triangleleft}$, and $M_{\circ, \square, \triangleleft, /}$ to predict within all the families of geometries in flows characterized by the training Re number, $Re = 100$, as presented in Table 5.

From Table 5 an increase in training data diversity allows the model to improve its generalization to multiple geometries. Incrementation in training datasets resulted in a tendency to increase overall performance, but also impaired results for a particular dataset, i.e. a model trained with multiple datasets (including the test data) had lower performance in a particular obstacle than the model trained only with the test dataset. This effect came from the difficulty of overfitting to a particular case. Regarding the number of samples from each time frame, N , it is already possible to conclude the inadequacy of using N_2 datasets since the improvements seen do not compensate for the training increased computational cost. This amount of samples from each flow field introduced redundant data leading to overfit. To visually illustrate predictions from Table 5, in Fig. 7 a single flow field is shown for each geometry.

Table 6 shows the contribution of each dataset to the training to inspect the influence of each training set on the surrogate model accuracy. The results from M_{\triangleleft} showed to be promising since, by learning from only the \triangleleft dataset, the model could make accurate predictions across all the other flows. The overall accuracy across different datasets was

accompanied by the difficulty in predicting the training datasets, i.e., the higher complexity of these flows dynamical structures. This allowed the NN to learn different flow patterns with only one family of geometries, indicating a richer learning process. Therefore, an increase in the number of simulations or further obtaining larger flow diversity, by producing additional simulations with the same obstacles tilted in the \triangleleft and $/$ datasets, could be very beneficial to the training.

The uncertainty in the results may be decomposed into two parts: the uncertainty of the trained NN to accurately output the pressure in each block regardless of the rest of the flow domain geometry; and the uncertainty in the assembly procedure.

As mentioned in Sect. 2.3 the assembly procedure ensures the consistency between neighbor blocks by removing bias estimated based on the overlapping regions. The discussion of the results was focused on the first part of uncertainty, but the assembly algorithm may be responsible for most of the remaining error seen in the easier-to-predict cases (such as the \circ and \square datasets), and by increasing error in the cases where the NN performs the worst. Small errors in the blocks may be heavily weighted by this algorithm. The uncertainty here arises both from how the domain is integrated, i.e. whether it is integrated over the longitudinal direction and afterward spanwise or the reverse order, and the way the agreement between the pressure field of neighbor blocks is ensured.

Tested each surrogate model at $Re = 100$, its performance in extrapolation was inspected on flow conditions corresponding to $Re = 10, 50, 500$ and 1000 flows over the

Table 5 M_u results from training with multiple datasets and tested in $Re = 100$ flows. Every error metric, $Error(\phi)$ was normalized according to $Error(\phi)_{norm} = Error(\phi)/(\phi_{max} - \phi_{min})$

Model	Error metric (%)	\circ dataset		\square dataset		\triangleleft dataset		$/$ dataset	
		N1	N2	N1	N2	N1	N2	N1	N2
M_{\circ}	BIAS	0.42	- 0.04	0.61	0.30	- 6.90	- 9.06	- 0.11	- 2.49
	STDE	1.13	1.03	2.08	2.45	10.01	12.3	12.87	13.84
	RMSE	1.20	1.04	2.17	2.46	12.16	15.27	12.87	14.07
$M_{\circ, \square}$	BIAS	- 0.83	- 0.28	- 0.33	- 0.17	- 8.86	- 8.71	- 4.51	- 2.76
	STDE	1.32	1.19	0.91	0.87	10.76	10.09	15.37	12.66
	RMSE	1.56	1.22	0.97	0.89	13.94	13.33	16.02	13.93
$M_{\circ, \square, \triangleleft}$	BIAS	- 0.17	- 0.62	- 0.13	- 0.36	- 0.18	0.86	3.25	3.53
	STDE	1.70	1.58	1.00	0.98	1.89	1.81	5.51	5.60
	RMSE	1.71	1.70	1.01	1.04	1.90	2.00	6.39	6.62
$M_{\circ, \square, \triangleleft, /}$	BIAS	- 0.48	- 1.11	- 0.45	- 0.45	- 0.54	- 0.51	0.20	- 0.64
	STDE	1.90	2.07	1.34	1.49	1.85	2.14	3.01	3.06
	RMSE	1.96	2.35	1.41	1.56	1.93	2.20	3.02	3.13

Bold values indicate the best results

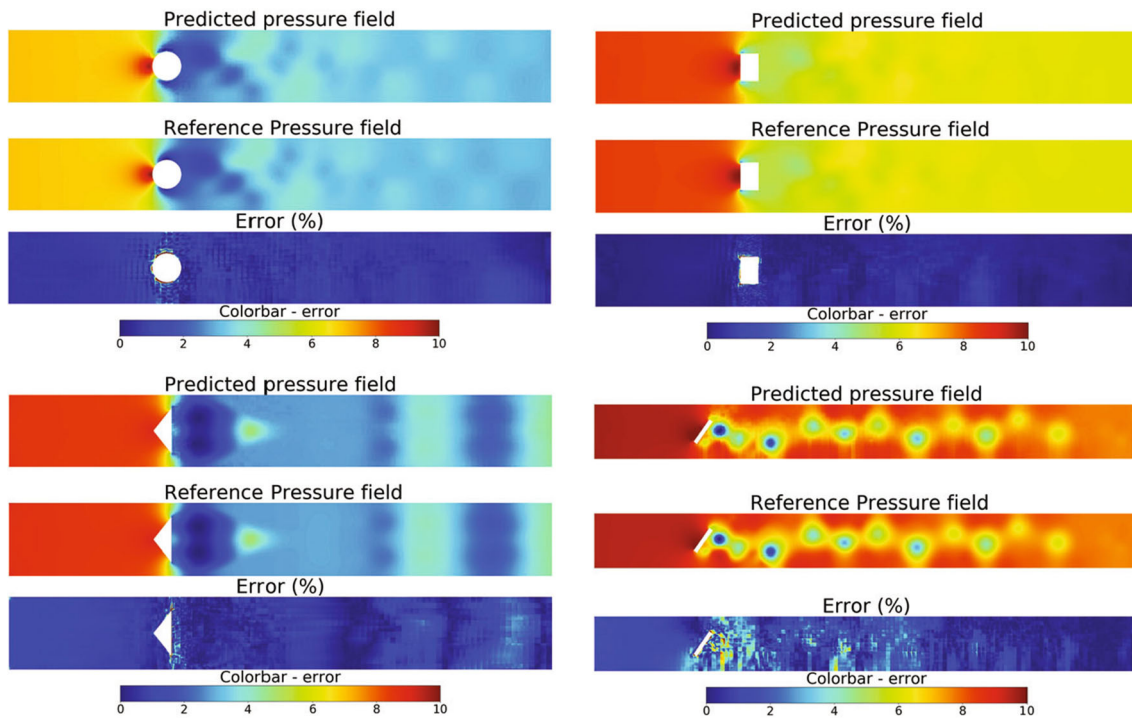


Fig. 7 M_u prediction examples for each geometry at $Re = 100$

Table 6 M_u results trained with each dataset and tested in $Re = 100$ flows. Every error metric, $Error(\phi)$ was normalized according to $Error(\phi)_{norm} = Error(\phi)/(\phi_{max} - \phi_{min})$

Model	Error metric (%)	o dataset		□ dataset		◁ dataset		/ dataset	
		N1	N2	N1	N2	N1	N2	N1	N2
M_o	BIAS	0.42	− 0.04	0.61	0.30	− 6.90	− 9.06	− 0.11	− 2.49
	STDE	1.13	1.03	2.08	2.45	10.01	12.30	12.87	13.84
	RMSE	1.20	1.04	2.17	2.46	12.16	15.27	12.87	14.07
M_{\square}	BIAS	− 1.96	− 3.17	0.04	− 0.27	− 11.50	− 9.05	− 2.32	− 3.78
	STDE	4.05	6.48	0.77	0.75	13.53	11.16	12.02	12.79
	RMSE	4.50	7.22	0.77	0.80	17.75	14.37	12.24	13.37
M_{\triangleleft}	BIAS	− 0.99	− 2.52	− 1.04	− 1.58	− 0.03	− 0.39	3.03	2.98
	STDE	3.89	4.28	2.56	3.35	1.83	1.42	5.81	6.23
	RMSE	4.01	4.97	2.77	3.70	1.83	1.47	6.55	5.47
$M_{/}$	BIAS	− 2.17	− 2.78	− 0.66	− 1.55	− 6.16	0.70	− 1.34	− 1.16
	STDE	4.88	5.84	3.60	4.42	9.04	9.25	3.15	2.53
	RMSE	5.34	6.47	3.96	4.68	10.94	9.28	3.42	2.78

Bold values indicate the best results

o obstacle in Table 7. N_2 datasets did not bring important improvements in the tests at the training Re number, therefore N_2 results were removed from the following tables. The reference data to evaluate the predictions within turbulent regime consisted of simulations at $Re = 3 \times 10^5$ and 4×10^5 . The results from these evaluations were also appended to Table 7.

Table 7 results showed the difficulty of generalization to other flow conditions, particularly at low- Re numbers where the flow becomes stationary with no dynamical structures, and at Re numbers orders of magnitude higher than the training Re number. As the test simulations were at different flow regimes, the NN failed to predict the pressure upstream of the obstacle and the overall pressure gradient, resulting in large errors. At $Re = 10$, friction

Table 7 M_u extrapolation test in the laminar regime. Prediction case: \circ dataset. Every error metric, $Error(\phi)$ was normalized according to $Error(\phi)_{norm} = Error(\phi)/(\phi_{max} - \phi_{min})$

Model	Error metric (%)	Re = 10	Re = 50	Re = 500	Re = 1000	Turbulent
M_{\circ}	BIAS	- 28.36	- 4.62	- 3.58	- 4.92	54.82
	STDE	28.43	13.05	14.42	17.62	33.38
	RMSE	40.16	13.84	14.86	19.23	64.19
M_{\square}	BIAS	- 29.9	- 13.41	- 16.44	- 20.52	48.68
	STDE	30.1	15.20	14.99	22.89	31.12
	RMSE	42.43	20.27	22.25	30.74	57.78
M_{\triangleleft}	BIAS	- 29.07	- 2.75	7.53	6.60	54.74
	STDE	29.07	15.70	14.21	16.25	28.09
	RMSE	41.11	15.94	16.08	17.54	51.66
$M_{/}$	BIAS	- 28.43	- 3.11	1.43	- 1.79	43.36
	STDE	27.58	14.13	13.54	15.97	33.38
	RMSE	39.61	14.47	13.62	16.07	64.19
$M_{\circ, \square}$	BIAS	- 28.86	- 9.62	- 5.93	- 10.14	46.99
	STDE	28.64	12.92	13.83	16.95	31.51
	RMSE	40.66	16.32	15.05	19.75	56.58
$M_{\circ, \square, \triangleleft}$	BIAS	- 29.12	- 5.62	3.95	0.60	52.78
	STDE	29.35	13.26	13.16	15.58	34.09
	RMSE	41.34	14.40	13.74	15.59	34.57
$M_{\circ, \square, \triangleleft, /}$	BIAS	- 28.37	- 5.10	1.85	2.35	51.92
	STDE	28.36	11.2	10.08	11.68	35.49
	RMSE	40.11	12.31	10.24	11.91	62.90

Bold values indicate the best results

losses introduced by the obstacle resulted in an increase of the pressure upstream and steeper pressure gradients in the downstream region (cf. Fig. 8a). On the opposite side, at turbulent regime, $Re = 3 \times 10^5$, friction losses were overpredicted (cf. Fig. 8b). Both were not accurately predicted. The error was mainly due to BIAS.

In the laminar regime, friction losses are known to increase with the viscosity (or decrease with the Re number), i.e. with the decrease of the Re, therefore it is clear that in different Re number flows estimating these

quantities must correctly account for the viscosity. The surrogate models lacked information to correctly find the relation between velocity and pressure fields consistently for different Re without the explicit inclusion of the viscosity, thus the problem may be ill-posed for training with different Re numbers.

Despite the mentioned limitation, it was still possible to see an improvement trend within the tests at $Re = 50, 500,$ and $1000,$ as long as the flow conditions are similar. Training with more complicated geometries (cf. $M_{\circ, \square, \triangleleft},$

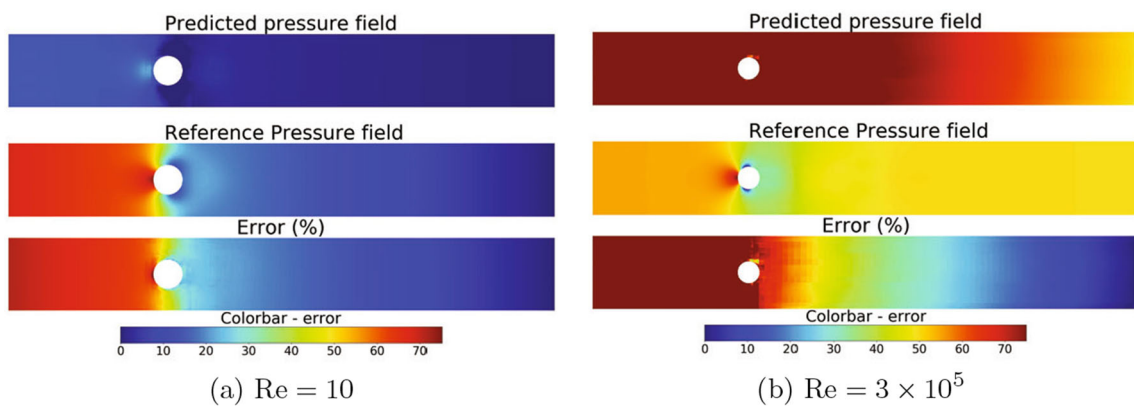


Fig. 8 M_u accuracy in extrapolation

/ results on Table 7), introduced different flow characteristics that could be similar to other Re flows in the \square cases, considerably improving the estimations. This trend could steadily go into lower errors, however, the ill-posed optimization problem may eventually set a lower limit impossible of being surpassed. This limit would go up in concordance with the increase in the Re test range.

The difficulties in extrapolation tests allowed us to understand the inadequacy of extending a model trained with only one Re number flows to general usage, however, an improving trend was visible as long as new training examples were added, hereby represented by the increment of simulation data with different obstacles to the training dataset, showing an increase in performance up until \mathbf{M}_{\square} , \square , \triangleleft , \lrcorner - \mathbf{u} in moderate Re-number flows. For both high and low Re numbers the results were completely inadequate (cf. Table 7). The results showed the model difficulty in estimating the upstream pressure, directly related to an inadequate prediction of the head pressure losses caused by the obstacle, and pressure gradient downstream of the object resulting in high BIAS. Further improving the surrogate model and allowing training/extrapolation to different Re numbers, it could be required to either pre-process the data differently, allowing to disconnect the pressure values from the viscous effects, or change the inputs and outputs given in training. More details about the training can be found in Appendix B.

4.2 Model $\mathbf{M}_{f(\mathbf{u})}$

The relevance of this approach comes from leveraging the knowledge of the differential equation governing the phenomenon. Results from $\mathbf{M}_{f(\mathbf{u})}$ were captured for each geometry in Table 8 enabling direct comparison with the $\mathbf{M}_{\mathbf{u}}$ results from Table 6.

The current approach proved challenging as the immense amount of variance present in the $f(\mathbf{u})$ field severely crippled the model learning process. However, with the proper pre-processing, given by the PCA transformation, it could potentially be used to generalize to any problem represented by the Poisson differential equation as in equation (7) provided the potential field here represented by $f(\mathbf{u})$ applied in its general form. Several PCs were necessary to represent a sufficient amount of the input variance, therefore an upper limit was defined: the number of PCs had to jointly represent up to 95% of the total variance without surpassing the upper limit defined by the hidden layers width (512 neurons). Since this was an additional test, hyperparameter tuning was not performed prior to the application of this NN.

From Table 8, $\mathbf{M}_{f(\mathbf{u})}$ had far lower performance when compared to results from models $\mathbf{M}_{\mathbf{u}}$, but with the

appropriate hyperparameter tuning these results have a large margin of improvement. However, some information was in agreement with the results from Table 6 with complex flow conditions providing richer training.

$\mathbf{M}_{f(\mathbf{u})}$ models were also tested in extrapolation and the results are shown in Table 9. Every model failed in predicting low Re number flows exactly as $\mathbf{M}_{\mathbf{u}}$ models due to the same reason. However, in medium to high-Re numbers, models \mathbf{M}_{\square} - $f(\mathbf{u})$ and \mathbf{M}_{\lrcorner} - $f(\mathbf{u})$ achieved better results.

$\mathbf{M}_{f(\mathbf{u})}$ in moderate Re numbers was able to achieve relatively good predictions when compared to its prediction at the training Re number, which suggests that the $f(\mathbf{u})$ input field could be more sensible to changes in Re number by heavily weighting variations in the velocity field.

At $\text{Re} = 10$, $\mathbf{M}_{f(\mathbf{u})}$ also provided unacceptable predictions. Lastly, in extrapolation into turbulent regime, the results from $\mathbf{M}_{f(\mathbf{u})}$ were far superior to results of models $\mathbf{M}_{\mathbf{u}}$, reaching an overall superior accuracy with every single model as illustrated in Fig. 9 for a single example. These results at $3000 - 4000 \times$ the training Re number may be an indication of the superior adequacy of using $f(\mathbf{u})$ as input to predict in such different flow conditions, or just an indication of the incomplete adjust to the training set providing moderate errors in a large range of flow conditions. Regarding the evaluation tests at turbulent regime, given that the simulation set was generated based on a uRaNS turbulence model simulation approach, the corresponding $f(\mathbf{u})$ field formulation should account for Reynolds stress terms. However, the DNS formulation, eq. (7), was taken to use the NNs previously trained with the laminar cases directly.

4.3 Performance evaluation

To assess the usefulness of the developed surrogate model, its computational performance was benchmarked against the conventional PISO algorithm as implemented in OpenFOAM pisoFoam solver. Model \mathbf{M}_{\square} , \square , \triangleleft , \lrcorner was selected for the benchmarking.

The overlap ratio, i.e. the ratio between the overlap region size and the block size is used to assemble the full flow domain based on multiple blocks (cf. Section 2.3 and Fig. 3), determines the number of calls to the NN and the number of calls to the reconstruction loop, hence it has high impact on the model performance. Therefore it is taken as a parameter on the following evaluations.

The evaluation procedure consisted of comparing the execution time taken in 1000 calls to the PISO method and the surrogate model using a single core from an Intel® Xeon® Processor E5-2650.

Figure 10 shows the speedup factor attained when using the surrogate model, with the PISO algorithm

Table 8 $M_{f(u)}$ results trained with each dataset and tested in $Re = 100$ flows. Every error metric, $Error(\phi)$ was normalized according to $Error(\phi)_{norm} = Error(\phi)/(\phi_{max} - \phi_{min})$

Model	Error metric (%)	○ dataset	□ dataset	◁ dataset	/ dataset
M_{\circ}	BIAS	0.59	− 0.68	− 4.25	2.52
	STDE	2.52	9.03	14.01	11.06
	RMSE	2.59	9.06	14.64	11.35
M_{\square}	BIAS	3.03	5.14	− 8.42	7.92
	STDE	18.26	12.80	14.30	13.24
	RMSE	18.51	13.80	16.60	15.43
M_{\triangleleft}	BIAS	− 8.88	− 6.40	1.08	− 5.76
	STDE	8.60	9.41	5.98	8.09
	RMSE	12.37	11.38	6.08	9.93
$M_{/}$	BIAS	− 3.15	− 2.22	− 3.43	− 5.82
	STDE	28.81	5.60	11.19	6.35
	RMSE	8.33	6.02	11.70	8.61

Bold values indicate the best results

Table 9 $M_{f(u)}$ extrapolation test. Prediction case: ○ dataset in laminar regime. Every error metric, $Error(\phi)$ was normalized according to $Error(\phi)_{norm} = Error(\phi)/(\phi_{max} - \phi_{min})$

Model	Error metric (%)	Re = 10	Re = 50	Re = 500	Re = 1000	Turbulent
M_{\circ}	BIAS	− 28.44	− 2.40	6.82	3.14	11.82
	STDE	27.92	14.07	12.74	17.58	13.58
	RMSE	39.85	14.27	14.45	17.86	18.01
M_{\square}	BIAS	− 26.68	− 12.57	− 6.65	− 19.39	5.95
	STDE	21.97	26.72	30.77	33.10	14.37
	RMSE	34.56	29.52	31.48	38.36	15.55
M_{\triangleleft}	BIAS	− 33.38	− 15.77	− 10.73	− 16.30	− 24.52
	STDE	30.16	14.78	13.37	17.33	12.36
	RMSE	44.99	21.62	17.15	23.79	27.45
$M_{/}$	BIAS	− 30.40	− 9.59	2.39	− 1.11	3.56
	STDE	28.81	10.55	12.29	12.22	8.50
	RMSE	41.95	14.26	12.52	12.27	9.22

Bold values indicate the best results

implementation as is in OpenFOAM as a reference. As detailed in Appendix B the results shown in the Results Section used an overlap ratio = 0.75 since it seemed to show the best possible results. However, the current results show that lower overlap ratios can be used to decrease computational cost with a very low loss of accuracy as shown in Fig. 11 which displays the average RMSE normalized as a function of the overlap ratio for all simulation types.

The simulation cases showed very little influence on the time of a surrogate model call (c.f. Tables 14, 15 and 16). The performance of the surrogate model was agnostic to the case geometric complexity since it predicts each block and reconstructs the full domain with the same set of steps independently of the simulation geometry. The number of cells in the CFD mesh had a low impact on the surrogate model, since it only impacted the interpolation operations, i.e. interpolating from a grid defined from the CFD mesh cell centers converted to a uniform grid of points from

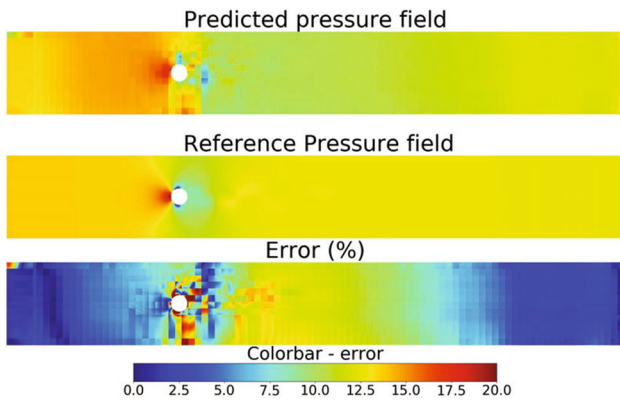


Fig. 9 $M_{f(u)}$ accuracy in extrapolation at $Re = 3 \times 10^5$

where the blocks were extracted. For a complete view of these results the reader may refer to Appendix D. In these detailed results, some uncertainty can be detected. This came from the variability in the high-performance computing node performance as a function of the load it was submitted to. This variability could not be removed.

The surrogate model yielded higher advantages in terms of computation expenses for complex cases (where the PISO would perform many iterations until reaching convergence) and cases with meshes with a high number of cells.

For the sake of visualization, only the average speedups attained in each type of flow case were shown. Tables with the time-dependent results are provided in Appendix D.

Fig. 10 Average speedup obtained by using the ML surrogate model to solve the pressure Poisson equation in a CFD simulation. The average speedup is defined as the ratio between the execution time of the PISO algorithm as implemented in OpenFOAM and the execution time of the surrogate model

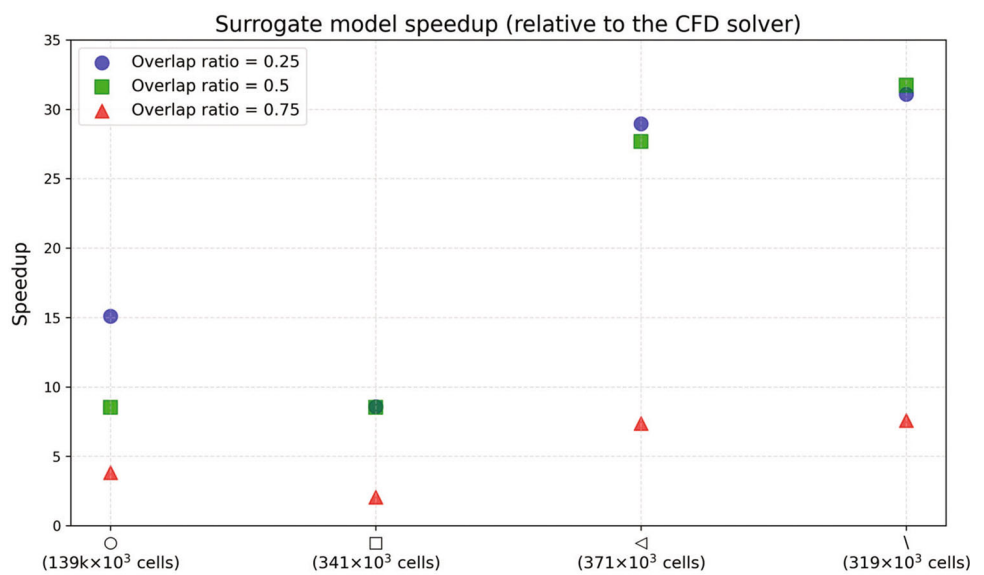
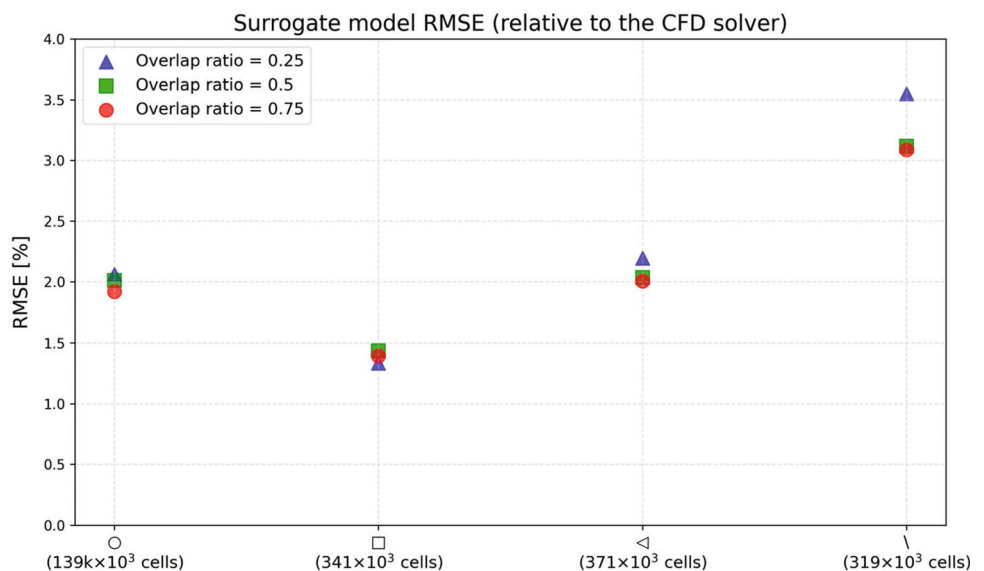


Fig. 11 Average RMSE normalized when using the ML surrogate model to solve the pressure Poisson equation in a CFD simulation. The results from the PISO algorithm, as implemented in OpenFOAM, were used as a reference



These results both in terms of performance and accuracy show that a combination of PCA and a simple MLP was found to be efficient and capable of being adequately trained with low computational resources. Using PCA to replace whether the encoding or decoding layers provided training efficiency as demonstrated in Sousa [37] and Appendix B.

5 Conclusions

An algorithm capable of dealing with any geometry was proposed and used to construct ML surrogate models capable of solving the pressure Poisson equation in CFD solvers with low to moderate errors by ensuring the neural network (NN) is decoupled from the CFD mesh. This enables the application of a trained surrogate model to generic flow geometries.

Two types of ML surrogate models were established based on the selected set of inputs: models that took the velocity field as input were referred to as \mathbf{M}_u , and the ones receiving the source term of the pressure Poisson equation as input were called $\mathbf{M}_{f(u)}$.

- \mathbf{M}_u models revealed to be good interpolators, managing to predict with very low error in every test case within training flow conditions (cf. Tables 5 and 6).
- \mathbf{M}_u models trained with every dataset ($\mathbf{M}_\square, \mathbf{M}_\square, \mathbf{M}_\square, \mathbf{M}_\square$) had a maximum amount of RMSE normalized of 3% and performed with RMSE normalized lower than 2% in the remaining datasets while enabling speedups of up to 30 times when compared to the classical PISO algorithm.
- The effect of a fivefold increase in the number of training samples from each CFD simulation (i.e. from N_1 to N_2) was depicted by Tables 6 and 7 and was shown not to be beneficial.
- Increased diversity in the training dataset improved the overall performance of each model in the training flow conditions and even in extrapolation (cf. Tables 5 and 7).
- The surrogate models failed at extrapolation into very different flow conditions, as those failed to learn the correct mapping between friction losses and the Re number. However, the improvement trend with the increase in training data diversity suggested the possibility of this model having practical application near the training conditions given that it is trained with larger diversified datasets.
- $\mathbf{M}_{f(u)}$ models showed to be able to learn how to solve the Poisson equation with medium errors and had some capacity for generalization. The moderate accuracy in some extrapolation tests accompanied by medium

errors in the training tests also indicated that the NN could not properly fit the training examples. This may have come from the non-optimization of the NN hyperparameters and the amount of variance from its input flow field.

- A combination of PCA and a simple MLP was found to be efficient and capable of being adequately trained with low computational resources, without losing accuracy. Using PCA to replace whether the encoding or decoding layers provided training efficiency.

Future works will focus on:

- Improving the training formulation of \mathbf{M}_u to enable a well-formulated optimization problem for learning flow characteristics over a wide range of Re numbers. It was hypothesized that having the pressure gradient (instead of the pressure) as the output of the NN and integrating it over the domain could lead to obtaining a pressure field with more consistent gradients over a wide range of flow conditions;
- Better tuning the $\mathbf{M}_{f(u)}$ parameters and adding additional pre-processing to the input field to access if this model can be capable of accurate extrapolation;
- Optimizing the assembly algorithm and reducing the uncertainty associated with it. This requires an uncertainty analysis of the assembly algorithm and NN separately, as individual components.

Appendix A The $k-\omega$ SST turbulence model

Following eq. (1) the Newtonian deviatoric stress tensor, τ , was defined for laminar flows in eq. (2). For turbulent flows, it is given by the combination of both viscous and turbulent stresses, the latter modeled with a Boussinesq eddy-viscosity hypothesis [43]:

$$\tau = \tau_v - \rho \overline{\mathbf{u}'\mathbf{u}'} = (\mu + \mu_t) \left[2\mathbf{S} - \frac{2}{3}\mathbf{I}\nabla \cdot \mathbf{U} \right] - \frac{2}{3}\mathbf{I}\rho k, \quad (\text{A1})$$

where μ and μ_t are the molecular and eddy viscosities, $\mathbf{S} = \frac{1}{2}[\nabla\mathbf{U} + (\nabla\mathbf{U})^T]$ is the rate-of-strain tensor, \mathbf{I} the identity tensor, and ρ the fluid density. \mathbf{u}' is the velocity fluctuations coming from the decomposition of the velocity vector, \mathbf{u} , in its average, \mathbf{U} , and fluctuation components:

$$\mathbf{u}' = \mathbf{u} - \mathbf{U}, \quad (\text{A2})$$

and k is the turbulent kinetic energy defined as $k = \frac{1}{2}\overline{\mathbf{u}' \cdot \mathbf{u}'}$.

To close the system of equations, it is necessary to estimate a value for the eddy viscosity, μ_t . To this end, among the two-equation models, the $k-\epsilon$ and $k-\omega$ could be used. These have their respective advantages, but the $k-\omega$

SST (Shear Stress Transport) turbulence model [41] connected both by the use of blending functions, F_1 and F_2 (below defined in equations (A6) and (A10)). It is well known that the $k-\epsilon$ is not accurate in near-wall regions and $k-\omega$ is inaccurate when dealing with high-pressure gradients and too sensitive to the free stream values of ω [44]. To overcome the previous challenges, the blending functions retain the $k-\omega$ model characteristics in the near-wall flow, and $k-\epsilon$ is used away from the walls. The $k-\omega$ SST is often characterized by its good behavior in adverse pressure gradients and separating flow [45]. The transport equations for k and ω are:

$$\rho \frac{Dk}{Dt} = \nabla \cdot ([\mu + \eta_k \mu_t] \nabla k) + P - \rho C_\mu \omega k, \tag{A3}$$

$$\begin{aligned} \rho \frac{D\omega}{Dt} = & \nabla \cdot ([\mu + \eta_\omega \mu_t] \nabla \omega) + \frac{\rho \gamma}{\mu_t} P - \rho \beta \omega^2 \\ & + (1 - F_1) \frac{2 \rho \eta_{\omega 2}}{\omega} \nabla k \cdot \nabla \omega, \end{aligned} \tag{A4}$$

respectively, with the mechanical turbulence production defined as

$$P = -\rho \overline{\mathbf{u}'\mathbf{u}'} : \nabla \mathbf{U} = \mu_t \left[2\mathbf{S} : \mathbf{S} - \frac{2}{3}(\nabla \cdot \mathbf{U})^2 \right] - \frac{2}{3} \rho k \nabla \cdot \mathbf{U}, \tag{A5}$$

which is limited by $\min(P, 10 \rho C_\mu \omega k)$ to prevent excessive turbulence in stagnant regions [46].

The constants γ , β and the turbulent Prandtl inverse numbers, η_k and η_ω , are blended by a function F_1 , such that $\phi = F_1 \phi_1 + (1 - F_1) \phi_2$, where ϕ is any of γ , β , η_k or η_ω .

The blending function F_1 is given by

$$F_1 = \tanh(\Gamma)^4, \tag{A6}$$

where Γ is defined as

$$\Gamma = \min \left(\max \left(\frac{\sqrt{k}}{\beta^* \omega \delta_w}, \frac{500\mu}{\rho \omega \delta_w^2} \right), \frac{4 \rho \eta_{\omega 2} k}{CD_{k\omega} \delta_w^2} \right), \tag{A7}$$

with δ_w as the distance to the nearest wall, and $CD_{k\omega}$

$$CD_{k\omega} = \max \left(\frac{2 \rho \eta_{\omega 2}}{\omega} \nabla k \cdot \nabla \omega, 10^{-10} \right), \tag{A8}$$

μ_t is defined as

$$\mu_t = \frac{\rho \alpha_1 k}{\max(\alpha_1 \omega, \sqrt{2\mathbf{S} : \mathbf{S}} F_2 F_3)} \tag{A9}$$

using two additional blending functions: F_2 and F_3 defined as

$$F_2 = \tanh \left(\left[\max \left(\frac{2 \sqrt{k}}{C_\mu \omega \delta_w}, \frac{500 \mu}{\rho \omega \delta_w^2} \right) \right]^2 \right) \tag{A10}$$

$$F_3 = 1 - \tanh \left(\left[\frac{150 \mu}{\rho \omega \delta_w^2} \right]^4 \right) \tag{A11}$$

F_3 is activated for rough-wall flows. The coefficients are based on Menter et al [44] and defined as $C_\mu = 0.09$, $\eta_{k1} = 0.85$, $\eta_{k2} = 1$, $\eta_{\omega 1} = 0.5$, $\eta_{\omega 2} = 0.856$, $\beta_1 = 0.075$, $\beta_2 = 0.0828$, $\gamma_1 = 5/9$, $\gamma_2 = 0.44$ and $\alpha_1 = 0.31$. More information about different versions of the $k-\omega$ SST turbulence model can be found at NASA [47].

Appendix B Model parameters selection

This section holds the selection process of the ML surrogate model parameters. For this purpose, the training process was employed for 1000 epochs without early stopping. The error metrics used consisted of the RMSE normalized, BIAS normalized, and STDE normalized from the ML surrogate model predictions at multiple temporal frames with the results from CFD simulations taken as the reference. The training set here consisted of bi-dimensional cylinders (N1 dataset) using the \mathbf{M}_u model. The best-behaving parameters here were selected for all the models.

The first and most impactful choice was the architecture of the model. Alongside the MLP architecture shown in the body of the article, multiple CNN architectures, represented in Fig. 12, were tested. The inspiration for the proposed CNN architectures was the U-Net [48] since it has been previously applied to fluid dynamics in the form of standard U-Net's [49, 50] or even a combination of those [51].

In Fig. 12, Models 1 (M1), 2 (M2), and 3 (M3) were schematically illustrated. Model 4 (M4) refers to the MLP architecture from Fig. 4.

To synthesize the results, in the error chart from Fig. 13 only the best performances of each architecture and block size combination are presented. The results from M1 were not shown in Fig. 13 due to its inferior performance, which would deteriorate the plot. Considering the low computational effort required by Model 4 and its overall superior accuracy (c.f. Fig. 13), it was selected.

To select the block size and overlap ratio, the influence of those was depicted in Fig. 14. From this representation, a block size of 64 seemed to work significantly worse than the other options. The best combination found was a block size of 128 and an overlap ratio of 0.75, thus these were taken as the default values.

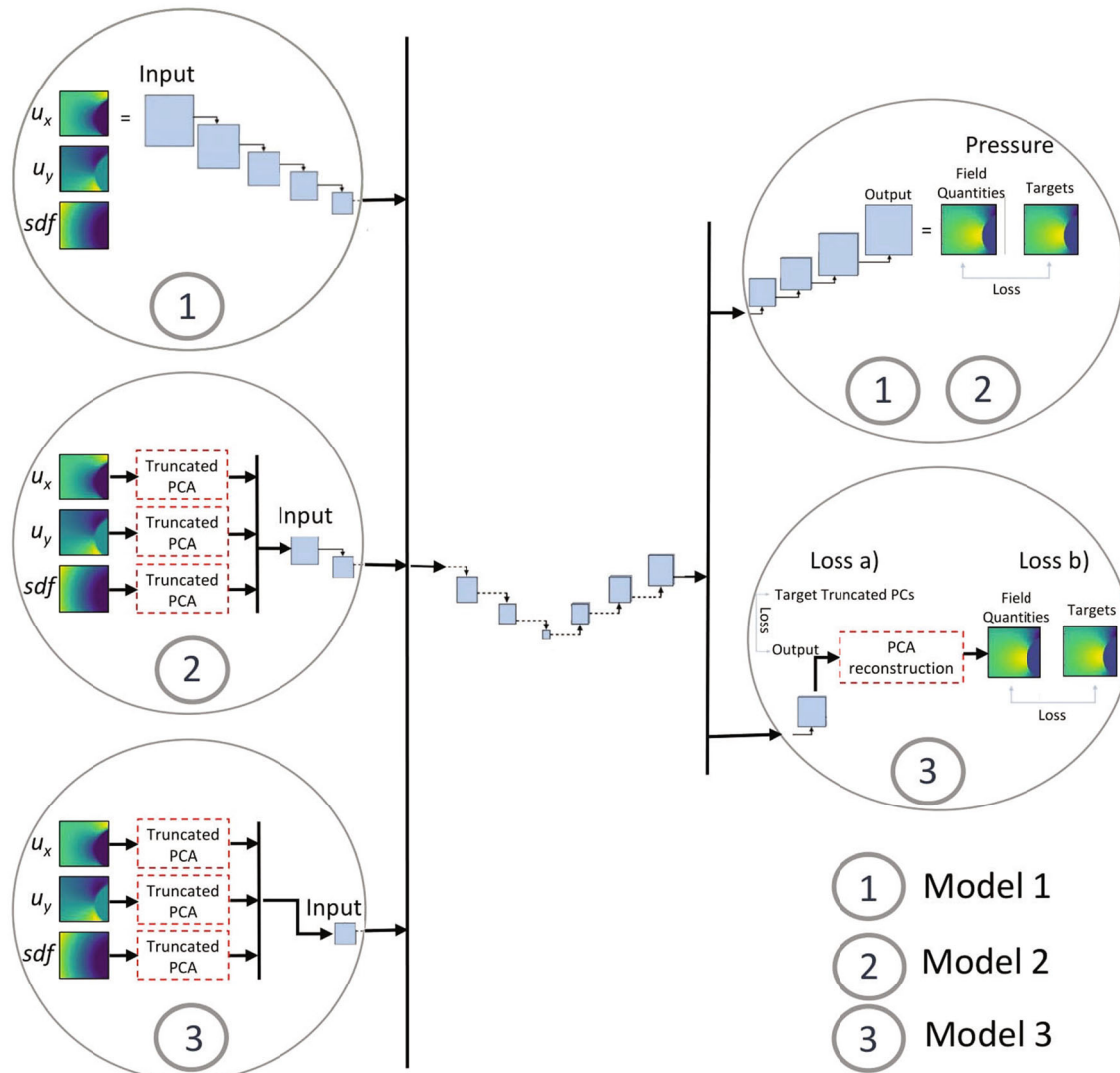


Fig. 12 U-net-based modified architectures

Selected the most relevant surrogate model parameters, it was still necessary to optimize the training hyper-parameters. These included:

- Loss formulation.
- Batch size.
- Learning rate, α , and moving average parameter, β .
 These are parameters required by the Adam optimization algorithm [52].

The most elementary loss function would be the MSE of the model prediction as defined in eq. 12, therefore it constituted L_1 . To force the NN to better predict the most important parameters, a modification was applied to the loss function to weight each component’s importance in the loss computation, being the weight defined by each PC

contribution to the total variance, L_2 and L_3 , respectively represented in equations

$$L_2 = \frac{1}{N_{points}} \sum_{i=1}^{N_{points}} (\hat{\theta} - \theta)^2 Explained_{var} \tag{B12}$$

and

$$L_3 = \frac{1}{N_{points}} \sum_{i=1}^{N_{points}} (\hat{\theta} - \theta)^2 Explained_{var}^2, \tag{B13}$$

where $Explained_{var}$ is a vector with the total variance explained by each PC. The influence of this parameter was analyzed in conjunction with the batch size as shown in Fig. 15. Since it behaved reasonably well and is conceptually simpler, L_1 was selected.

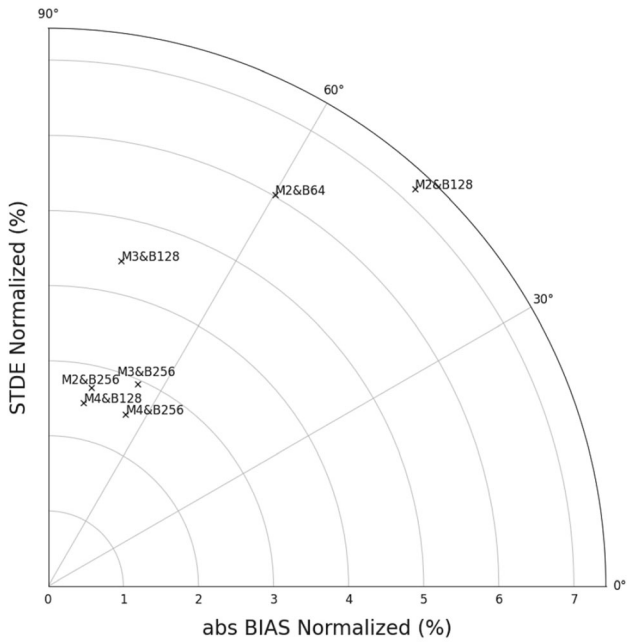


Fig. 13 Error chart for performance comparison. Mx &By represents the error for model x predictions with a block size of y

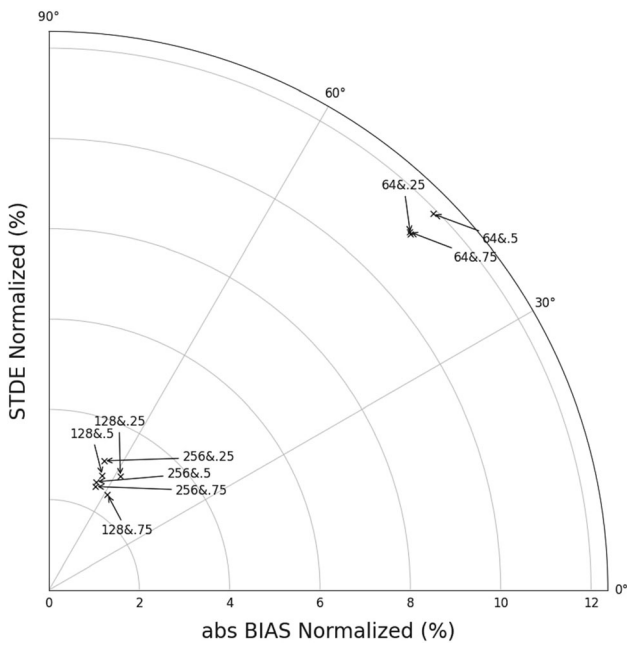


Fig. 14 Error chart of Model 4 results, where x &y represents the error for a block size of x and overlap ratio y

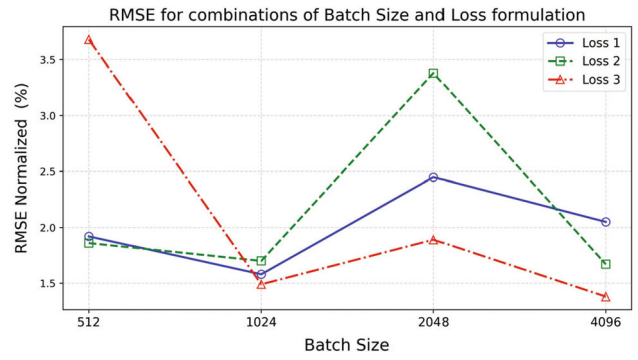


Fig. 15 Batch size and loss function selection study

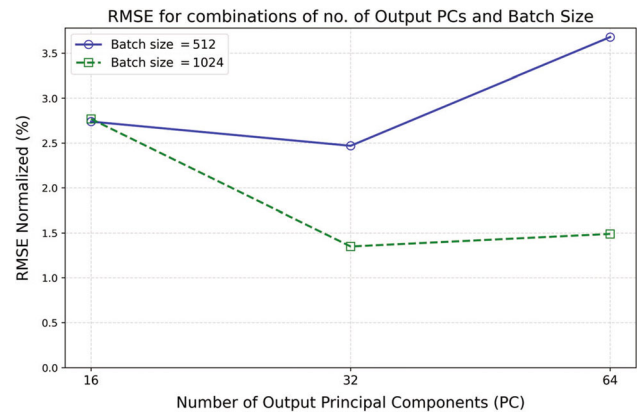


Fig. 16 The number of truncated principal components from pressure/output principal component analysis (PCA)

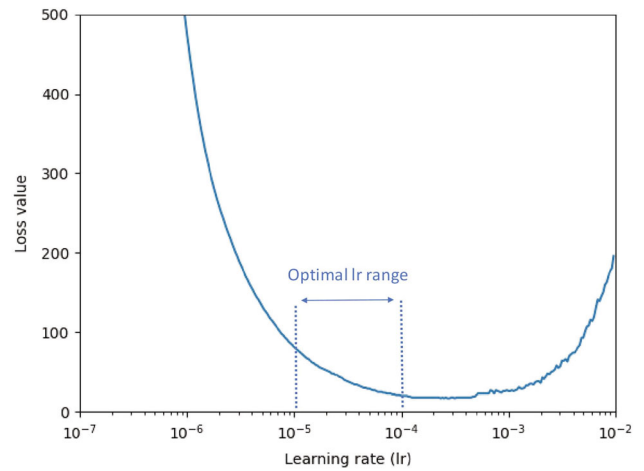


Fig. 17 Learning rate (lr) tuning

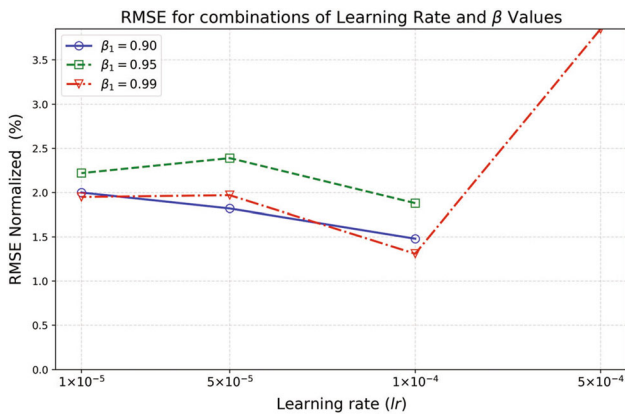


Fig. 18 Learning rate (lr) and moving average parameter (β_1) study

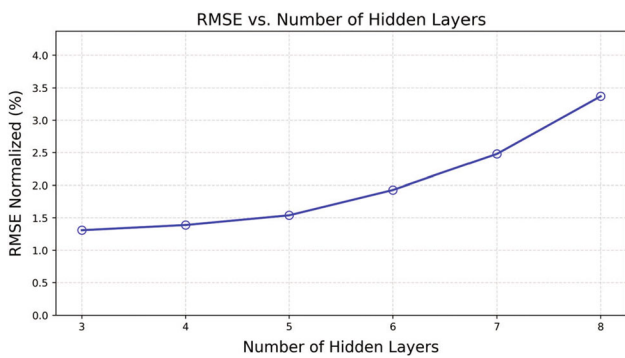


Fig. 19 Accuracy of the surrogate model at multiple MLP depths

Table 10 Description of dataset sizes before and after pre-processing for M_u datasets. The input and output size refers to the size of each input and output sample, while the No. of points refers to the total of points across all the samples

Training datasets	Input size		Output size		No. of points	
	N1	N2	N1	N2	N1	N2
Before pre-processing						
o	(128 x 128) x 3		(128 x 128)		3.28E+10	1.64E+11
□					3.28E+10	1.64E+11
<					3.28E+10	1.64E+11
/					3.28E+10	1.64E+11
o, □					6.55E+10	3.28E+11
o, □, <					9.83E+10	4.92E+11
o, □, <, /					1.31E+11	6.55E+11
After pre-processing						
o	32	45	32	48	3.20E+7	2.33E+8
□	128	128	128	128	1.28E+8	6.40E+8
<	35	35	13	13	2.40E+7	1.20E+8
/	47	46	11	11	2.90E+7	1.43E+8
o, □	32	128	32	128	6.40E+7	1.28E+9
o, □, <	70	48	128	51	2.97E+8	7.43E+8
o, □, <, /	45	46	31	31	1.52E+8	7.70E+8

The influence of the number of PCs used (or the total variance explained) by these was also studied resulting in the results presented in Fig. 16. A total of 32 principal components were selected which represented 95% of the total variance on the output dataset.

To select the most appropriate learning rate range, the evolution of the loss is plotted for a range of learning rates as in Fig. 17 allowing to first pick an optimal learning rate range to further study.

From Fig. 17 the range from 10^{-5} to 10^{-4} seemed optimal, and values in that range were tested resulting in

Table 11 Description of dataset sizes before and after pre-processing for $M_{f(u)}$ datasets. The input and output size refers to the size of each input and output sample, while the No. of points refers to the total of points across all the samples

Training datasets	Input size	Output size	No. of points
Before pre-processing			
o	(128 x 128) x 3	(128 x 128)	3.28E+10
□			3.28E+10
<			3.28E+10
/			3.28E+10
After pre-processing			
o	116	39	3.88E+8
□	512	11	1.31E+9
<	512	12	1.31E+9
/	512	11	1.31E+9

Table 12 M_u training specifications

Model	N_1			N_2		
	Epochs	$\frac{t}{epoch} [s]$	$t[h]$	Epochs	$\frac{t}{epoch} [s]$	$t[h]$
M_{\circ}	2623	31	22.59	4621	215	275.98
M_{\square}	2239	45	27.50	5000	177	245.83
M_{\triangleleft}	1915	72	65.78	3953	161	176.79
$M_{/}$	1916	21	11.18	2860	162	128.70
$M_{\circ, \square}$	3555	45	44.38	2713	351	264.52
$M_{\circ, \square, \triangleleft}$	3289	31	65.78	3046	271	230.30
$M_{\circ, \square, \triangleleft, /}$	1773	78	38.42	2279	457	289.31

Bold values indicate the best results

Table 13 $M_f(u)$ training specifications

Model	Epochs	$\frac{t}{epoch} [s]$	$t[h]$
M_{\circ}	4308	22	26.33
M_{\square}	5000	40	55.56
M_{\triangleleft}	2717	38	28.68
$M_{/}$	3437	27	25.78

Bold values indicate the best results

selecting the learning rate as $lr = 10^{-4}$ and $\beta = 0.99$ accordingly to results from Fig. 18.

The last parameter optimized consisted of the model depth, defined by the number of hidden layers with a constant width of 512 neurons as represented in Fig. 4. Since increased model complexity typically allows for a better fit of the data, namely in the BIAS of each block field prediction (note that after assembling this effect can no longer be analyzed), increasing it could be beneficial only up to a point. The optimal point was selected from the results shown in Fig. 19 and the number of hidden layers was fixed at 3.

Appendix C Datasets description and Training convergence curves

In Tables 10 and 11 the original and pre-processed datasets used for training models M_u and $M_{f(u)}$ are described. PCA reduced the dimensionality of the data by considering only the PCs necessary to represent 99.5% and 95% of the input variance for M_u and $M_{f(u)}$ respectively, and 95% of the total variance of the output.

M_u N1 - Train and Validation Losses

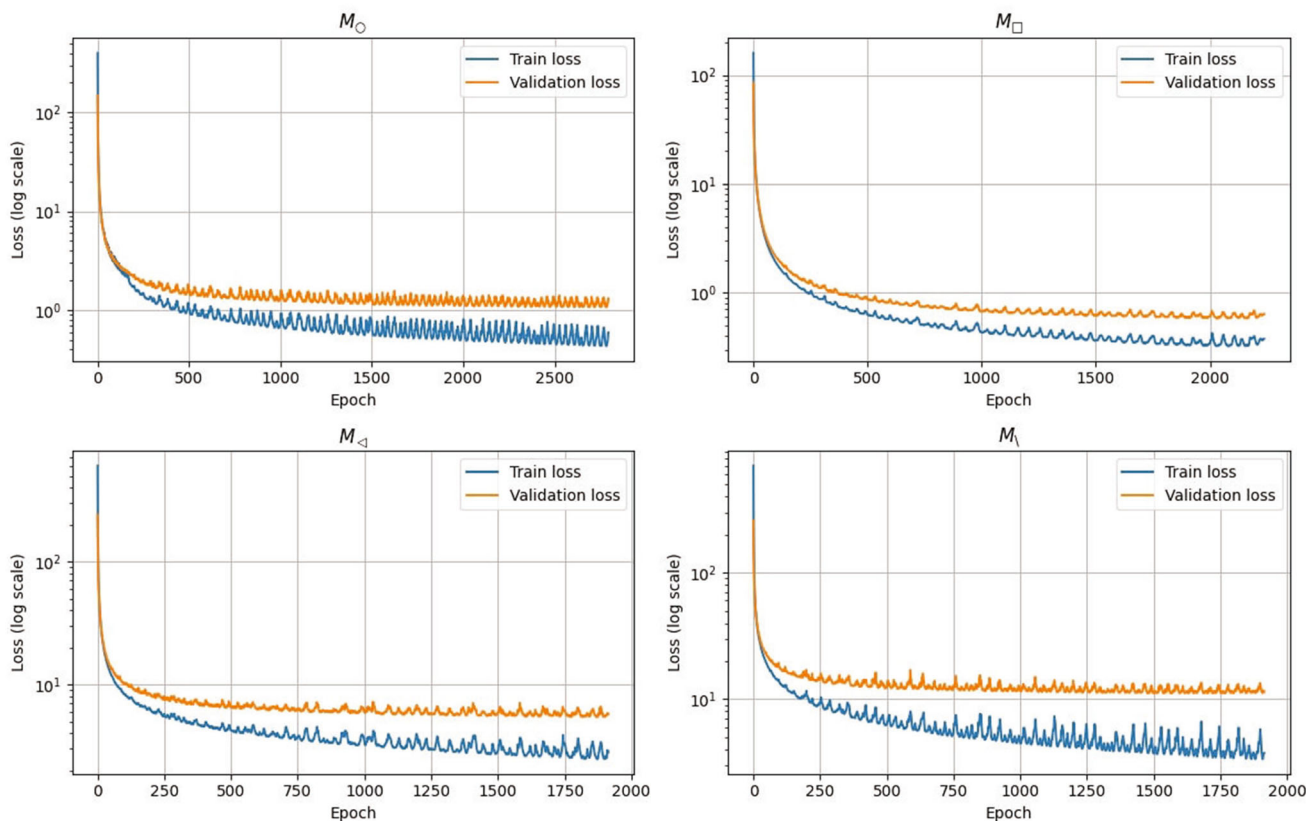


Fig. 20 Convergence curves from the training of Model M_u with N_1 datasets

M_u N2 - Train and Validation Losses

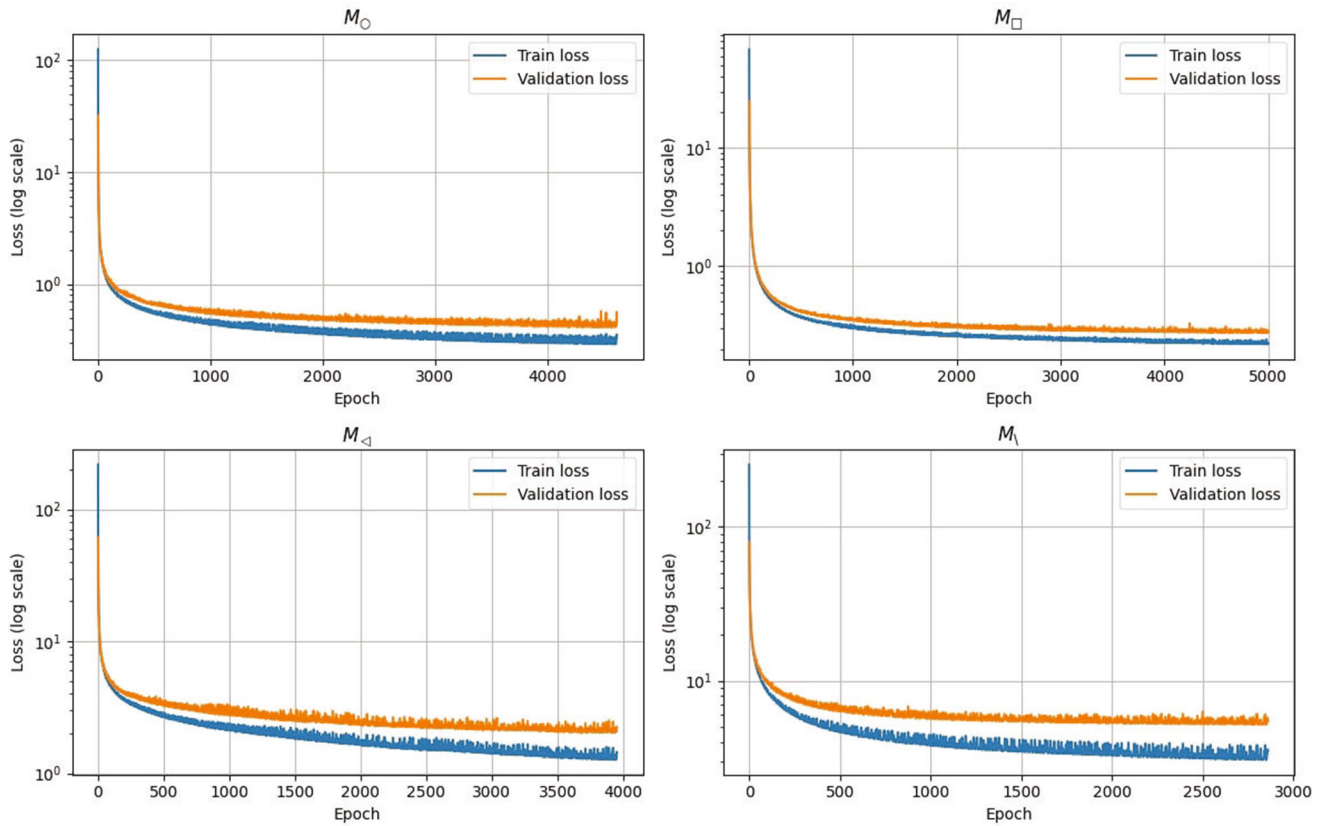


Fig. 21 Convergence curves from the training of Model M_u with N_2 datasets

After pre-processing, the data is split as 90% for training and 10% for validation.

The training of both M_u and $M_{f(u)}$ NNs was performed using 16 CPUs in parallel. The training execution times are shown in Tables 12 and 13 for M_u and $M_{f(u)}$, respectively.

The training of each NN was characterized by the convergence curves, i.e. the evolution of the training and validation losses along the epochs, from Figs. 20, 21, 22 and 23 for the NNs used in Models M_u and Fig. 24 for the NNs of Models $M_f(u)$.

Appendix D Surrogate model performance evaluation

Tables 14, 15, and 16 contain the elapsed time taken to run the surrogate model for each family of obstacle geometries at a timestamp, t , as a function of the characteristic time scale, t^* .

Tables 17, 18, and 19 contain the speedup attained by benchmarking the surrogate model against the OpenFOAM PISO implementation.

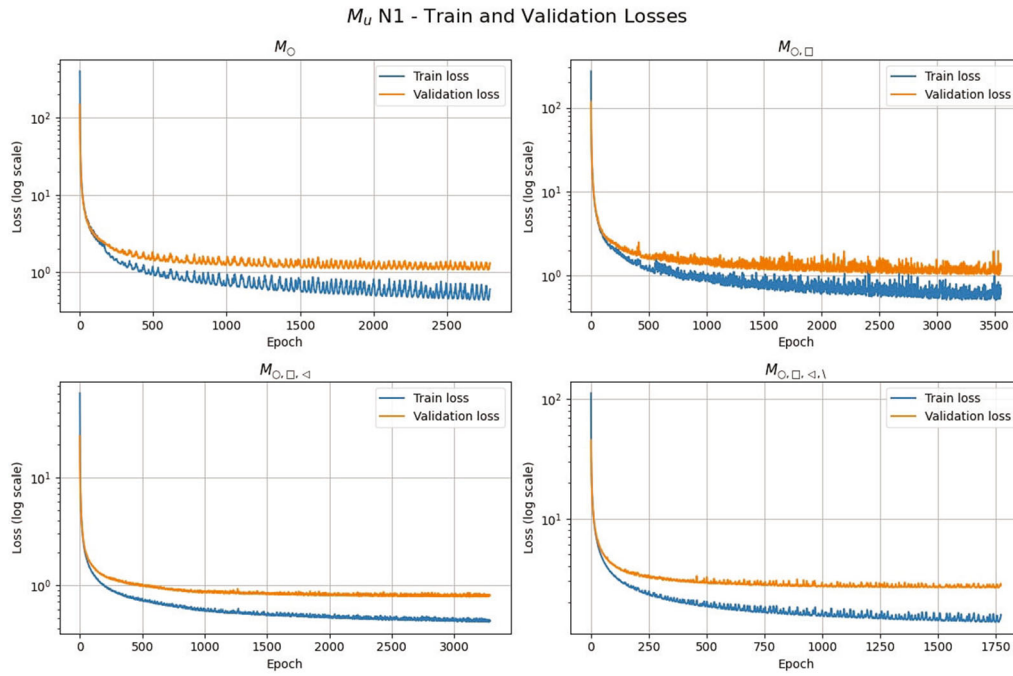


Fig. 22 Convergence curves from the training of Model M_u with N_1 datasets

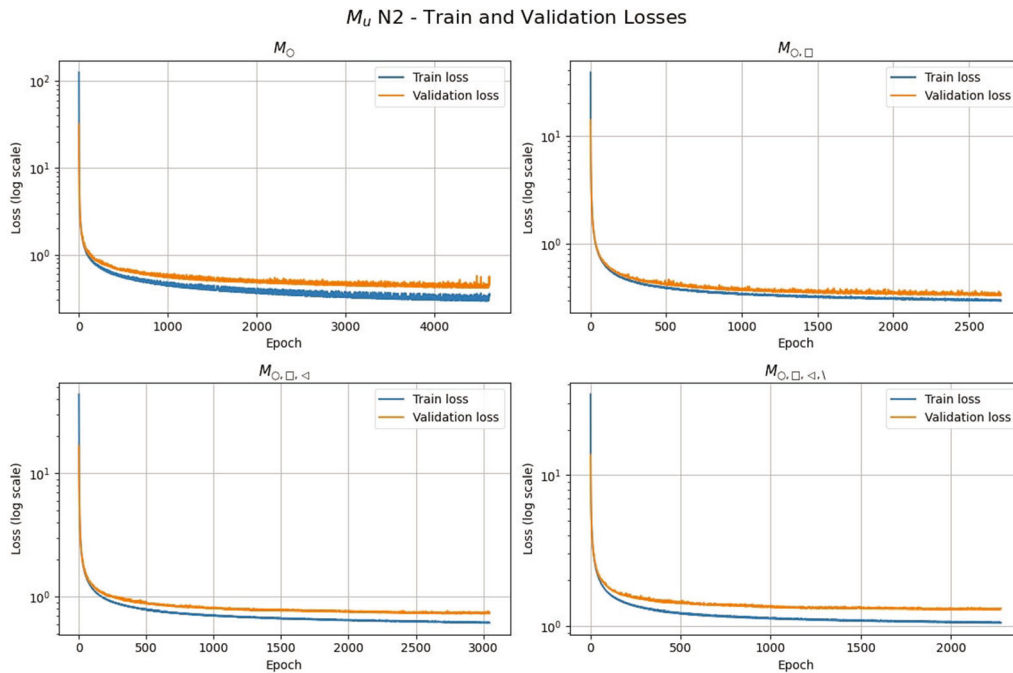


Fig. 23 Convergence curves from the training of Model M_u with N_2 datasets

$M_{\bar{r}(u)}$ - Train and Validation Losses

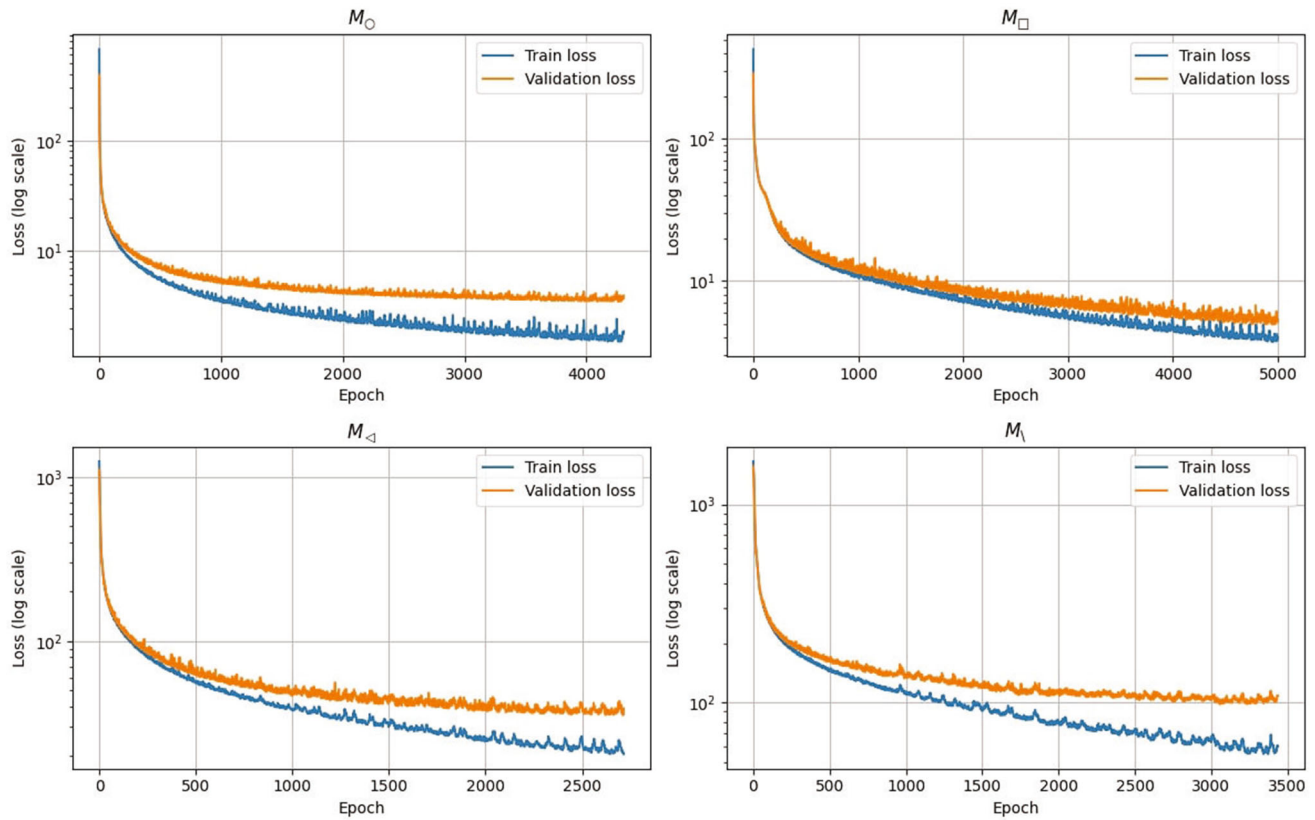


Fig. 24 Convergence curves from the training of Model $M_f(u)$

Table 14 Execution time (in [ms]) when using $M_{\circ, \square, \triangleleft, /}$ to solve the pressure Poisson Equation for multiple times defined as a function of the characteristic time, t^* . The characteristic time is defined as

$t^* = \phi / U_{\infty,c}$ where ϕ is the characteristic length and $U_{\infty,c}$ is the centerline velocity at the inlet. Overlap ratio = 0.25

Case	$t = 0$	$t = 3t^*$	$t = 6t^*$	$t = 15t^*$	$t = 45t^*$	$t = 90t^*$	$t = 150t^*$
\circ	335	334	335	334	336	333	337
\square	641	595	747	666	554	669	673
\triangleleft	426	493	635	642	713	748	782
$/$	629	887	736	886	933	820	811

Table 15 Execution time (in [ms]) when using $M_{\circ, \square, \triangleleft, /}$ to solve the pressure Poisson Equation for multiple times defined as a function of the characteristic time, t^* . The characteristic time is defined as

$t^* = \phi / U_{\infty,c}$ where ϕ is the characteristic length and $U_{\infty,c}$ is the centerline velocity at the inlet. Overlap ratio = 0.50

Case	$t = 0$	$t = 3t^*$	$t = 6t^*$	$t = 15t^*$	$t = 45t^*$	$t = 90t^*$	$t = 150t^*$
\circ	671	662	686	542	542	531	531
\square	800	874	580	517	518	519	517
\triangleleft	758	774	563	547	743	515	546
$/$	933	690	690	688	650	965	986

Table 16 Execution time (in [ms]) when using $\mathbf{M}_\circ, \square, \triangleleft, /$ to solve the pressure Poisson Equation for multiple times defined as a function of the characteristic time, t^* . The characteristic time is defined as

$t^* = \phi/U_{\infty,c}$ where ϕ is the characteristic length and $U_{\infty,c}$ is the centerline velocity at the inlet. Overlap ratio = 0.75

Case	$t = 0$	$t = 3t^*$	$t = 6t^*$	$t = 15t^*$	$t = 45t^*$	$t = 90t^*$	$t = 150t^*$
\circ	1352	1351	1350	1340	1291	1326	1305
\square	2954	2463	2448	2512	2671	2698	2667
\triangleleft	2008	2042	2385	2580	2436	2722	2596
$/$	2931	2905	3232	3265	3687	3469	3341

Table 17 Acceleration factor obtained by using $\mathbf{M}_\circ, \square, \triangleleft, /$ to solve the pressure Poisson Equation for multiple times defined as a function of the characteristic time, t^* . The characteristic time is defined as

$t^* = \phi/U_{\infty,c}$ where ϕ is the characteristic length and $U_{\infty,c}$ is the centerline velocity at the inlet. Overlap ratio = 0.25

Case	$t = 0$	$t = 3t^*$	$t = 6t^*$	$t = 15t^*$	$t = 45t^*$	$t = 90t^*$	$t = 150t^*$	Average
\circ	24.3	14.0	13.8	13.9	14.1	13.7	12.2	15.1
\square	25.8	5.9	4.8	5.1	6.8	6.0	5.5	8.6
\triangleleft	48.7	37.1	24.0	26.1	23.4	22.2	21.2	29.0
$/$	49.9	30.2	18.1	29.7	28.3	31.9	29.5	31.1

Table 18 Acceleration factor obtained by using $\mathbf{M}_\circ, \square, \triangleleft, /$ to solve the pressure Poisson Equation for multiple times defined as a function of the characteristic time, t^* . The characteristic time is defined as

$t^* = \phi/U_{\infty,c}$ where ϕ is the characteristic length and $U_{\infty,c}$ is the centerline velocity at the inlet. Overlap ratio = 0.50

Case	$t = 0$	$t = 3t^*$	$t = 6t^*$	$t = 15t^*$	$t = 45t^*$	$t = 90t^*$	$t = 150t^*$	Average
\circ	12.1	7.1	6.8	8.6	8.7	8.6	7.8	8.5
\square	20.7	4.0	6.2	6.6	7.3	7.8	7.2	8.5
\triangleleft	27.4	23.6	27.1	30.6	22.5	32.2	30.3	27.7
$/$	33.6	38.8	19.3	38.3	40.7	27.1	24.3	31.7

Table 19 Acceleration factor obtained by using $\mathbf{M}_\circ, \square, \triangleleft, /$ to solve the pressure Poisson Equation for multiple times defined as a function of the characteristic time, t^* . The characteristic time is defined as

$t^* = \phi/U_{\infty,c}$ where ϕ is the characteristic length and $U_{\infty,c}$ is the centerline velocity at the inlet. Overlap ratio = 0.7

Case	$t = 0$	$t = 3t^*$	$t = 6t^*$	$t = 15t^*$	$t = 45t^*$	$t = 90t^*$	$t = 150t^*$	Average
\circ	6.0	3.5	3.4	3.5	3.7	3.5	3.2	3.8
\square	5.6	1.4	1.5	1.3	1.4	1.5	1.4	2.0
\triangleleft	10.3	9.0	6.4	6.5	6.9	6.1	6.4	7.4
$/$	10.7	8.3	4.1	8.1	7.2	7.5	7.2	7.6

Acknowledgements We would like to acknowledge the Faculty of Engineering of the University of Porto (FEUP) and the Transport Phenomena Research Center (CEFT) for providing all the computational resources that made this study possible. A. M. Afonso acknowledges FCT - Fundação para a Ciência e a Tecnologia for

financial support through LA/P/0045/2020 (ALiCE), UIDB/00532/2020 and UIDP/00532/2020 (CEFT), funded by national funds through FCT/MCTES (PIDDAC).

Funding Open access funding provided by FCTIFCCN (b-on).

Data availability Not applicable.

Declarations

Conflict of interest Not applicable.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Frank M, Drikakis D, Charissis V (2020) Machine-learning methods for computational science and engineering. *Computation* 8(1). <https://doi.org/10.3390/computation8010015>
- Senior A, Evans R, Jumper J et al (2020) Improved protein structure prediction using potentials from deep learning. *Nature* 577:1–5. <https://doi.org/10.1038/s41586-019-1923-7>
- Vamathevan J, Clark D, Czodrowski P et al (2019) Applications of machine learning in drug discovery and development. *Nat Rev Drug Discov* 18(6):463–477. <https://doi.org/10.1038/s41573-019-0024-5>
- Dara S, Dhamecherla S, Jadav SS et al (2022) Machine learning in drug discovery: a review. *Artif Intell Rev* 55(3):1947–1999. <https://doi.org/10.1007/s10462-021-10058-4>. (epub 2021 Aug 11)
- Pyzer-Knapp EO, Pitera JW, Staar PW et al (2022) Accelerating materials discovery using artificial intelligence, high performance computing and robotics. *NPJ Comput Mater* 8:84. <https://doi.org/10.1038/s41524-022-00765-z>
- Bochenek B, Ustrnul Z (2022) Machine learning in weather prediction and climate analyses-applications and perspectives. *Atmosphere* 13(2):180. <https://doi.org/10.3390/atmos13020180>
- de Burgh-Day CO, Leeuwenburg T (2023) Machine learning for numerical weather and climate modelling: a review. *EGU sphere* 2023:1–48. <https://doi.org/10.5194/egusphere-2023-350>
- Molina MJ, O'Brien TA, Anderson G, et al (2023) A review of recent and emerging machine learning applications for climate variability and weather phenomena. *Artif Intell Earth Syst* pp 1–46. <https://doi.org/10.1175/AIES-D-22-0086.1>
- Brunton SL, Noack BR, Koumoutsakos P (2020) Machine learning for fluid mechanics. *Annual Rev Fluid Mech* 52(1):477–508. <https://doi.org/10.1146/annurev-fluid-010719-060214>
- Calzolari G, Liu W (2021) Deep learning to replace, improve, or aid CFD analysis in built environment applications: a review. *Build Environ* 206:108315. <https://doi.org/10.1016/j.buildenv.2021.108315>
- Vinuesa R, Brunton SL (2022) Enhancing computational fluid dynamics with machine learning. *Nat Comput Sci* 2(6):358–366. <https://doi.org/10.1038/s43588-022-00264-7>
- Raissi M, Perdikaris P, Karniadakis G (2019) Physics-informed neural networks: a deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J Comput Phys* 378:686–707. <https://doi.org/10.1016/j.jcp.2018.10.045>
- Beck A, Kurz M (2021) A perspective on machine learning methods in turbulence modeling. *GAMM-Mitteilungen* 44(1):e202100002. <https://doi.org/10.1002/gamm.202100002>
- Noda K, Yamaguchi Y, Nakadai K et al (2014) Audio-visual speech recognition using deep learning. *Appl Intell*. <https://doi.org/10.1007/s10489-014-0629-7>
- Kumar LA, Renuka DK, Rose SL et al (2022) Deep learning based assistive technology on audio visual speech recognition for hearing impaired. *Int J Cognit Comput Eng* 3:24–30. <https://doi.org/10.1016/j.ijcce.2022.01.003>
- Carracedo-Reboredo P, Liñares-Blanco J, Rodríguez-Fernández N, et al (2021) A review on machine learning approaches and trends in drug discovery. *Comput Struct Biotechnol J* 19:4538–4558. <https://doi.org/10.1016/j.csbj.2021.08.011>
- Wang L, Wang H, Huang Y, et al (2022) Trends in the application of deep learning networks in medical image analysis: evolution between 2012 and 2020. *European J Radiol* 146:110069. <https://doi.org/10.1016/j.ejrad.2021.110069>
- Yang J, Li S, Wang Z et al (2020) Using deep learning to detect defects in manufacturing: a comprehensive survey and current challenges. *Materials*. <https://doi.org/10.3390/ma13245755>
- Justesen N, Bontrager P, Togelius J, et al (2017) Deep learning for video game playing. *CoRR* abs/1708.07902
- He K, Zhang X, Ren S, et al (2015) Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: 2015 IEEE international conference on computer vision (ICCV), pp 1026–1034. <https://doi.org/10.1109/ICCV.2015.123>
- Dumoulin V, Visin F (2016). A guide to convolution arithmetic for deep learning. <https://doi.org/10.48550/ARXIV.1603.07285>
- Shan T, Tang W, Dang X, et al (2017) Study on a poisson's equation solver based on deep learning technique. [arXiv:1712.05559](https://arxiv.org/abs/1712.05559)
- Aggarwal R, Ugail H (2019) On the solution of poisson's equation using deep learning. In: 2019 13th international conference on software, knowledge, information management and applications (SKIMA), pp 1–8. <https://doi.org/10.1109/SKIMA47702.2019.8982518>
- Nastorg M, Bucci MA, Faney T, et al (2023) An implicit GNN solver for poisson-like problems. [arXiv:2302.10891](https://arxiv.org/abs/2302.10891)
- Özbay AG, Hamzehloo A, Laizet S et al (2021) Poisson CNN: convolutional neural networks for the solution of the Poisson equation on a cartesian mesh. *Data-Centric Eng*. <https://doi.org/10.1017/dce.2021.7>
- Illarramendi EA, Bauerheim M, Cuenot B (2021) Performance and accuracy assessments of an incompressible fluid solver coupled with a deep convolutional neural network. [arXiv:2109.09363](https://arxiv.org/abs/2109.09363)
- Weymouth GD (2022) Data-driven multi-grid solver for accelerated pressure projection. *Comput Fluids* 246:105620. <https://doi.org/10.1016/j.compfluid.2022.105620>
- Foundation TO (2022) OpenFOAM v6 User Guide. The OpenFOAM Foundation. <https://doc.cfd.direct/openfoam/user-guide-v6/>
- Moukalled F, Mangani L, Darwish M (2015) The finite volume method in computational fluid dynamics: an advanced introduction with openFOAM® and Matlab®, vol 113. <https://doi.org/10.1007/978-3-319-16874-6>
- Schlichting H, Gersten K (2017) *Boundary-Layer Theory*, 9th edn. Springer. <https://doi.org/10.1007/978-3-662-52919-5>
- Issa R (1986) Solution of the implicitly discretised fluid flow equations by operator-splitting. *J Comput Phys* 62(1):40–65. [https://doi.org/10.1016/0021-9991\(86\)90099-9](https://doi.org/10.1016/0021-9991(86)90099-9)

32. Animasaun I, Shah NA, Wakif A et al (2022) Ratio of momentum diffusivity to thermal diffusivity: introduction. *Meta-anal Scrutinizat*. <https://doi.org/10.1201/9781003217374>
33. Wang F, Animasaun I, Al-Mdallal Q et al (2023) Dynamics through three-inlets of t-shaped ducts: Significance of inlet velocity on transient air and water experiencing cold fronts subject to turbulence. *Int Commun Heat Mass Transfer* 148:107034. <https://doi.org/10.1016/j.icheatmasstransfer.2023.107034>
34. Issa R, Gosman A, Watkins AP (1986) The computation of compressible and incompressible recirculating flows by a non-iterative implicit scheme. *J Comput Phys* 62:66–82. [https://doi.org/10.1016/0021-9991\(86\)90100-2](https://doi.org/10.1016/0021-9991(86)90100-2)
35. Ferziger JH, Perić M (1999) *Computational Methods for Fluid Dynamics*, 2nd edn. Springer, Berlin
36. McKay MD, Beckman RJ, Conover WJ (1979) A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 21(2):239–245
37. Sousa PAC (2022) Solving Poisson's equation through deep learning for CFD applications. Master's thesis, Faculty of Engineering of the University of Porto. <https://hdl.handle.net/10216/140713>
38. Liang L, Liu M, Martin C et al (2018) A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis. *J R Soc Interf* 15(138):20170844. <https://doi.org/10.1098/rsif.2017.0844>
39. Versteeg H, Malalasekera W (2007) *An introduction to computational fluid dynamics: the finite volume method*. Pearson Educ Limit. <https://books.google.pt/books?id=RvBZ-UMpGzIC>
40. OpenCFD (2022) Openfoam user guide: k-omega shear stress transport (sst). <https://www.openfoam.com/documentation/guides/latest/doc/guide-turbulence-ras-k-omega-sst.html>, Accessed: 2022-08-20
41. Menter FR (1994) Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA J* 32(8):1598–1605. <https://doi.org/10.2514/3.12149>
42. Pope SB (2000) *Turbulent Flows*. Cambridge Univ Press. <https://doi.org/10.1017/CBO9780511840531>
43. Schmitt FG (2007) About Boussinesq's turbulent viscosity hypothesis: historical remarks and a direct evaluation of its validity. *Comptes Rendus Mécanique* 335(9–10):617–627. <https://doi.org/10.1016/j.crme.2007.08.004>
44. Menter FR, Kuntz M, Langtry R (2003) Ten years of industrial experience with the SST turbulence model
45. Apsley D, Leschziner M (2012) Advanced turbulence modelling of separated flow in a diffuser. *Flow Turbulence Combust* 63:81–112. <https://doi.org/10.1023/A:1009930107544>
46. Menter FR (1993) Zonal two equation k-w turbulence models for aerodynamic flows
47. NASA LRC (2022) The menter shear stress transport turbulence model. <https://turbmodels.larc.nasa.gov/sst.html>, Accessed: 2022-08-20
48. Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. *CoRR abs/1505.04597*. <http://arxiv.org/abs/1505.04597>,
49. Le QT, Ooi C (2021) Surrogate modeling of fluid dynamics with a multigrid inspired neural network architecture. *Mach Learn Appl* 6:100176. <https://doi.org/10.1016/j.mlwa.2021.100176>
50. Thuerey N, Weißenow K, Prantl L et al (2020) Deep learning methods for Reynolds-averaged Navier-stokes simulations of airfoil flows. *AIAA J* 58(1):25–36. <https://doi.org/10.2514/1.J058291>
51. Chen J, Viquerat J, Hachem E (2019) U-net architectures for fast prediction in fluid mechanics. <https://hal.archives-ouvertes.fr/hal-02401465>, working paper or preprint
52. Reddi SJ, Kale S, Kumar S (2019) On the convergence of adam and beyond. *CoRR abs/1904.09237*. <http://arxiv.org/abs/1904.09237>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.