Data-Centric Engineering (2025), 6: e48
doi:10.1017/dce.2025.10022

CAMBRIDGE UNIVERSITY PRESS

RESEARCH ARTICLE

Mean flow data assimilation using physics-constrained graph neural networks

Michele Quattromini[1,2], Michele Alessandro Bucci[3], Stefania Cherubini[1] and Onofrio Semeraro[2]

[1]Dipartimento di Meccanica, Matematica e Management, Politecnico di Bari, Bari, Italy
[2]LISN-CNRS, Université Paris-Saclay, Orsay, France
[3]Digital Sciences and Technologies Department, SafranTech, Magny-Les-Hameaux, France
Corresponding author: Michele Quattromini; Email: michele.quattromini@poliba.it

Received: 08 January 2025; **Revised:** 02 June 2025; **Accepted:** 16 July 2025

Keywords: computational fluid dynamics; data assimilation; machine learning; physics-constrained graph neural networks; surrogate modeling

Abstract

Despite their widespread use, purely data-driven methods often suffer from overfitting, lack of physical consistency, and high data dependency, particularly when physical constraints are not incorporated. This study introduces a novel data assimilation approach that integrates Graph Neural Networks (GNNs) with optimization techniques to enhance the accuracy of mean flow reconstruction, using Reynolds-averaged Navier–Stokes (RANS) equations as a baseline. The method leverages the adjoint approach, incorporating RANS-derived gradients as optimization terms during GNN training, ensuring that the learned model adheres to physical laws and maintains consistency. Additionally, the GNN framework is well-suited for handling unstructured data, which is common in the complex geometries encountered in computational fluid dynamics. The GNN is interfaced with the finite element method for numerical simulations, enabling accurate modeling in unstructured domains. We consider the reconstruction of mean flow past bluff bodies at low Reynolds numbers as a test case, addressing tasks such as sparse data recovery, denoising, and inpainting of missing flow data. The key strengths of the approach lie in its integration of physical constraints into the GNN training process, leading to accurate predictions with limited data, making it particularly valuable when data are scarce or corrupted. Results demonstrate significant improvements in the accuracy of mean flow reconstructions, even with limited training data, compared to analogous purely data-driven models.

Impact Statement

This study presents an innovative approach that integrates graph neural networks with physics-based constraints derived from Reynolds-averaged Navier–Stokes equations to improve fluid flow reconstruction. By incorporating physical laws into the training process through adjoint optimization, the framework enhances prediction accuracy and consistency. The present work demonstrates that this data assimilation approach can effectively reconstruct flow fields from sparse, noisy, or incomplete data, and is applicable to complex geometries, offering a promising solution for addressing challenges in modeling fluid behavior under a variety of conditions.

This research article was awarded Open Data and Open Materials badges for transparent practices. See the Data Availability Statement for details.

© The Author(s), 2025. Published by Cambridge University Press. This is an Open Access article, distributed under the terms of the Creative Commons Attribution licence (http://creativecommons.org/licenses/by/4.0), which permits unrestricted re-use, distribution and reproduction, provided the original article is properly cited.

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

1. Introduction

In recent years, the integration of machine learning (ML) algorithms into computational fluid dynamics (CFDs) has experienced significant growth, driven by the increasing efficiency of ML models in processing large datasets and their impressive capabilities in inference and prediction. The literature is already rich with various effective approaches for combining ML algorithms with CFDs, as highlighted in the review works by Brunton et al. (2020) and Vinuesa and Brunton (2022). These applications have found a natural domain in turbulence modeling problems, ranging from the closure of Reynolds-averaged Navier–Stokes (RANS) equations to wall-modeled large eddy simulations (Lapeyre, 2019; Zhou et al., 2021; Bae and Koumoutsakos, 2022). Among the many research efforts addressing these challenges, Ling and Templeton (2015) applied classification methods to identify regions of uncertainty where the closure term in RANS might fail, while other approaches leverage baseline models such as the Spalart–Allmaras closure (Singh and Duraisamy, 2016), physics-informed neural networks (NNs) (Eivazi et al., 2022), regression methods (Schmelzer et al., 2020), decision trees (Duraisamy et al., 2019), ensemble methods (McConkey et al., 2022), tensor-basis neural networks (Ling et al., 2016; Cai et al., 2024), and genetic programming (Weatheritt and Sandberg, 2016; Zhao et al., 2020). For a broader overview, we refer to Beck and Kurz (2021), Cinnella (2024), and Duraisamy et al. (2019), where thorough reviews of ML techniques applied to the subject are provided.

On the other hand, alternative approaches are based on data assimilation. In this framework, diverse observations are integrated with the governing dynamical laws of a system to optimally estimate physical variables. Starting from a background state and incorporating imperfect observational data, it yields the best possible estimate of the system's true state by accounting for the statistical reliability of each data source. These methodologies are routinely employed in meteorology, where accurate weather forecasting is achieved by combining satellite imagery, meteorological measurements, and numerical models (Navon, 2009), (Titaud et al., 2010). Two main approaches to data assimilation can generally be identified: (i) stochastic estimation, which relies on probabilistic methods such as Kalman filtering, and (ii) variational data assimilation. In the context of fluid mechanics, recent studies employing stochastic estimation include Meldi and Poux (2017) and Cheng et al. (2019). Variational assimilation, in particular the four-dimensional variational assimilation methods, has been applied in Cordier et al. (2013) for turbulence model identification, and in a similar spirit in Dimitry et al. (2014). The success of these applications in fluid mechanics is not surprising, as variational methods based on adjoint formulations are commonly applied across a broad range of problems. These include sensitivity analysis in stability studies (Giannetti and Luchini, 2007; Marquet et al., 2008; Luchini et al., 2009), optimization (Schmid and Henningson, 1994; Cherubini et al., 2010; Magri, 2019; Giannotta et al., 2022), receptivity analysis (Giannetti and Luchini, 2006), and flow control (Cherubini et al., 2013; Semeraro et al., 2013). A thorough overview of these applications is provided by Luchini and Bottaro (Luchini and Bottaro, 2014). Returning to the topic of mean flow reconstruction, Dimitry et al. (2014) proposed a variational data assimilation method for reconstructing the mean flow field from partial measurements using the forced RANS equations. Their approach minimizes the discrepancy between experimental data and numerical solutions by identifying an optimal forcing term that effectively represents the unknown Reynolds stresses. This is achieved through a direct-adjoint looping procedure, and the closure model is interpreted as an optimal control variable informed by the adjoint field.

Building on this, recent efforts combining data assimilation with NN modeling have been proposed in Ströfer and Xiao (2021), Volpiani et al. (2021), and Patel et al. (2024). Following this philosophy, in this work, we propose a novel approach that integrates graph NNs (GNNs) with RANS equations to enhance the training process. The resulting method is referred to as the physics-constrained GNN (PhyCo-GNN). Specifically, the forcing term in the RANS equations is modeled using a GNN, where the network is informed by gradients obtained via autodifferentiation and analytical gradients derived from the adjoint equations during the iterative process. By embedding the RANS closure term into the optimization framework via the adjoint method, we leverage adjoint methods to efficiently compute gradients necessary for optimization. This ensures that the gradients used during the GNN training are informed

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

by a deterministic physical model, rather than relying solely on available data, thus ensuring physical consistency. A key feature of this approach is the use of GNNs (Hamilton, 2020), which are characterised by complex, multi-connected nodes that can be naturally adapted to unstructured meshes. In a GNN, convolution is performed by aggregating information from neighboring nodes, overcoming the limitations of geometry typically encountered in convolutional NNs. Additionally, GNNs exhibit superior generalization capabilities compared to standard models (Sanchez-Gonzalez et al., 2018), are differentiable, and enable direct learning of operators through discrete stencils (Shukla et al., 2022). Due to these advantages, GNNs have recently gained attention in fluid mechanics. A comprehensive review by Lino et al. (2023) and examples from Toshev et al. (2023) and Dupuy et al. (2023), where GNNs are used to model wall shear stress in Large Eddy Simulations (LES) simulations, highlight their potential. In Quattromini et al. (2025), a GNN is used to predict the closure term in RANS equations at low Reynolds numbers, using a supervised learning strategy. In that setting, the NN model receives the mean flow field as input and is trained to regress the corresponding forcing term, with both quantities derived from high-fidelity DNS data. The approach is designed to map the mean flow to the forcing term in a purely data-driven setting. In contrast, the present work incorporates this GNN architecture as a pre-trained module within a broader physics-constrained optimization framework. The GNN model is here repurposed to act as a prior for a data assimilation scheme governed by the RANS equations. The key novelty lies in the integration of the GNN into an adjoint-based inverse problem, where the NN model is trained not only to replicate the direct numerical simulation (DNS)-derived forcing term, but also to minimize discrepancies in the mean flow field reconstruction through gradient-based optimization, informed by the governing physics. The remainder of this article is structured as follows. The mathematical framework is described in Section 2, where the physical baseline model for the numerical simulations is detailed in Section 2.1, and the adjoint optimization method is presented in Section 2.2. Section 4 outlines the ML framework, detailing the GNN architecture (Section 4.1), the dataset preprocessing (Section 4.2), and the training algorithm (Section 4.3). We then proceed to present the PhyCo-GNN approach in Section 5. Results are presented in Section 6, where mean flow reconstruction is shown for several test cases, including sparse data recovery, denoising, and inpainting of missing flow data. Conclusions are drawn in Section 7.

2. Mean flow reconstruction using data assimilation

2.1. Governing equations

This study focuses on two-dimensional (2D) incompressible fluid flows around bluff bodies in unsteady regimes. We begin with the Navier–Stokes (NS) equations for incompressible flows. Let $\mathbf{x} = (x, y)$ denote the spatial Cartesian coordinates. The velocity field $\mathbf{u}(\mathbf{x}, t)$ and pressure field $p(\mathbf{x}, t)$ follow these dynamics:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla p + \frac{1}{Re}\nabla^2 \mathbf{u} \tag{2.1a}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{2.1b}$$

The above equations are made dimensionless by introducing the characteristic length scale D (e.g., the diameter of the circumscribed bluff body circumference), the velocity U_∞ of the incoming flow, and ρU_∞^2 as the reference pressure. The Reynolds number is defined as $Re = U_\infty \frac{D}{\nu}$, where ν represents the kinematic viscosity. This number represents the ratio between the inertial forces and the viscous forces in the fluid flow and is used to characterize the flow regime, indicating whether the flow is laminar or turbulent, depending on the specific case. In this study, we focus on unsteady nonlinear flows that settle on limit cycles or quasi-periodic regimes, a configuration typically occurring when supercritical flows have not yet become turbulent.

The NS equations can be computationally intensive. Hence, various approximate models are used based on the required accuracy. In this study, we adopt the RANS model, a time-averaged formulation of the NS equations. By introducing the Reynolds decomposition:

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

$$\mathbf{u}(\mathbf{x},t) = \overline{\mathbf{u}}(\mathbf{x}) + \mathbf{u}'(\mathbf{x},t), \tag{2.2}$$

the unsteady velocity field $\mathbf{u} = (u,v)^T$ is split into an ensemble-averaged velocity field $\overline{\mathbf{u}} = (\overline{u},\overline{v})^T$ and a fluctuating component $\mathbf{u}' = (u',v')^T$. Formally, this decomposition allows any unsteady flow to be expressed as a sum of a steady mean flow and an unsteady fluctuating part. Plugging the Reynolds decomposition (Eq. 2.2) into the NS equations (Eq. 2.1) and ensemble-averaging results in:

$$(\overline{\mathbf{u}} \cdot \nabla)\overline{\mathbf{u}} = -\nabla \overline{p} + \frac{1}{Re}\nabla^2 \overline{\mathbf{u}} + \mathbf{f} \tag{2.3a}$$

$$\nabla \cdot \overline{\mathbf{u}} = 0, \tag{2.3b}$$

where $\overline{p}$ is the mean pressure field. These equations are completed with a set of boundary conditions, detailed in Section 3, together with a description of the numerical setup.

In this context, the Reynolds stress tensor $\mathbf{f}$ acts as a closure term for the underdetermined system of nonlinear equations in Eq. 2.3. Ideally, $\mathbf{f}$ can be directly computed, when data are available, as:

$$\mathbf{f} = -\nabla \cdot \left(\overline{\mathbf{u}'\mathbf{u}'}\right). \tag{2.4}$$

In practice, mathematically computing $\mathbf{f}$ requires either DNS or sufficiently long sampling of experimental or numerical data to accurately capture the statistical properties of the fluctuating components $\mathbf{u}'$. Unlike the mean flow $\overline{\mathbf{u}}$, the fluctuating components are inherently statistical in nature, and their proper representation depends on achieving statistical convergence rather than full time-resolved data. This is known as the closure problem. Several approximations, such as the Boussinesq hypothesis—for example, k-ϵ, k-ω, and Spalart–Allmaras models (Wilcox, 1998)—or more complex models, such as the explicit algebraic Reynolds stress model (Wallin and Johansson, 2000) and differential Reynolds stress models (Cécora et al., 2015), can be introduced to address this problem. Nevertheless, in this work, we do not assume any modeling approximation of the forcing term $\mathbf{f}$, but instead aim to compute it by combining neural-network modeling and data assimilation; a GNN model is trained to infer the Reynolds stress term $\mathbf{f}$ from a given mean flow $\overline{\mathbf{u}}$ introduced as input. The Reynolds stress term $\mathbf{f}$ is the output of the GNN and is assumed in this context as a control variable in the adjoint optimization process (see Section 5).

2.2. Adjoint-based data assimilation

Among the various strategies available in the literature, data assimilation schemes can be formulated by casting optimization loops based on the adjoint method. In this approach, a cost function is defined to be either maximized or minimized, alongside a control variable to be adjusted accordingly. To this end, we introduce a mapping that projects the mean flow field $\overline{\mathbf{u}}$ onto the measurement vector $\overline{\mathbf{m}}$ as follows

$$\overline{\mathbf{m}} = \mathcal{M}(\overline{\mathbf{u}}).$$

The operator $\mathcal{M}$ defines the ground truth data used in the optimization process. If $\mathcal{M} = I$, we consider the full-field $\overline{\mathbf{u}}$; otherwise, $\mathcal{M}$ may be designed to project the field onto a subset of measurements (e.g., sparse observations or an incomplete flow field, as in the inpainting scenario; see Section 6). Based on this definition, the cost function to be minimized is given by the discrepancy between the ground truth $\overline{\mathbf{m}}$ and the reconstructed measurements $\mathcal{M}(\widehat{\overline{\mathbf{u}}})$

$$\mathcal{J}(\widehat{\overline{\mathbf{u}}}) = \frac{1}{2}\left\|\overline{\mathbf{m}} - \mathcal{M}(\widehat{\overline{\mathbf{u}}})\right\|_2^2, \tag{2.5}$$

where $\|\cdot\|_2$ represents the L^2-norm, defined in association with the scalar product over the domain $\mathcal{M}$:

$$\langle \mathbf{a}, \mathbf{b} \rangle = \int_\Omega \mathbf{a} \cdot \mathbf{b} d\Omega. \tag{2.6}$$

We consider the RANS equations (Eq. 2.3) as the baseline equation and choose the unknown forcing stress term $\mathbf{f}$ as the control variable. This procedure is inspired by the study in Dimitry et al. (2014), where an optimization loop is designed to reconstruct the mean flow $\overline{\mathbf{u}}$.

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

Since the control variable is the forcing stress term $\mathbf{f}$, the cost function $\mathcal{J}$ (Eq. 2.5) does not explicitly depend on it. Therefore, to relate the cost function $\mathcal{J}$ to the forcing stress tensor $\mathbf{f}$, we need to define an augmented Lagrangian functional $\mathcal{L}$:

$$\mathcal{L}\left(\mathbf{f},\widehat{\mathbf{u}},\widehat{p},\widehat{\mathbf{u}}^{\dagger},\widehat{p}^{\dagger}\right)=\mathcal{J}\left(\widehat{\mathbf{u}}\right)-\langle\widehat{\mathbf{u}}^{\dagger},\widehat{\mathbf{u}}\cdot\nabla\widehat{\mathbf{u}}+\nabla\widehat{p}-\frac{1}{Re}\nabla^{2}\widehat{\mathbf{u}}-\mathbf{f}\rangle-\langle\widehat{\mathbf{u}}^{\dagger},\nabla\cdot\widehat{\mathbf{u}}\rangle \tag{2.7}$$

where $\langle\cdot,\cdot\rangle$ represents the spatial scalar product, as defined in Eq. 2.6. The augmented Lagrangian functional $\mathcal{L}$ allows us to represent the constrained optimization problem as an unconstrained one by introducing two a priori unknown variables, the Lagrangian multipliers $\widehat{\mathbf{u}}^{\dagger}$ and $\widehat{p}^{\dagger}$. To minimize the functional defined in Eq. 2.7, its partial derivatives with respect to all independent variables of the problem must be set to zero.

Following this approach, nullifying the partial derivatives of Eq. 2.7 with respect to the direct variables $\overline{\mathbf{u}}$ and $\overline{p}$ yields the adjoint NS equations:

$$-\widehat{\mathbf{u}}\cdot\nabla\widehat{\mathbf{u}}^{\dagger}+\widehat{\mathbf{u}}^{\dagger}\cdot\nabla\widehat{\mathbf{u}}^{T}-\nabla\widehat{p}^{\dagger}-\frac{1}{Re}\nabla^{2}\widehat{\mathbf{u}}^{\dagger}=\frac{\partial\mathcal{J}}{\partial\widehat{\mathbf{u}}} \tag{2.8a}$$

$$\nabla\cdot\widehat{\mathbf{u}}^{\dagger}=0 \tag{2.8b}$$

along with an appropriate set of boundary conditions, detailed in Section 3. It is worth noting that the adjoint NS equations (Eq. 2.8) are forced on the right-hand side by the partial derivative of the error function $\mathcal{J}$ with respect to the reconstructed mean flow $\overline{\mathbf{u}}$. This latter can be easily derived from Eq. 2.5 as

$$\frac{\partial\mathcal{J}}{\partial\widehat{\mathbf{u}}}=-\frac{\partial\mathcal{M}}{\partial\widehat{\mathbf{u}}}\left(\overline{\mathbf{m}}-\mathcal{M}\left(\widehat{\mathbf{u}}\right)\right). \tag{2.9}$$

Finally, the partial derivative of Eq. 2.7 with respect to the forcing term $\mathbf{f}$ yields:

$$\frac{\partial\mathcal{J}}{\partial\mathbf{f}}=\widehat{\mathbf{u}}^{\dagger}. \tag{2.10}$$

More details can be found in Dimitry et al. (2014), where the case of localized measurements is also included in the analytical description of the direct-adjoint loop.

By exploiting the gradients of the cost function $\mathcal{J}$ with respect to the control variable $\mathbf{f}$ (Eq. 2.10), a gradient descent algorithm can be applied to optimize $\mathbf{f}$ and iteratively converge toward the optimal solution that minimizes the cost function $\mathcal{J}$ (Eq. 2.5). Specifically, the iterative direct-adjoint process refines the forcing term $\mathbf{f}$, ensuring that the RANS model accurately captures the mean flow characteristics observed in high-fidelity DNS data. To summarize the process algorithmically, the adjoint optimization procedure involves the following steps:

1. **Initialization:** An initial guess for the control variable $\mathbf{f}$ is chosen. We start from $\mathbf{f}=0$ to ensure the divergence-free property ($\nabla\cdot\mathbf{f}=\mathbf{0}$) and consistency with the adopted boundary conditions (Dimitry et al., 2014).
2. **Forward step:** A prediction of the mean flow $\widehat{\mathbf{u}}$ is obtained given the forcing term $\mathbf{f}$, by solving Eq. 2.3.
3. **Cost function evaluation:** The distance between the reconstructed mean flow $\widehat{\mathbf{u}}$ and the ground truth mean flow $\overline{\mathbf{u}}$ is assessed using the cost function $\mathcal{J}$ (Eq. 2.5).
4. **Adjoint step:** The adjoint equations (Eq. 2.8), forced by the term in Eq. 2.9, are solved to find $\widehat{\mathbf{u}}^{\dagger}$. The term in Eq. 2.10 expresses the variation of the cost function $\mathcal{J}$ with respect to the control variable $\mathbf{f}$.
5. **Control variable update:** using this gradient, corresponding to the adjoint variable $\widehat{\mathbf{u}}^{\dagger}$, the forcing term $\mathbf{f}$ is adjusted as:

$$\mathbf{f}^{(n+1)}=\mathbf{f}^{(n)}+\beta\frac{\partial\mathcal{J}^{(n)}}{\partial\mathbf{f}^{(n)}}=\mathbf{f}^{(n)}+\beta\widehat{\mathbf{u}}^{\dagger(n)}, \tag{2.11}$$

where the superscript $^{(n)}$ indicates the n-th iteration of the optimization loop and β a coefficient related to the step along the gradient direction. It is noteworthy that the variable $\widehat{\mathbf{u}}^{\dagger}$ is physically divergence-free, thus justifying the initialization of $\mathbf{f}$ as a null field.

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

The direct-adjoint process is iteratively repeated until the cost function $\mathcal{J}$ (Eq. 2.5) drops below a predefined threshold, based on the required accuracy.

3. Numerical setup

The unsteady wake behind a bluff body is a well-established benchmark in CFDs. As a reference case, the cylinder bluff body exhibits stable behavior up to a critical Reynolds number, $Re_c \cong 46.7$ (Provansal et al., 1987; Giannetti and Luchini, 2007). Beyond this threshold, irregular velocity fluctuations appear, accompanied by periodic vortex formation (Roshko, 1954), and the unsteady flow evolves into a limit cycle known as the von Karman street. This phenomenon is observable up to $Re = 150$ for 2D cases (Roshko, 1954). At these Re, dynamics can also settle in quasi-periodic regimes or show chaotic behavior —for example, the case of fluidic pinball (Deng et al., 2020). This study focuses on 2D scenarios exhibiting periodic or quasi-periodic behavior within the range $50 \leq Re \leq 150$. In our numerical setup, the characteristic dimension is the diameter D of the circumscribed circle around the bluff body. Based on this dimension, the computational domain extends $L_x = 27$ units in the streamwise direction and $L_y = 10$ units in the transverse direction. The system's origin $O(0,0)$ is positioned $\Delta x = 9$ units downstream from the inlet and $\Delta y = 5$ units from the symmetry boundaries. A pictorial sketch of the geometric configuration of the computational domain is shown in Figure 1.

The flow evolves from left to right with a dimensionless uniform velocity $\mathbf{u} = (1,0)^T$, normalized by the reference velocity U_∞ of the undisturbed flow. Boundary conditions follow the setup described by Dimitry et al. (2014). For the direct NS equations (Eq. 2.1), they are as follows:

$$\begin{cases} u = 1, \quad v = 0 \text{ at the inlet,} \\ u = 0, \quad v = 0 \text{ on the cylinder surface,} \\ \partial_y u = 0, \quad v = 0 \text{ on symmetry boundaries,} \\ \dfrac{1}{Re}\partial_x u - p = 0, \quad \partial_x v = 0 \text{ at the outlet.} \end{cases} \tag{3.1}$$

For the adjoint NS equations (Eq. 2.8), the boundary conditions are:

$$\begin{cases} u^\dagger = 1, \quad v^\dagger = 0 \text{ at the inlet,} \\ u^\dagger = 0, \quad v^\dagger = 0 \text{ on the cylinder surface,} \\ \partial_y u^\dagger = 0, \quad v^\dagger = 0 \text{ on symmetry boundaries,} \\ \dfrac{1}{Re}\partial_x u^\dagger + p^\dagger = -uu^\dagger, \quad \dfrac{1}{Re}\partial_x v^\dagger = -uv^\dagger \text{ at the outlet.} \end{cases} \tag{3.2}$$

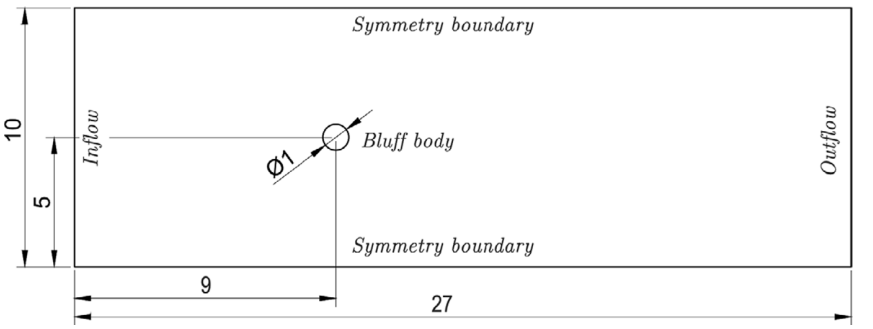


Figure 1. *Sketch of the computational domain geometry, where the diameter of the circumscribed circle around the bluff body, along with the height and length of the domain, are provided in nondimensional units.*

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

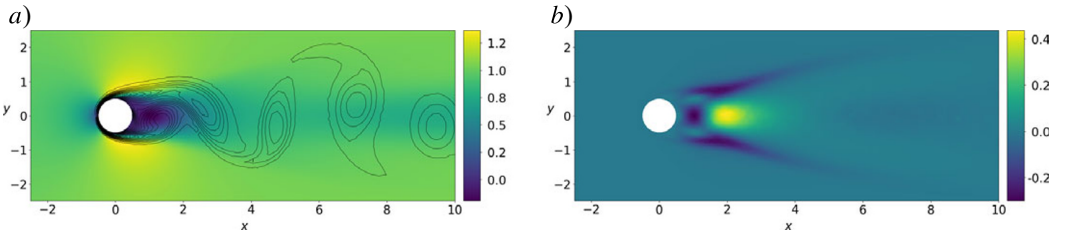


Figure 2. *(a) Streamwise component of the mean flow,* $\overline{\mathbf{u}}$*, and vorticity isolines,* $\omega = \nabla \times \mathbf{u}$*, for the flow past a cylinder at* $Re = 150$*. (b) For the same case, the streamwise component of the closure term,* $\mathbf{f}$*, is shown. In both cases, only a portion of the domain is displayed.*

Simulations start with null flow fields at $t = 0$. Required statistics, such as the mean flow $\overline{\mathbf{u}}$ and forcing stress term $\mathbf{f}$, are computed on-the-fly. The final simulation time T is determined by a convergence criterion, specifically when the L2-norm difference between consecutive mean flows falls below 10^{-8}.

Spatial discretization is achieved using a finite element method (FEM) approach via the `FEniCS` Python library (Alnæs et al., 2015), with time integration handled by a second-order backward differentiation formula. Meshes are refined near the obstacle and in the wake region to accurately capture flow dynamics. Depending on the specific case, they typically consist of around 13,500 nodes on average. Figure 2a depicts the streamwise component of the mean flow $\overline{\mathbf{u}}$ along with the vorticity isolines $\omega = \nabla \times \mathbf{u}$, while Figure 2b shows the streamwise component of the closure term $\mathbf{f}$ for the cylinder bluff body reference case at $Re = 150$.

4. GNN overview

In this section, we present an overview of the GNN architecture used in this study. While a comprehensive description of the GNN architecture can be found in Hamilton (2020), our focus here is to describe the specific implementation used in this work, tailored to the reconstruction of mean flow fields in CFD scenarios. In particular, we describe how unstructured mesh data are encoded as graphs, how physical quantities such as mean velocity fields and geometrical distances are integrated into the model as node and edge features, respectively, and how the GNN outputs the predicted forcing term required to solve the RANS equations. The implementation relies on the `PyTorch Geometric` library (Fey and Lenssen, 2019), which provides an efficient and scalable infrastructure for the message passing (MP) algorithms at the core of the GNN model.

Section 4.1 delves into the MP process adopted in our GNN; Section 4.2 outlines the specific structure of the input data used to encode CFD problems as graphs; and the GNN training and inference algorithm is presented in Section 4.3. For readers already familiar with Quattromini et al. (2025), we note that the GNN architecture described here is unchanged. However, it is employed in this work with a fundamentally different objective; those readers may wish to skip directly to Section 5, where the novel data assimilation strategy is introduced.

4.1. MP process

In GNNs, nodes iteratively exchange information with their neighbors to update their latent representations. This process, known as MP, allows GNNs to capture complex dependencies and patterns within data by considering the edges as important information relevant to the problem. Depending on the graph's size, the MP process can be repeated an arbitrary number of times, defined by a hyperparameter k.

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

A detailed list of the hyperparameters used is provided in Appendix A. The MP process is node-centered and consists of three fundamental steps:

1. **Message creation**: Each node i begins with an embedding state represented by a vector $\mathbf{h}_i$. Initially set to zero, this vector accumulates and processes information during the MP iterations. The dimension d_h of $\mathbf{h}_i$ is constant across all nodes and is a key model hyperparameter. This dimension determines the expressivity of the GNN, which reflects its ability to model complex functions (Hornik et al., 1989; Gühring et al., 2020). It is noteworthy that the embedded state itself does not have a direct physical interpretation, while the distance matrix indeed captures some physical relation between the nodes.
2. **Message propagation**: Information is propagated between nodes. To capture the convective and diffusive dynamics of the underlying CFD system, messages are exchanged bidirectionally between connected nodes. For a pair of connected nodes i and j, with a directed connection $\mathbf{a}_{ij}$ from i to j, the message generated is defined as:

$$\boldsymbol{\phi}_{i,j}^{(k)} = \zeta^{(k)}\left(\mathbf{h}_i^{(k-1)}, \mathbf{a}_{ij}, \mathbf{h}_j^{(k-1)}\right), \tag{4.1}$$

   where $\mathbf{h}_i^{(k-1)}$ is the embedded state from the previous MP layer $(k-1)$, and $\zeta^{(k)}$ is a differentiable operator, such as a multilayer perceptron (MLP) (Goodfellow et al., 2016). By swapping the indices i and j in Eq. 4.1, we obtain the message flowing from j to i. Depending on the number of j connected nodes in the neighboring set of i, namely $\mathcal{N}_i$, for each node i, the global outgoing message is then computed as:

$$\boldsymbol{\phi}_{i,\rightarrow} = \bigoplus_{j \in \mathcal{N}_i} \boldsymbol{\phi}_{i,j}, \tag{4.2}$$

   where $\oplus$ represents a differentiable, permutation-invariant function (e.g., sum, mean, or max). In our study, we chose the mean function to preserve permutation invariance, which is crucial when working with unstructured meshes where the neighborhood size of each node may vary. This ensures that the model can generalize across different mesh configurations.
3. **Message aggregation**: Each node i aggregates the received information to update its embedded state $\mathbf{h}_i^{(k)}$:

$$\mathbf{h}_i^{(k)} = \mathbf{h}_i^{(k-1)} + \alpha \Psi^{(k)}\left(\mathbf{h}_i^{(k-1)}, \mathbf{G}_i, \boldsymbol{\phi}_{i,\rightarrow}^{(k)}, \boldsymbol{\phi}_{i,\leftarrow}^{(k)}, \boldsymbol{\phi}_{i,\circlearrowleft}^{(k)}\right), \tag{4.3}$$

   where $\mathbf{G}_i = \{\overline{\mathbf{u}}, Re\}$ represents external quantities injected into the GNN (mean flow $\overline{\mathbf{u}}$ and Reynolds number Re in our case), provided at each update k. The terms $\boldsymbol{\phi}_{i,\rightarrow}^{(k)}$ and $\boldsymbol{\phi}_{i,\leftarrow}^{(k)}$ represent, respectively, the message sent to and received from all the neighboring nodes. The term $\boldsymbol{\phi}_{i,\circlearrowleft}^{(k)}$ is the self-message that the node i send to itself in order to maintain the node's own information while aggregating messages from its neighbors. Their mathematical definition, with the appropriate change in notation, is expressed in Eq. 4.1. The term $\Psi^{(k)}$ is a differentiable operator, typically an MLP, used to handle together the gathered information. The term α is a hyperparameter relaxation coefficient controlling the update scale (Appendix A).

At the end of the MP process, each node's embedded state has been updated k times, incorporating information from its neighbors. The choice of k is crucial for the model's generalization across graphs of varying sizes. As shown by Nastorg (2024), k should be proportional to the graph's diameter to ensure effective information propagation (Donon et al., 2020). However, since the graphs in the dataset used in this study exhibit a relatively consistent diameter, we opted to optimize this hyperparameter using genetic algorithms (Appendix A). Additionally, Nastorg (2024) explored recurrent or adaptive architectures, where k could dynamically adjust based on the graph's structure. Once the MP process concludes, the latest k-updated embedded state of each node is projected back into the physical space to predict the desired target, in this case, the forcing stress term $\mathbf{f}$. A differentiable operator, such as an MLP (decoder D), handles this operation.

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

4.2. Data structuring

Applying GNNs to unstructured CFD data requires representing the CFD mesh as a graph. In order to obtain the CFD-GNN interface, we align each mesh node with a GNN node. To this end, we structure the data into tensors that maintain the mesh's adjacency properties. Specifically, for each case in the ground truth dataset, we construct:

- A matrix $\mathbf{A} \in \mathbb{R}^{n_i \times d_h}$,

$$\mathbf{A}_{n_i,d_h} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,d_h} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,d_h} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n_i,1} & a_{n_i,2} & \cdots & a_{n_i,d_h} \end{bmatrix},$$

  where n_i is the number of nodes in the mesh and d_h is the dimension of the embedded state. This matrix $\mathbf{A}$ stacks together all the embedded arrays h_i defined on all the nodes.
- A matrix $\mathbf{C} \in \mathbb{R}^{c \times 2}$, where c is the number of edges in the mesh, defining the node connections. This matrix represents the adjacency scheme of the mesh.
- A matrix $\mathbf{E} \in \mathbb{R}^{c \times 2}$, containing the distances between connected nodes in the x and y directions. This matrix captures the geometric properties of the mesh's adjacency scheme, expressed as the node distances.

$$\mathbf{C}_{c,2} = \begin{bmatrix} i_1 & j_1 \\ i_2 & j_2 \\ \vdots & \vdots \\ i_c & j_c \end{bmatrix}, \quad \mathbf{E}_{c,2} = \begin{bmatrix} x_{i_1} - x_{j_1} & y_{i_1} - y_{j_1} \\ x_{i_2} - x_{j_2} & y_{i_2} - y_{j_2} \\ \vdots & \vdots \\ x_{i_c} - x_{j_c} & y_{i_c} - y_{j_c} \end{bmatrix}.$$

Each column of $\mathbf{A}$ serves as a feature vector for neurons in the MLPs used in the GNN (ζ, Ψ, and the decoder D). The MLPs architecture is defined by the dimension d_h of the embedded state, while the number of nodes n_i corresponds to the feature count per neuron. This approach enables the use of the same MLP architectures across different CFD simulations, regardless of geometry or node count, as the number of nodes does not affect the underlying structure of the MLP. This approach makes the GNNs particularly well-suited for interacting with unstructured meshes and learning from various geometries and configurations.

4.3. GNN training algorithm

The GNN training framework is shown in Figure 3. The process begins with $\mathbf{A}^0$, a matrix of zero-initialized embedded states (see Section 4.2). This matrix, along with external inputs $\mathbf{G}$ (mean flow $\overline{\mathbf{u}}$ and Reynolds number Re), is fed into the first MP^1 layer. The updated embedded state matrix $\mathbf{A}^1$ is then

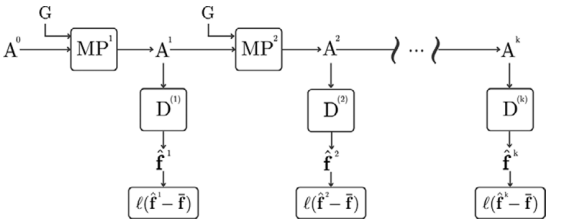


Figure 3. *The overall framework of our GNN training process. MP^k denotes the message passing algorithms, $D^{(k)}$ are the k decoders, which are trainable MLPs, A^k are the k matrices containing the embedded states of each node, and G is the vector containing the input injected into the GNN. The figure is inspired by Donon et al. (2020).*

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

passed through the decoder D^1, an MLP responsible for reconstructing the physical state $\hat{\mathbf{f}}^1$. The predicted forcing term $\hat{\mathbf{f}}^1$ is compared to the DNS ground truth $\mathbf{f}$ using a loss function:

$$\ell^k = \frac{1}{n_i}\sum_{i=1}^{n_i}\left(\mathbf{f}_i^k - \hat{\mathbf{f}}_i\right)^2, \tag{4.4}$$

where n_i is the number of nodes, and k is the layer number in the MP process. This procedure is repeated across the k layers of the GNN. Following Donon et al. (2020), the intermediate losses from all layers of the MP process are combined in a global loss function L to robustify the learning process:

$$L = \sum_{k=1}^{\overline{k}} \gamma^{\overline{k}-k} \cdot \ell^k, \tag{4.5}$$

where $\overline{k}$ is the number of layers, and γ is a hyperparameter controlling the weight of each loss term (see Appendix A). As the MP process goes on, each node collects more and more information. This exponential weighting ensures that later updates, which are richer in information, have greater influence on the learning process.

5. Methodology

This section describes the data assimilation scheme developed in this study, which combines the training process of a neural model based on GNN (Section 4) with the RANS equations (Section 2.1). The approach primarily focuses on using analytically derived gradients through the adjoint method (Section 2.2) to enhance the GNN learning process and ensure physical consistency in its predictions. This scheme will be referred to as the PhyCo-GNN. The complete end-to-end training process is illustrated in Figure 4, with technical details provided in Section 5.1 regarding the training process, in Section 5.2 for the GNN pretraining phase, and in Section 5.3 for the approach used to handle the transition from pretraining to the main training phase of the GNN.

5.1. Algorithm and training process

With reference to Figure 4, the global training process can be ideally divided into two phases: the forward step and the backward step.

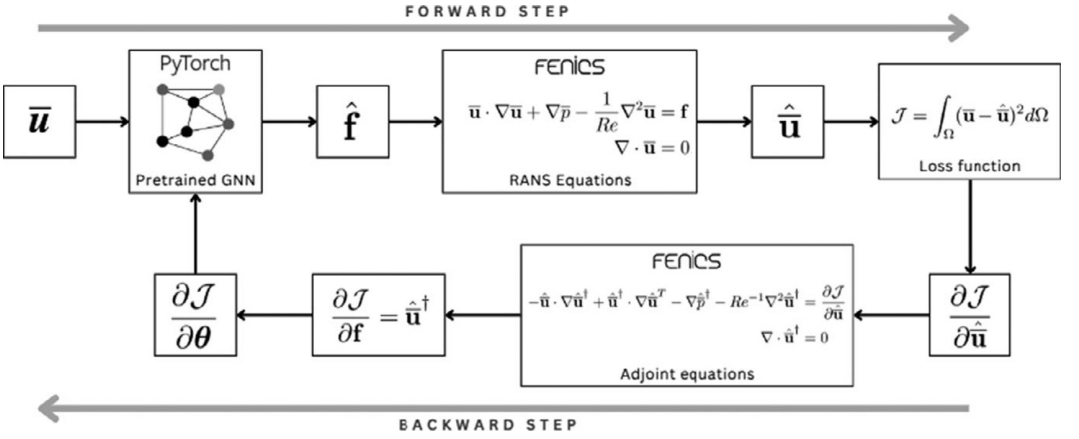


Figure 4. *End-to-end training loop: $\overline{\mathbf{u}}$ is the GNN's input mean flow, $\hat{\mathbf{f}}$ is the GNN's predicted forcing stress term, $\boldsymbol{\theta}$ represents the GNN's trainable parameters, and $\mathcal{J}(\overline{\mathbf{u}})$ is the cost function to minimize. For simplicity of notation, we consider the case where the ground truth corresponds to the entire flow field.*

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

The forward step begins by associating an embedded state vector $\mathbf{h}_0$, initially set to zero, as a node feature to each node of the GNN. These vectors, one for each mesh node, define the initial latent representation of the graph. Each node of the graph corresponds to a point in the computational mesh, and this mapping is preserved through the data structure described in Section 4.2. The Euclidean distances between nodes are used as edge features to define the graph connectivity and encode the local geometric relationships. The constructed graph is passed through a pretrained GNN (Section 5.2) and the state vector on each node is updated through a k-step MP process, as outlined in Section 4.1. During each MP iteration, nodes exchange and aggregate information with their neighbors, leveraging the geometric edge features. External physical quantities (the mean flow field $\overline{\mathbf{u}}$ and the Reynolds number Re) are injected during the aggregation step at each layer, providing the GNN with the necessary context to learn physically meaningful patterns. After k MP iterations, the GNN outputs a predicted forcing term $\hat{\mathbf{f}}$, which represents an estimate of the Reynolds stress closure term. This predicted forcing term $\hat{\mathbf{f}}$ is used as the closure term in the RANS equations (Eq. 2.3). These equations are numerically solved in an FEM framework using the Python library FEniCS (Alnæs et al., 2015), producing a reconstructed mean flow field $\widehat{\overline{\mathbf{u}}}$. This result is then compared with the mean flow ground truth $\overline{\mathbf{u}}$, obtained from the DNS, to compute the loss function $\mathcal{J}$, as expressed in Eq. 2.5.

The second phase, the backward step, starts with the requirement to compute the derivative of the loss function $\mathcal{J}$ with respect to the trainable $\boldsymbol{\theta}$ parameters of the GNN. The gradient chain rule for this term can be mathematically expressed as:

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathcal{J}}{\partial \widehat{\overline{\mathbf{u}}}} \cdot \frac{\partial \widehat{\overline{\mathbf{u}}}}{\partial \hat{\mathbf{f}}} \cdot \frac{\partial \hat{\mathbf{f}}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathcal{J}}{\partial \hat{\mathbf{f}}} \cdot \frac{\partial \hat{\mathbf{f}}}{\partial \boldsymbol{\theta}}. \tag{5.1}$$

The first term, $\frac{\partial \mathcal{J}}{\partial \hat{\mathbf{f}}}$, on the right-hand side, is obtained from Eq. 2.10 after solving the adjoint equations (Eq. 2.8) with the support of Eq. 2.9. This yields an analytical gradient of the cost function $\mathcal{J}$ with respect to $\widehat{\mathbf{f}}$. The second term, $\frac{\partial \hat{\mathbf{f}}}{\partial \boldsymbol{\theta}}$, represents the gradient of the GNN's output with respect to its $\boldsymbol{\theta}$ trainable parameters, which can be computed efficiently using the `PyTorch Geometric` automatic differentiation.

These two gradients—one analytically derived and discretized using FEniCS, and the other numerically computed via automatic differentiation in `PyTorch Geometric` (Fey and Lenssen, 2019)—are combined to complete the chain rule (Eq. 5.1) and obtain the full derivative that is used to update the GNN learnable parameters via gradient descent. The full forward–backward process is repeated iteratively until convergence, that is, when the cost function $\mathcal{J}$ falls below a predefined threshold.

5.2. On the pretraining step

A crucial step in the algorithm is the GNN model's pretraining phase. We can motivate this choice from two different perspectives. From a theoretical Bayesian viewpoint, the objective is to infer the posterior distribution of the GNN parameters θ given observations of the mean velocity field $\overline{\mathbf{u}}$, that is, $p(\theta|\overline{\mathbf{u}})$. Since the forcing field $\mathbf{f}$ is not observed directly, but acts as a latent variable linking θ to $\overline{\mathbf{u}}$ via the RANS equations, we introduce $\mathbf{f}$ as an auxiliary variable and marginalize over it. This yields:

$$p(\theta|\overline{\mathbf{u}}) \propto \int p(\overline{\mathbf{u}}|\mathbf{f}) p(\mathbf{f}|\theta) p(\theta) d\mathbf{f}, \tag{5.2}$$

where $p(\overline{\mathbf{u}}|\mathbf{f})$ models the likelihood of the observed mean flow given a candidate forcing field, $p(\mathbf{f}|\theta)$ is the GNN model predicting $\mathbf{f}$ from input features, and $p(\theta)$ is the prior over the model parameters, which we assume to be uniform in the absence of other information. This decomposition highlights the role of $\mathbf{f}$ as a latent intermediate quantity through which the model learns to match the observed $\overline{\mathbf{u}}$. Pretraining the GNN on DNS-based samples of $\mathbf{f}$ provides a meaningful initialization for $p(\mathbf{f}|\theta)$, thus guiding the posterior inference on θ more stably and efficiently during the later reconstruction of $\overline{\mathbf{u}}$. Moreover, the GNN's weights and biases are set using a default initialization (He et al., 2015), meaning that early predictions from the GNN are nonphysical and cannot reliably be used in the forward step, where the forcing term is introduced into the RANS equations to compute the mean flow (Section 5.1). In fact, the solution to the RANS problem may not

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

exist if the initial guess for the forcing term, $\hat{\mathbf{f}}$, is too far from a physically meaningful value. Thus, from the numerical viewpoint, the pretraining phase stabilizes the GNN's output and mitigates this issue, predicting the forcing stress term $\hat{\mathbf{f}}$ suitable for integration into the RANS equations.

The pretrained model is obtained through pure supervised learning (Quattromini et al., 2025), where the mean flow $\overline{\mathbf{u}}$ (and Reynolds number Re) are used as input, and the forcing stress term $\mathbf{f}$ from DNS data is the target. The loss function used in this phase is the mean squared error loss $\mathcal{L}$, already discussed in Section 4.3, where n_i is the number of nodes in the GNN. The number of epochs required to reach the desired closure term accuracy is not fixed a priori, but is determined based on the convergence behavior of the GNN during the pretraining phase. In particular, the transition to the main training phase occurs only when the GNN has learned to produce physically plausible forcing terms $\hat{\mathbf{f}}$, which allow the RANS equations to be solved via the FEM framework. To ensure consistency across test cases, a standard protocol is adopted: for simple reference scenarios, such as those presented in Section 6.1, the pretraining is fixed at 500 epochs. For more complex cases, which involve larger datasets, the pretraining phase is extended to 1,000 epochs. This choice reflects the increased training effort required to reach comparable levels of accuracy in more challenging settings.

In this context, the closure term accuracy refers to the level of precision necessary for the GNN to generate predictions that allow the FEM solver to successfully solve the RANS equations. Throughout the pretraining phase, the GNN's predictions are periodically evaluated by solving a test FEM step. If the solver converges and accurately resolves the RANS equations using the GNN-predicted closure term, the pretraining phase is considered complete. This ensures that the GNN has learned a reliable and accurate representation of the closure term, making it suitable for the full training process.

5.3. On the loss function

During the pretraining step (Section 5.2), the GNN model is updated using a loss function designed to align the model's predictions with the available DNS data. As previously mentioned, this phase serves as a warm-up for the subsequent main training (Section 5.1). However, when pretraining ends and the main training begins, a different loss function is adopted, as can be seen by comparing Eq. 4.5 with Eq. 2.5. This change may negatively affect convergence and destabilize the training process, as the two optimization landscapes can be quite different. To mitigate this risk, both loss functions are retained during the main training phase and combined using a weight coefficient, β. In particular, the loss function $\mathcal{L}$ (Eq. 4.5), associated with supervised pretraining, continues to enforce a data-driven alignment and ensures "continuity" in the optimization process. On the other hand, the term $\mathcal{J}$ (Eq. 2.5), corresponding to the physics-constrained loss, minimizes the mean flow reconstruction error. Thus, the global loss function for the main training loop (Section 5.1) evolves as:

$$\mathcal{H} = (1-\beta)\mathcal{L} + \beta\mathcal{J} = (1-\beta)\left(\frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{f}_i - \hat{\mathbf{f}}_i\right)^2\right) + \beta\left(\frac{1}{2}\int_{\Omega}\left(\overline{\mathbf{m}} - \mathcal{M}\left(\widehat{\overline{\mathbf{u}}}\right)\right)^2 d\Omega\right). \tag{5.3}$$

This strategy ensures a smooth transition between the two optimization steps by adjusting the relative importance of the pretraining and main training loss functions.

The next section discusses the results, showing that the converged GNN model effectively predicts a forcing term $\hat{\mathbf{f}}$ that aligns with the ground truth, $\mathbf{f}$ allowing an effective learned model to reconstruct the mean flow $\overline{\mathbf{u}}$.

6. Results

In this section, we present the improvements achieved using the PhyCo-GNN data assimilation scheme for reconstructing the mean flow field $\overline{\mathbf{u}}$. The tests are conducted across several scenarios, with a particular focus on reconstructing the mean flow from sparse measurements, noisy probes, and incomplete flow fields (inpainting). The models are compared with the pure supervised learning method, which serves as a

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

baseline. The supervised learning approach is thoroughly discussed in Quattromini et al. (2025): the GNN model is trained solely by learning the forcing stress from the DNS data. Therefore, the cost function is chosen to minimize the discrepancy between the predicted forcing stress and the ground truth, without incorporating any constraints based on the system's physics. The mean flow $\overline{\mathbf{u}}$ is then computed by using the forcing stress modeled with the GNN as input to the RANS equations in Eq. 2.3. In contrast, the hybrid data assimilation scheme introduced in this work incorporates a physics-based constraint during the training process of the GNN. To compare the two methods, we evaluate their training curves (after the pretraining phase) by identifying the minimum loss values achieved by each model during training. Unless otherwise stated, all test cases use the full velocity field from DNS as input; the sparse probe setting applies exclusively to the test in Section 6.3. As a metric for comparison, we consider the percentage improvement, defined as:

$$\mathcal{I}(\%) = \frac{min(\mathcal{J}_S) - min(\mathcal{J}_{PC})}{min(\mathcal{J}_S)} \cdot 10^2, \tag{6.1}$$

where $min(\mathcal{J}_S)$ and $min(\mathcal{J}_{PC})$ represent the minimum values of the loss function for mean flow reconstruction (Eq. 2.5) in the supervised learning method and the adjoint-based PhyCo-GNN method, respectively. In the following, we introduce different learning tasks, focusing on the technical features of the method and discussing the improvements achieved in terms of mean flow reconstruction.

6.1. Reference cases

The first test case we consider is the reconstruction of the flow field when the input to the GNN is the complete mean flow $\overline{\mathbf{u}}$ and Reynolds number Re over the entire computational domain Ω. This test case is introduced as a reference for the method. Throughout this section, the mapping operator $\mathcal{M}$ used in the definition of the cost function is taken as the identity operator, implying that the comparison is performed over the full velocity field without any spatial filtering or subsampling.

We consider two cases of increasing complexity. The first case involves flow past a 2D cylinder at a Reynolds number of $Re = 150$. This case is well-documented in the literature, and its mean flow is shown in Figure 5a. The training dataset includes only the mean flow $\overline{\mathbf{u}}$ (used as input to the GNN) and its corresponding forcing term $\mathbf{f}$ (the GNN target). The training curves in Figure 5b show that, starting from the pretraining phase, the implementation of the approach described in this article leads to a substantial improvement in mean flow reconstruction. Specifically, the improvement reaches a value of $\mathcal{I} = 58.59\%$.

The second case involves a two side-by-side cylinder configuration, also known in the literature as the "flip-flop" case, at a Reynolds number of $Re = 90$ (Carini et al., 2014). Its mean flow is shown in Figure 6a. The training curves for this case in Figure 6b demonstrate an even more pronounced improvement, with a reduction of $\mathcal{I} = 82.90\%$ in the loss curve. The results not only indicate the broad adaptability of the PhyCo-GNN but also show how, in more complex scenarios, the underlying physics and governing equations play a crucial role in further enhancing the accuracy of the GNN model's predictions.

6.2. Generalization

In this section, we test the generalization capabilities of the model obtained using the PhyCo-GNN scheme. The training dataset consists of three cases involving a 2D cylinder at Reynolds numbers of $Re = [90,110,130]$. In contrast, the validation dataset includes data points not present in the training set, corresponding to simulations of the flow around a 2D cylinder at Reynolds number $Re = 120$ for interpolation testing, and at $Re = 150$ for testing the extrapolation capabilities. Throughout this generalization test, the mapping operator $\mathcal{M}$ used in the cost function remains the identity operator. In Figure 7a, the ground truth mean flow $\overline{\mathbf{u}}$ for the $Re = 120$ case is shown. Based on the validation cases, we observe an improvement in the mean flow reconstruction, with an average improvement of $\mathcal{I} = 73.27\%$ across the entire validation dataset. Specifically, we achieve an improvement of $\mathcal{I} = 78.96\%$ for the interpolation

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

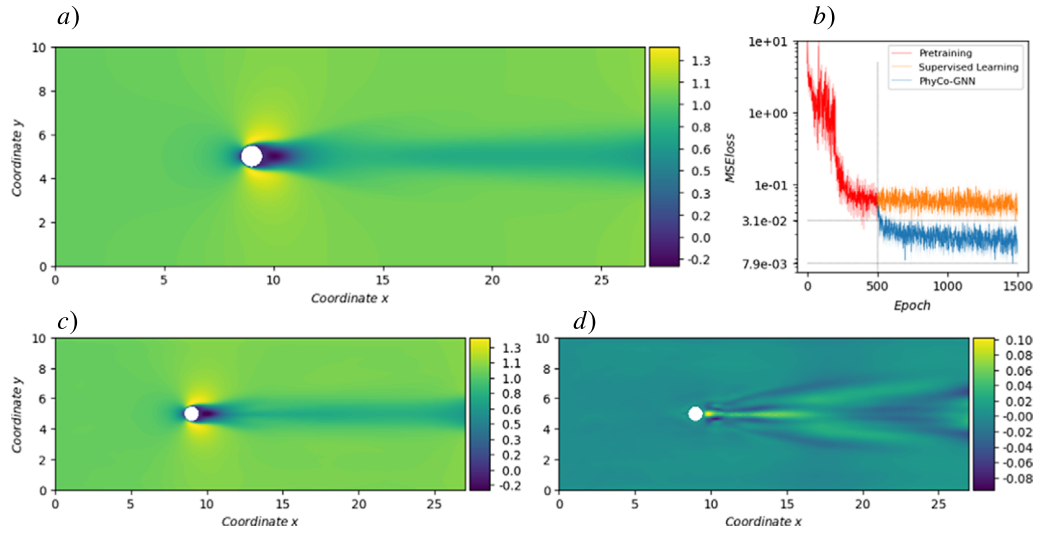


Figure 5. *Training dataset: one mean flow-forcing pair ($Re = 150$); the GNN input of the mean flow from the ground truth is shown in (a). (b) Loss curves for the pure supervised approach (orange line), the proposed PhyCo-GNN method (blue line), and the pretraining phase (red line). Shadow colors highlight standard deviations computed over five independent training runs with different parameter initializations. The two horizontal dotted lines indicate the minimum values of the supervised and proposed methods. (c) Reconstructed mean flow obtained with the PhyCo-GNN approach. (d) Contour plot of the reconstruction error difference between the pure supervised approach and PhyCo-GNN.*

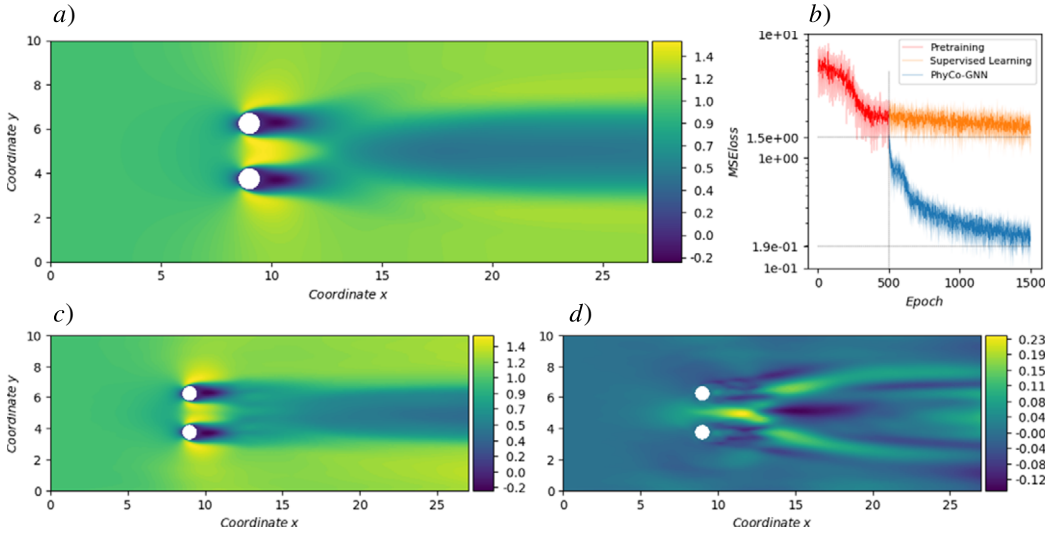


Figure 6. *Training dataset: one mean flow-forcing pair at $Re = 90$; the mean flow is shown in (a) and is used as GNN input; legend for (b)–(d) are as in Figure 5.*

case at $Re = 120$, and $\mathcal{I} = 13.96\%$ for the extrapolation case at $Re = 150$. The improvement on the training cases is $\mathcal{I} = 40.16\%$ on average across the entire training dataset.

The primary objective of this test case is to demonstrate that the presented approach enhances the generalization capabilities of the GNN model. To ensure clarity in our analysis, this generalization test

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

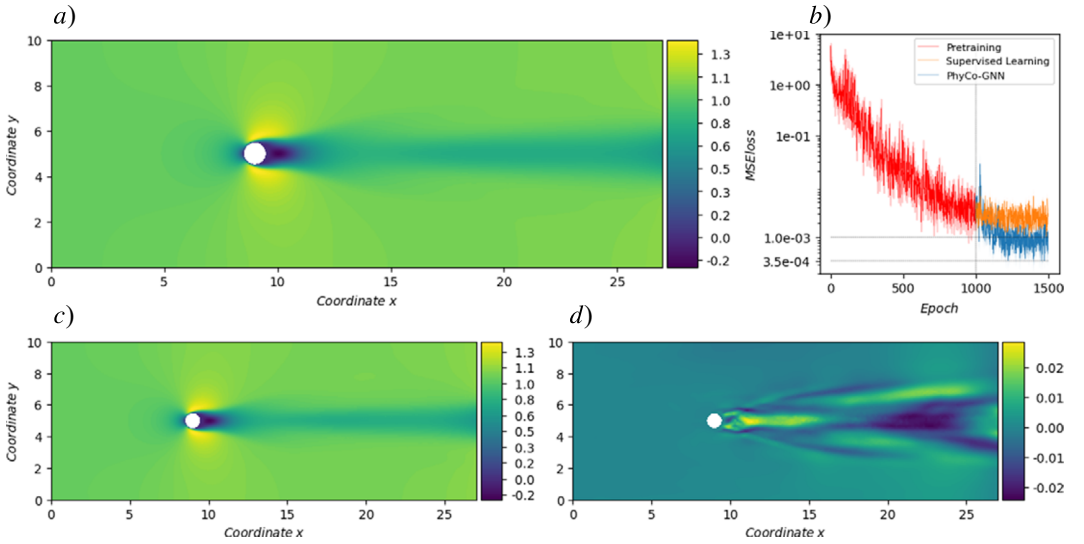


Figure 7. *Generalization test—training dataset: three mean flow-forcing pairs at* $Re = [90,110,130]$*; validation dataset:* 2 *mean flow-forcing pairs at* $Re = [120,150]$*. (a) ground truth mean flow at* $Re = 120$ *from the validation dataset, used as GNN input; legend for (b)–(d) are as in Figure 5.*

case is intentionally separated from the others. This separation allows for a focused evaluation of each individual test case, targeting the specific goals of those tests without introducing confounding variables related to generalization. It should be noted that a more extensive assessment of the GNN's ability to generalize across a range of geometric configurations—such as variations in bluff body shapes or positions—is beyond the scope of this study. These aspects have been examined in detail by Quattromini et al. (2025), in combination with active learning strategies. In the present work, we focus on a simplified setting with limited generalization in order to clearly evaluate the role of the physics-constrained training scheme in predicting the mean flow.

6.3. Sparse measurements

The learning task presented here involves the reconstruction of the mean flow over the entire computational domain using measurements from randomly distributed probes as input for the GNN. In this case, the mapping operator $\mathcal{M}$ appearing in the cost function is defined as the projection onto the sparse set of probe locations, so that the comparison between predicted and reference fields is performed only at the measurement points. The numerical treatment of the term forcing the adjoint equation—based on the sparse measurements—follows the one included in Dimitry et al. (2014). The training dataset consists of two simulations of the flow past a cylinder for each Reynolds number in the range $Re = [90,110,130]$, resulting in six cases. For each case, 450 probes are placed along the mean flow stream, uniformly distributed across the entire computational domain Ω. Subsequently, 200 of these probes are randomly removed, leaving a sparse set of 250 probes. This sparse set of measurements of the mean flow $\overline{\mathbf{u}}$ is used as input to the GNN, while the output prediction is compared with the corresponding forcing stress tensor from the DNS ground truth. Figure 8a shows the positioning of the random probes on the mean flow, while Figure 8b shows the average training curves across the training dataset.

In this case, we demonstrate an improvement in mean flow reconstruction across all the training cases, with an average improvement of $\mathcal{I} = 55.09\%$. This result highlights the robustness of the proposed approach in scenarios with sparse and randomly distributed measurements.

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

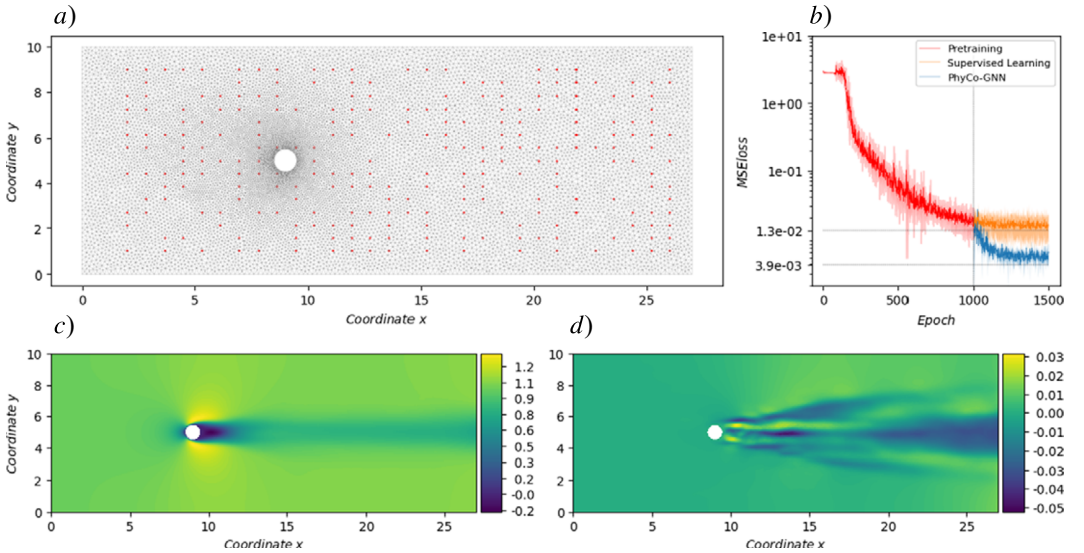


Figure 8. *Sparse measurements—training dataset: 250 randomly distributed probes on* 6 *mean flow-forcing pairs at* $Re = [90,110,130]$*, with two instances for each* Re*). (a) Random probes positioning on the mean flow; legend for (b)–(d) are as in Figure 5.*

6.4. Denoising

In this test case, the input mean flow field is perturbed with Gaussian noise. The probability density function used for the Gaussian distribution to generate the noise is given by:

,

$$\psi(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(z-\mu)^2}{2\sigma^2}} \tag{6.2}$$

where z is the random variable, μ is the mean value of the normal distribution, and σ represents its standard deviation. In this case, we assume $\mu = 0$, that is, a standard normal distribution. The mapping operator $\mathcal{M}$, in this case, remains the identity operator. The training dataset consists of three cases of cylinder flows at Reynolds numbers $Re = [90,110,130]$, each perturbed with Gaussian noise of fixed standard deviation $\sigma = 0.4$. Figure 9a shows the effect of $\sigma = 0.4$ Gaussian noise on the mean flow (at $Re = 130$), while Figure 9b presents the accuracy of the mean flow reconstruction. The goal is to remove the Gaussian noise and accurately reconstruct the denoised mean flow field. Our approach demonstrates an improvement of $\mathcal{I} = 47.52\%$ on the training dataset, on average, across all cases.

Although not shown here, we also conducted additional tests at fixed Reynolds number ($Re = 130$), with varying noise levels $\sigma = [0.2,0.4,0.6]$. The results confirmed superior performance of the proposed approach compared to purely supervised learning, highlighting its robustness to different noise intensities.

6.5. Inpainting

In this test, masking patches are randomly applied to the input mean flow field. The training dataset consists of three cases of a cylinder obstacle at Reynolds numbers $Re = [90,110,130]$, each with different patch locations (Figure 10a). In this setup, the mapping operator $\mathcal{M}$ used in the cost function excludes the masked regions, meaning that the comparison between the predicted and reference fields is performed only over the unmasked areas of the domain. The goal is to reconstruct the mean flow field by filling in the missing patches. The approach shows improvements on the training cases with an average increase of $\mathcal{I} = 41.73\%$, successfully restoring the missing portions of the field and enhancing the overall reconstruction accuracy.

A summary of the results is shown in Table 1, where the relative errors between the reconstructed mean flow and the ground truth are reported for both the pure supervised technique based on the GNN models

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

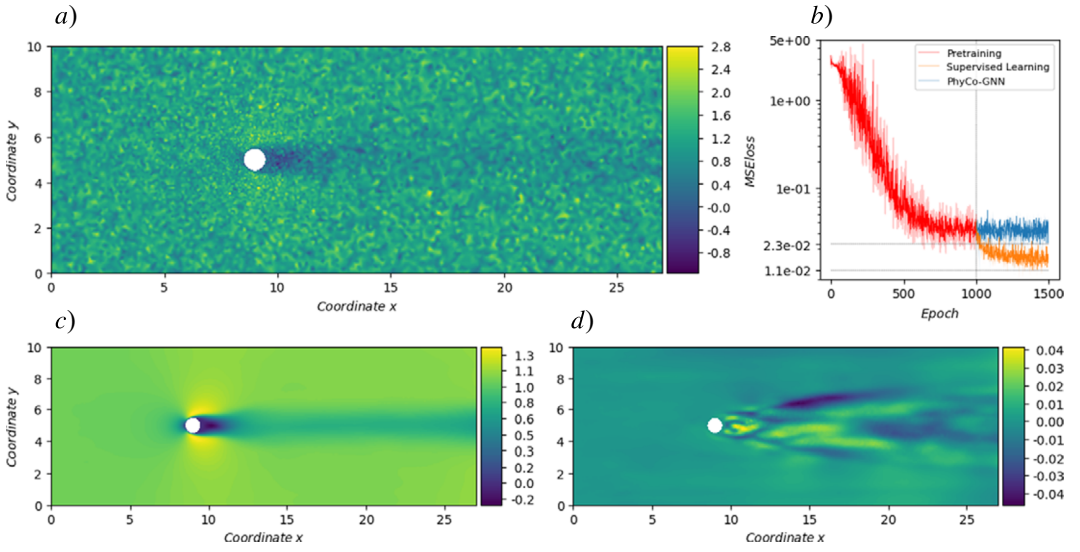


Figure 9. *Denoising—training dataset: three mean flow-forcing pairs at* $Re = [90,110,130]$*, perturbed with Gaussian noise; (a) mean flow at* $Re = 130$*; legend for (b)–(d) are as in Figure 5.*

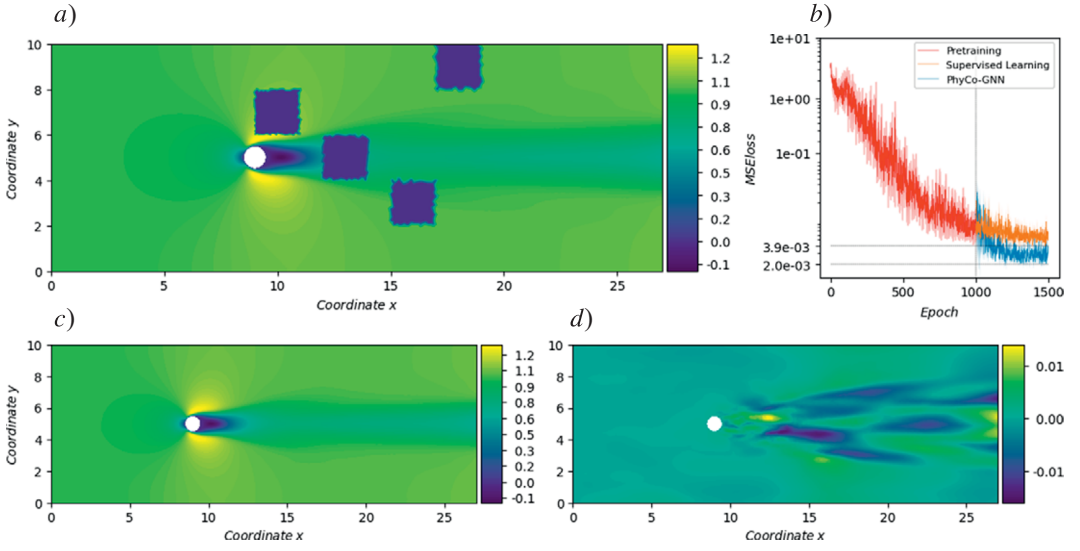


Figure 10. *Inpainting—training dataset: three mean flow-forcing pairs at* $Re = [90,110,130]$*, with randomly located patching masks; (a) mean flow at* $Re = 110$*; legend for (b)–(d) are as in Figure 5.*

(see also Quattromini et al., 2025) and the data-assimilation scheme PhyCo-GNN. The latter outperforms vanilla supervised learning in all cases, as already shown in the previous paragraphs.

7. Conclusion

In this work, we introduced a direct-adjoint data assimilation scheme based on the RANS equations, where the closure model is based on GNNs. The resulting scheme, named PhyCo-GNN, demonstrated the potential of combining ML techniques with well-established physical principles to improve the accuracy

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

Table 1. *Summary of the reconstruction errors for each test considered in Section 6. The cases listed correspond to those presented in Figures 5–10. For each case, a comparison is made between the baseline method based on supervised learning and the performance of PhyCo-GNN, with the errors reported as relative errors in the 2-norm*

	Reconstruction error	
Cases	Supervised	Phy-Co GNN
Reference case, Re = 150	0.0190	0.0126
Reference case, Re = 90 (flip-flop)	0.0910	0.0843
Generalization, Re = 120	0.0087	0.0069
Sparse measurements, Re = 110	0.0134	0.0092
Gaussian noise, Re = 130	0.0200	0.0103
Inpainting, Re = 110	0.0056	0.0026

and reliability of mean flow reconstruction. More specifically, the forcing term of the RANS equations is modeled using a GNN model informed by gradients obtained through autodifferentiation of the NN and gradients computed analytically based on the adjoint equations associated with the iterative process. On one hand, these gradients ensure that the learned model adheres to the underlying physics. On the other hand, the closure model benefits from the flexibility of GNNs in handling unstructured meshes, making the framework particularly well-suited for complex geometries often encountered in CFDs. Finally, the combination of GNN and adjoint-based methods mitigates the dependency on large datasets, as the inclusion of physics reduces the need for extensive training data to achieve reliable predictions.

We tested the PhyCo-GNN framework across several scenarios, with a particular focus on reconstructing the mean flow from sparse measurements, noisy probes, and incomplete flow fields (inpainting). The models are compared with a supervised learning method, which serves as a benchmark, where the learning process of the GNN model is not constrained by the adjoint but solely relies on numerical data. In particular, we consider as the reference case the mean flow reconstruction of flows past a 2D cylinder at a Reynolds number of $Re = 150$ and a two-cylinder configuration at $Re = 90$. Other tests include generalization by considering interpolation and extrapolation with respect to unseen Reynolds numbers during the training process, reconstruction from noisy inputs, and inpainting, which involves the reconstruction of the mean flow field by filling in the missing patches. For all test cases, the PhyCo-GNN approach showed substantial improvements compared to the reference method based on pure supervised learning.

Future work will explore the extension of the presented framework to more complex scenarios, including higher Reynolds number regimes, three-dimensional flows, and experimental datasets, which are often characterized by sparse, noisy, or incomplete measurements. While the methodological structure of the PhyCo-GNN framework is, in principle, compatible with such configurations, the transition to large-scale 3D domains poses significant computational challenges. These include the increasing cost of mesh resolution, memory usage during GNN training, and the resource demands of adjoint-based Partial Differential Equations (PDE) solvers. Addressing these limitations will require adopting scalable strategies, such as domain decomposition for graph-based learning (a review can be found in Dolean et al. (2024), modular and multiscale GNN architectures, and multi-GPU distributed training pipelines. In parallel, the adoption of more efficient FEM backends and the exploration of alternative network architectures, such as physics-informed transformers or recurrent graph networks, may further enhance the expressiveness and applicability of the framework.

Data availability statement. All code and data supporting this study are openly available at https://doi.org/10.17605/OSF.IO/YVK2N (Quattromini et al., n.d.) and allow full replication of the results.

Author contribution. Conceptualization: M.Q., M.A.B., S.C., and O.S. Methodology: M.Q. Data curation: M.Q. Data visualization: M.Q. and O.S. Writing—original draft: M.Q. Writing—review and edits: M.Q., M.A.B., S.C., and O.S. All authors approved the final submitted draft.

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

Funding statement. The PhD fellowship of M.Q. is supported by the Italian Ministry of University. This study has been partially funded under the National Recovery and Resilience Plan (NRRP), Mission 4 Component 2 Investment 1.3—Call for tender No. 1561 of 11.10.2022 Project code PE0000021, Project title Network 4 Energy Sustainable Transition (NEST), funded by the European Union–NextGenerationEU. The support from the Agence Nationale de la Recherche (ANR) through grant ANR-21-REASON is gratefully acknowledged.

Competing interests. The authors declare none.

References

Akiba T, **Sano S**, **Yanase T**, **Ohta T and Koyama M** (2019) Optuna: A next-generation hyperparameter optimization framework. In *ACM SIGKDD, Proceeding of the 25th ACM SIGKDD International Conference On Knowledge Discovery & Data Mining*. Anchorage, AK: Association for Computing Machinery, pp. 2623–2631.

Alnæs M, **Blechta J**, **Hake J**, **Johansson A**, **Kehlet B**, **Logg A**, **Richardson C**, **Ring J**, **Rognes ME and Wells GN** (2015) The FEniCS project version 1.5. *Architecture Numerical Software 3*(100).

Bae HJ and Koumoutsakos P (2022) Scientific multi-agent reinforcement learning for wall-models of turbulent flows. *Nature Communications 13*(1), 1443.

Beck A and Kurz M (2021) A perspective on machine learning methods in turbulence modeling. *GAMM Mitt. 44*(1), e202100002.

Brunton SL, **Noack BR and Koumoutsakos P** (2020) Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics 52*, 477–508.

Cai J, **Angeli P-E**, **Martinez J-M**, **Damblin G and Lucor D** (2024) Revisiting tensor basis neural network for Reynolds stress modeling: Application to plane channel and square duct flows. *Computers and Fluids 275*, 106246.

Carini M, **Giannetti F and Auteri F** (2014) On the origin of the flip–flop instability of two side-by-side cylinder wakes. *Journal of Fluid Mechanics 742*, 552–576.

Cécora RD, **Radespiel R**, **Eisfeld B and Probst A** (2015) Differential Reynolds-stress modeling for aeronautics. *AIAA Journal 53*(3), 739–755.

Cheng S, **Argaud J-P**, **Iooss B**, **Lucor D and Ponçot A** (2019) Background error covariance iterative updating with invariant observation measures for data assimilation. *Stochastic Environmental Research and Risk Assessment 33*(11), 2033–2051.

Cherubini S, **Robinet J-C**, **Bottaro A and De Palma P** (2010) Optimal wave packets in a boundary layer and initial phases of a turbulent spot. *Journal of Fluid Mechanics 656*, 231–259.

Cherubini S, **Robinet J-C and De Palma P** (2013) Nonlinear control of unsteady finite-amplitude perturbations in the blasius boundary-layer flow. *Journal of Fluid Mechanics 737*, 440–465.

Cinnella P (2024) Data-driven turbulence modeling. *arXiv preprint arXiv:2404.09074*.

Cordier L, **Noack BR**, **Tissot G**, **Lehnasch G**, **Delville J**, **Balajewicz M**, **Daviller G and Niven RK** (2013) Identification strategies for model-based control. *Experiments in Fluids 54*, 1–21.

Deng N, **Noack BR**, **Morzyński M and Pastur LR** (2020) Low-order model for successive bifurcations of the fluidic pinball. *Journal of Fluid Mechanics 884*, A37.

Dimitry PGF, **Dovetta N**, **Sipp D and Schmid PJ** (2014) A data-assimilation method for Reynolds-averaged Navier–Stokes-driven mean flow reconstruction. *Journal of Fluid Mechanics 759*, 404–431.

Dolean V, **Heinlein A**, **Mishra S and Moseley B** (2024) Multilevel domain decomposition-based architectures for physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering 429*, 117116.

Donon B, **Liu Z**, **Liu W**, **Guyon I**, **Marot A and Schoenauer M** (2020) *Deep statistical solvers*, adv. *Neural Information Processing Systems 33*, 7910–7921.

Dupuy D, **Odier N**, **Lapeyre CJ and Papadogiannis D** (2023) Modeling the wall shear stress in large-eddy simulation using graph neural networks. *Data-Centric Engineering 4*, e7.

Duraisamy K, **Iaccarino G and Xiao H** (2019) *Turbulence modeling in the age of data*, Annu. *Rev. Fluid Mech. 51*, 357–377.

Eivazi H, **Tahani M**, **Schlatter P and Vinuesa R** (2022) Physics-informed neural networks for solving Reynolds-averaged Navier–Stokes equations. *Physics of Fluids 34*(7), 075117.

Fey M and Lenssen JE (2019) Fast graph representation learning with PyTorch geometric. *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Giannetti F and Luchini P (2006) Leading-edge receptivity by adjoint methods. *Journal of Fluid Mechanics 547*, 21–53.

Giannetti F and Luchini P (2007) Structural sensitivity of the first instability of the cylinder wake. *Journal of Fluid Mechanics 581*, 167–197.

Giannotta A, **Cherubini S and De Palma P** (2022) Minimal energy thresholds for triggering in the rijke tube: The effect of the time delay. *Journal of Fluid Mechanics 938*, A23.

Goodfellow I, **Bengio Y and Courville A** (2016) *Deep Learning*. Massachusetts Institute of Technology. Available at http://www.deeplearningbook.org.

Gühring I, **Raslan M and Kutyniok G** (2020) Expressivity of deep neural networks. *arXiv preprint arXiv:2007.04759 34*.

Hamilton WL (2020) Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning 14*(3), 1–159.

He K, **Zhang X**, **Ren S and Sun J** (2015) Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proceeding IEEE International Conference Computer Visualization*, 1026–1034.

Hornik K, **Stinchcombe M and White H** (1989) Multilayer feedforward networks are universal approximators. *Neural Networks 2* (5), 359–366.

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

Lapeyre CJ (2019) Antony Misdariis, Nicolas Cazard, Denis Veynante, and Thierry Poinsot, *training convolutional neural networks to estimate turbulent sub-grid scale reaction rates*. *Combustion and Flame 203*, 255–264.

Ling J and Templeton J (2015) Evaluation of machine learning algorithms for prediction of regions of high Reynolds-averaged Navier Stokes uncertainty. *Physics of Fluids 27*(8), 085103.

Ling J, **Kurzawski A and Templeton J** (2016) Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics 807*, 155–166.

Lino M, **Fotiadis S**, **Bharath AA and Cantwell CD** (2023) Current and emerging deep-learning methods for the simulation of fluid dynamics. *Proceedings of the Royal Society A 479*(2275), 20230058.

Luchini P and Bottaro A (2014) Adjoint equations in stability analysis. *Annual Review of Fluid Mechanics 46*(1), 493–517.

Luchini P, **Giannetti F and Pralits JO** (2009) Structural sensitivity of the finite-amplitude vortex shedding behind a circular cylinder. *Solid Mechanics and its Applications 14*. https://doi.org/10.1007/978-1-4020-9898-7_12.

Magri L (2019) Adjoint methods as design tools in thermoacoustics. *Applied Mechanics Reviews 71*(2), 020801.

Marquet O, **Sipp D and Jacquin L** (2008) Sensitivity analysis and passive control of cylinder flow. *Journal of Fluid Mechanics 615*, 221–252.

McConkey R, **Yee E and Lien F-S** (2022) On the generalizability of machine-learning-assisted anisotropy mappings for predictive turbulence modelling. *International Journal of Fluid Dynamics 36*(7), 555–577.

Meldi M and Poux A (2017) A reduced order model based on Kalman filtering for sequential data assimilation of turbulent flows. *Journal of Computational Physics 347*, 207–234.

Nastorg M (2024) *Scalable GNN Solutions for CFD Simulations* [Thesis]. Université Paris-Saclay. Available at https://theses.hal.science/tel-04590477.

Navon IM (2009) Data assimilation for numerical weather prediction: A review. In Park SK and Xu L (eds), *Data Assimilation for Atmospheric, Oceanic and Hydrologic Applications*. Berlin, Heidelberg: Springer, pp. 21–65.

Patel Y, **Mons V**, **Marquet O and Rigas G** (2024) Turbulence model augmented physics-informed neural networks for mean-flow reconstruction. *Physical Review Fluids 9*(3), 034605.

Provansal M, **Mathis C and Boyer L** (1987) Bénard-von kármán instability: Transient and forced regimes. *Journal of Fluid Mechanics 182*, 1–22.

Quattromini M, **Bucci MA**, **Cherubini S and Semeraro O** (2025) Active learning of data-assimilation closures using graph neural networks. *Theoretical and Computational Fluid Dynamics 39*.

Quattromini M, **Bucci MA**, **Cherubini S and Semeraro O** Replication data for: Mean flow data assimilation using physics-constrained graph neural networks.

Roshko A (1954) On the drag and shedding frequency of two-dimensional bluff bodies. *Technical Report.*

Sanchez-Gonzalez A, **Heess N**, **Springenberg JT**, **Merel J**, **Riedmiller M**, **Hadsell R and Battaglia P** (2018) Graph networks as learnable physics engines for inference and control. *International Conference on Machanical Learning*, 4470–4479.

Schmelzer M, **Dwight RP and Cinnella P** (2020) Discovery of algebraic Reynolds-stress models using sparse symbolic regression. *Flow, Turbulence and Combustion 104*, 579–603.

Schmid PJ and Henningson DS (1994) Optimal energy density growth in Hagen–Poiseuille flow. *Journal of Fluid Mechanics 277*, 197–225.

Semeraro O, **Pralits JO**, **Rowley CW and Henningson DS** (2013) Riccati-less approach for optimal control and estimation: An application to two-dimensional boundary layers. *Journal of Fluid Mechanics 731*, 394–417.

Shukla K, **Xu M**, **Trask N and Karniadakis GE** (2022) Scalable algorithms for physics-informed neural and graph networks. *Data-Centric Engineering 3*, e24.

Singh AP and Duraisamy K (2016) Using field inversion to quantify functional errors in turbulence closures. *Physics of Fluids 28*(4), 045110.

Ströfer CA and Xiao H (2021) End-to-end differentiable learning of turbulence models from indirect observations. *TAML 11*(4), 100280.

Titaud O, **Vidard A and Souopgui I** (2010) *Assimilation of image sequences in numerical models*, Tellus a. *Dynamic Meteorology and Oceanography 62*(1), 30–47.

Toshev AP, **Galletti G**, **Brandstetter J**, **Adami S, and Adams NA** (2023) Learning lagrangian fluid mechanics with E(3)-equivariant graph neural networks. In *International Conference Geometry Science Information*. Cham: Springer, pp. 332–341.

Vinuesa R and Brunton SL (2022) Enhancing computational fluid dynamics with machine learning. *Nature Computational Science 2*(6), 358–366.

Volpiani PS, **Meyer M**, **Franceschini L**, **Dandois J**, **Renac F**, **Martin E**, **Marquet O and Sipp D** (2021) Machine learning-augmented turbulence modeling for RANS simulations of massively separated flows. *Physical Review Fluids 6*(6), 064607.

Wallin S and Johansson AV (2000) An explicit algebraic Reynolds stress model for incompressible and compressible turbulent flows. *Journal of Fluid Mechanics 403*, 89–132.

Weatheritt J and Sandberg R (2016) A novel evolutionary algorithm applied to algebraic modifications of the rans stress–strain relationship. *Journal of Computational Physics 325*, 22–37.

Wilcox DC (1998) *Turbulence Modeling for CFD*, Vol. *2*. La Canada Flintridge, CA: DCW Industries

Zhao Y, **Akolekar HD**, **Weatheritt J**, **Michelassi V and Sandberg RD** (2020) RANS turbulence model development using CFD-driven machine learning. *Journal of Computational Physics 411*, 109413.

Zhou Z, **He G and Yang X** (2021) Wall model based on neural networks for LES of turbulent flows over periodic hills. *Physical Review Fluids 6*(5), 054610.

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press

A. Appendix

Hyperparameters

The performance of a neural network is determined by its hyperparameters, which are set before training and influence both the model's architecture and the training process. These parameters define the network's ability to approximate complex functions, balance computational efficiency, and ensure stable training.

To identify the optimal set of hyperparameters, we used the open-source optimisation library `Optuna` (Akiba et al., 2019). Optuna combines efficient parameter space exploration with advanced pruning strategies, dynamically pruning unpromising trials to reduce computational costs while maximising performance. The final hyperparameters selected for the learning tasks in this study are:

- **Embedded dimension**: $d_h = 35$. The size of the hidden feature space for each node in the GNN. A higher dimension enables richer representations of node features, but excessively large values increase computational cost without necessarily improving performance.
- **Number of GNN layers**: $k = 40$. This depth ensures global information can propagate across the graph, while avoiding over-smoothing.
- **Update relaxation weight**: $\alpha = 6 \times 10^{-1}$. This parameter balances the influence of new and old information during feature updates.
- **Loss function weight**: $\gamma = 0.1$. Controls the weighting of loss terms from different layers in the GNN during training. Higher weights on later layers emphasise long-range dependencies in the training process.
- **Learning rate**: $LR = 3 \times 10^{-3}$. This optimises the step size for weight updates, balancing convergence speed and stability.

These hyperparameters were optimised to achieve a balance between model accuracy and computational efficiency. It is important to note that these hyperparameters are task-dependent and, in theory, should be re-optimised for each specific learning task to ensure optimal performance. However, re-optimisation for the different learning tasks in this study showed that the hyperparameters remained largely consistent. This suggests that the GNN architecture exhibits inherent robustness to variations in hyperparameters within the range of learning tasks addressed here.

Cite this article: Quattromini M, Bucci MA, Cherubini S and Semeraro O (2025). Mean flow data assimilation using physics-constrained graph neural networks. *Data-Centric Engineering*, 6, e48. doi:10.1017/dce.2025.10022

https://doi.org/10.1017/dce.2025.10022 Published online by Cambridge University Press