



HAL
open science

Enhancing Data-Assimilation in CFD using Graph Neural Networks

Michele Quattromini, Michele Alessandro Bucci, Stefania Cherubini, Onofrio Semeraro

► **To cite this version:**

Michele Quattromini, Michele Alessandro Bucci, Stefania Cherubini, Onofrio Semeraro. Enhancing Data-Assimilation in CFD using Graph Neural Networks. 37th Conference on Neural Information Processing Systems (Neurips 2023), Dec 2023, La Nouvelle Orléans (LA), United States. hal-04319697

HAL Id: hal-04319697

<https://hal.science/hal-04319697>

Submitted on 5 Dec 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ENHANCING DATA-ASSIMILATION IN CFD USING GRAPH NEURAL NETWORKS

A PREPRINT

Michele Quattromini

Department of Mechanics, Mathematics and Management, Polytechnical University of Bari, Italy, BA 70126
LISN-CNRS, Université Paris-Saclay Orsay, France, 91440
michele.quattromini@poliba.it

Michele Alessandro Bucci

Safran Tech, Digital Sciences & Technologies Magny-Les-Hameaux, France, 78114
michele-alessandro.bucci@safrangroup.com

Stefania Cherubini

Department of Mechanics, Mathematics and Management, Polytechnical University of Bari, Italy, BA 70126
stefania.cherubini@poliba.it

Onofrio Semeraro

LISN-CNRS, Université Paris-Saclay Orsay, France, 91440
onofrio.semeraro@universite-paris-saclay.fr

December 1, 2023

ABSTRACT

We present a novel machine learning approach for data assimilation applied in fluid mechanics, based on adjoint-optimization augmented by Graph Neural Networks (GNNs) models. We consider as baseline the Reynolds-Averaged Navier-Stokes (RANS) equations, where the unknown is the meanflow and a closure model based on the Reynolds-stress tensor is required for correctly computing the solution. An end-to-end process is cast; first, we train a GNN model for the closure term. Second, the GNN model is introduced in the training process of data assimilation, where the RANS equations act as a physics constraint for a consistent prediction. We obtain our results using direct numerical simulations based on a Finite Element Method (FEM) solver; a two-fold interface between the GNN model and the solver allows the GNN's predictions to be incorporated into post-processing steps of the FEM analysis. The proposed scheme provides an excellent reconstruction of the meanflow without any features selection; preliminary results show promising generalization properties over unseen flow configurations.

1 Introduction

Computational Fluid Dynamics (CFD) is a cornerstone for the simulation and analysis of fluid flows across a multitude of scientific and engineering fields. However, the computational complexity associated with solving the governing equations is a persistent challenge, as it requires high-performance computing resources and elaborate numerical techniques. The combination of Machine Learning (ML) techniques and CFD numerical solvers is emerging as a possible solution to overcome these limits (see reviews by [Duraismy et al. \[2019\]](#) and [Brunton et al. \[2020\]](#)).

In this paper, we consider a data-assimilation problem augmented by a Graph Neural Network (GNN) model. In particular, we aim at computing the meanflow based on known measurements. Standard approaches rely on data-assimilation via adjoint-based optimization [[Foures et al., 2014](#)]. Here, we introduce an end-to-end assimilation process, that includes a GNN model capable of predicting the closure term for the RANS equations, namely the Reynolds stress, following the works by [Donon et al. \[2020\]](#) and [Ströfer and Xiao \[2021\]](#). The choice of GNN models is justified by the strong interest for this architecture in the fluid mechanics community [[Lino et al., 2023](#)]. Indeed, GNN

allows to address some of the limitations often found when integrating Deep Neural Networks (DNN) and numerical solvers for physics. First, data hungriness is often a limiting factor when training a DNN, as from an industrial point of view the availability of data can be limited. GNN are known to be more data-parsimonious [Hamilton, 2020]. Second, some DNN architectures combined with numerical solvers lack of flexibility when considering CFD simulations in complex geometries. An example is given by standard Convolutional Neural Networks (CNN), often used as approximators of closure terms, known to face challenges with unstructured meshes. Although very recent works address the latter limitation in CNN [Coscia et al., 2023], successful works are already available relying on GNN, for instance, for the up-scaling of low-resolution simulations [Belbute-Peres et al., 2020].

In the following, we briefly introduce our data-assimilation scheme and consider the dynamics of the wake past a cylinder at low Reynolds number, $Re = 120$, as a test case. The flow exhibits the von Karman Street instability and is simulated by means of Direct Numerical Simulations (DNS) using a Finite Element Method (FEM) solver. The GNN model is pre-trained on a dataset composed by a single pair of meanflow (input) and Reynolds stress (output) of the wake past cylinder bluff body on unstructured mesh at $Re = 120$. Then, the training iteration proceeds by enforcing the RANS equations in the post-training loop. We show that such a model improves the learning path by effectively assimilating data under the constraint provided by the RANS equations.

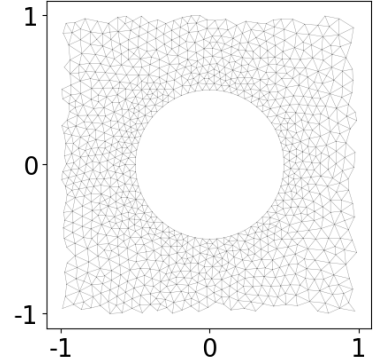


Figure 1: Unstructured mesh around a 2D cylindrical body.

2 Methodology

2.1 Baseline equations and numerical simulations

We consider incompressible, two-dimensional (2D) fluid flows developing past bluff bodies. We focus on time-averaged quantities and on second order-statistics, the Reynolds stress. To this end, we introduce the Reynolds decomposition

$$\mathbf{u}(\mathbf{x}, t) = \bar{\mathbf{u}}(\mathbf{x}) + \mathbf{u}'(\mathbf{x}, t), \quad (1)$$

where $\bar{\mathbf{u}} = (\bar{u}, \bar{v})^T$ is the meanflow, \mathbf{u}' the fluctuation field, and $\mathbf{x} = (x, y)^T$ is the vector composed by the streamwise direction x and the wall-normal direction y . Formally, any unsteady flow can be described using this decomposition, whether we consider a laminar case or turbulent one [Foures et al., 2014]. Plugging Eq. (1) in the Navier–Stokes equations, after time-averaging we get

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

$$\nabla \cdot \bar{\mathbf{u}} = 0, \quad (2b)$$

where \bar{p} is the average pressure field. The Reynolds number is defined as $Re = U_\infty D / \nu$, with the reference velocity chosen as the free stream velocity U_∞ at the inlet, D the reference length of the flow and ν the kinematic viscosity. Mathematically, the resulting system provides the Reynolds-averaged Navier–Stokes Equations (RANS) and the forcing \mathbf{f} is the closure term or Reynolds stress. This term can be modeled or – when data are available – directly computed as

$$\mathbf{f} = -\overline{\mathbf{u}' \cdot \nabla \mathbf{u}'}. \quad (3)$$

In this article, the numerical solution of the Eq. 2 is performed using a FEM solver based on FEniCS [Alnæs et al., 2015]. Temporal discretization is implemented using a second-order Backward Differentiation Formula (BDF). FEM solutions are computed on unstructured mesh, thus allowing for a great versatility in handling complex geometries (see Fig. 1). The computational domain is discretized using 13283 nodes and has dimensions $[L_x, L_y] = [27, 10]$.

2.2 Data assimilation problem

A common problem often found in fluid mechanics is to reconstruct a flow field or a meanflow starting from local measurements. The problem can be cast as an optimization process of data-assimilation, where the Eq. 2 can be used as baseline and the forcing term \mathbf{f} is the control parameter to be determined. The cost function to be minimized can be written as

$$J(\hat{\mathbf{u}}) = \frac{1}{2} \|(\bar{\mathbf{u}} - \hat{\mathbf{u}})\|^2, \quad (4)$$

where $\bar{\mathbf{u}}$ is a known field and $\hat{\mathbf{u}}$ is the solution of the RANS equations based on $\hat{\mathbf{f}}$. The variable $\hat{\mathbf{f}}$ is the iterative solution of an adjoint-based optimization, based on the procedure introduced in the paper by Foures et al. [2014]

and approximating the term \mathbf{f} . The solution is iteratively updated until convergence using the gradient $\partial J/\partial \hat{\mathbf{f}}$, which corresponds to the adjoint state. Here, this process is augmented through an additional GNN model that provides the closure term.

2.3 Augmented training process

The training process is inspired by the Deep Statistical Solver framework introduced by Donon et al. [2020], and the data assimilation scheme by Ströfer and Xiao [2021], both developed as end-to-end learning cycles. Our algorithm is sketched in Fig. 2, where the *forward step* path and the *backward step* are represented; θ are the GNN’s trainable parameters. Following the Stochastic Gradient Descent (SGD) method and employing the gradients chain-rule, the gradient of the cost function J with respect to θ is

$$\frac{\partial J}{\partial \theta} = \frac{\partial J}{\partial \hat{\mathbf{u}}} \cdot \frac{\partial \hat{\mathbf{u}}}{\partial \hat{\mathbf{f}}} \cdot \frac{\partial \hat{\mathbf{f}}}{\partial \theta} = \frac{\partial J}{\partial \hat{\mathbf{f}}} \cdot \frac{\partial \hat{\mathbf{f}}}{\partial \theta}. \quad (5)$$

The novelty of this paper lies in the combination of these gradients from diverse computational strategies, achieved through the use of our GNN-FEM interface. In particular, the term $\partial J/\partial \hat{\mathbf{f}}$ comes from the adjoint formulation of the RANS equations and is computed using the FEM simulation; $\partial \hat{\mathbf{f}}/\partial \theta$ is obtained from the Automatic Differentiation (AD) package provided by the Torch python library. To compute the chain-rule as stated in Eq. 5, these terms from different approaches need to be accumulated and back-propagated all together through the entire GNN model. This operation is made possible by the two-folded interface that can convert a FEM discretized vectorial field from FEniCS in a numerical tensor to be used by Torch and vice-versa, without any loss of information. In the following we will denote the data assimilation scheme as DA-GNN.

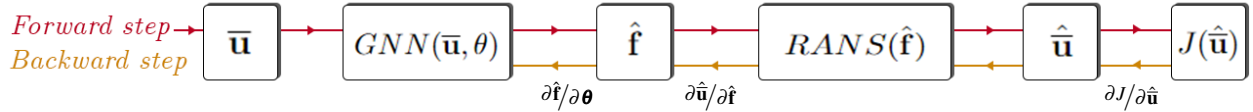


Figure 2: End-to-end training loop; $\bar{\mathbf{u}}$ is the GNN’s input meanflow; θ are the GNN’s trainable parameter; $\hat{\mathbf{f}}$ is the GNN’s predicted forcing; $J(\hat{\mathbf{u}})$ is the cost function as expressed in Eq. 4.

2.4 GNN architecture description

A custom GNN architecture has been designed to operate on the unstructured meshes. In the initial phase of the *forward step*, each node within the mesh is assigned with an embedded state, denoted as \mathbf{h}_i for each node i . The state is initialized to a zero vector. The architecture employs two dedicated Multi-Layer Perceptrons (MLPs) to perform the message-passing mechanism [Hamilton, 2020], one for the in-going messages to the node and one for the out-going information from the node itself. These MLPs extract and integrate the spatial flow information from the neighboring nodes and, together with the embedded state of the node itself of the current iteration, they are processed via a third MLP to provide an update of the embedded state at each node.

The operation reads as

$$\mathbf{h}_i^{(k)} = \mathbf{h}_i^{(k-1)} + \alpha \Psi^{(k)} \left(\mathbf{h}_i^{(k-1)}, \{\bar{\mathbf{u}}, Re\}, \phi_{\rightarrow,i}^{(k)}, \phi_{\leftarrow,i}^{(k)}, \phi_{\circ,i}^{(k)} \right), \quad (6)$$

where $\phi_{\rightarrow,i}^{(k)} = \frac{1}{N_d} \sum_{j=1}^{N_d} \phi_{j,i}^{(k)}$ is the message that the node i sends to its N_d neighbours, averaged over the number of neighbours; $\phi_{\leftarrow,i}^{(k)} = \frac{1}{N_d} \sum_{j=1}^{N_d} \phi_{i,j}^{(k)}$ is the message that the node i receives from its N_d neighbours, averaged over the number of neighbours; $\phi_{\circ,i}^{(k)} = \phi_{i,i}^{(k)}$ the message that the node i sends to itself to avoid loss of information as the process advances. $\Psi^{(k)}$ is a generic differentiable operator that can be approximated also in this case by MLP; α is a relaxation coefficient that allows to scale each update of the embedded states with the previous one during the message passing loop. At the end of the process, the last embedded state is projected back to a physical state using a decoder, a fully connected MLP, to obtain the closure term prediction of the GNN. We adopted an automatic hyper-parameters landscape explorer, the python library Optuna, for tuning the 5 hyper-parameters of the GNN model: the dimension of the embedded state on each node; the number k of GNN layers; the relaxation coefficient α used for the updates; the learning rate and γ , a coefficient used to weight the contribution to the cost function by each layer of the GNN.

3 Results

We consider as a test case the reconstruction of the meanflow of a cylinder wake at $Re = 120$. In Fig. 3, the training loss curve is shown. The training process consists of two phases; the first 1,000 epochs are based on a supervised learning of the forcing terms. This initial phase is necessary to provide a starting forcing term for the subsequent phase that allows the RANS solver to converge, addressing FEM numerical stability problems. Therefore, the cost function is designed to minimize the difference between the predicted forcing terms and the actual ones trained over a dictionary of synthetic data obtained for different bluff body shapes [Quattromini et al., 2023]. Then, we move to the full end-to-end training process for another 2,500 epochs, where the cost function J in Eq. 4 evolves, comparing the predicted meanflow $\hat{\mathbf{u}}$ with ground truth from DNS. The global training loss shows that the cost function is consistently decreasing, assuring

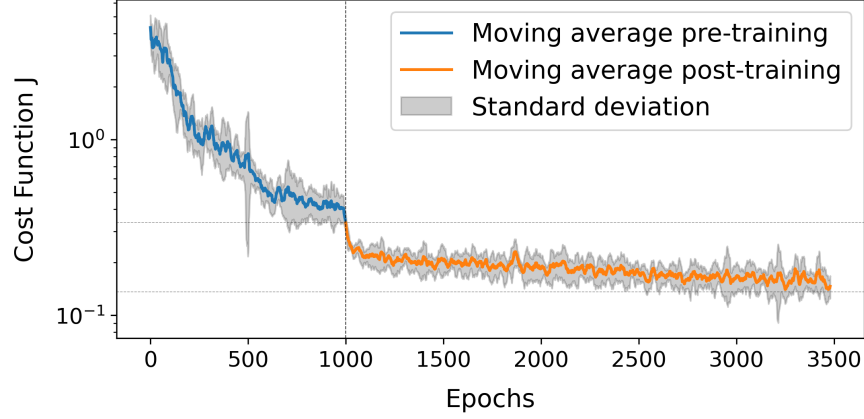


Figure 3: Training loss curve based on the cost function in Eq. 4. The blue line (pre-training) and the orange line (post-training) are the moving average of the mean curve from 5 different GNN models, with different random seeds. The grey area shows the standard deviation around the mean curve.

that the gradients are correctly computed and back-propagated as the model’s predictions are continuously refined. The RANS equations are thus correctly enforced in the training process and the prediction of the DA-GNN holds now physical consistency with the CFD equations. Fig. 4(a-d) shows the predicted $\hat{\mathbf{u}}$ and $\hat{\mathbf{f}}$ as compared to the ground truth for the presented cylinder test case, while Fig. 4(e-h) shows the generalization capabilities on a random shape bluff body at $Re = 130$, unseen during the training process.

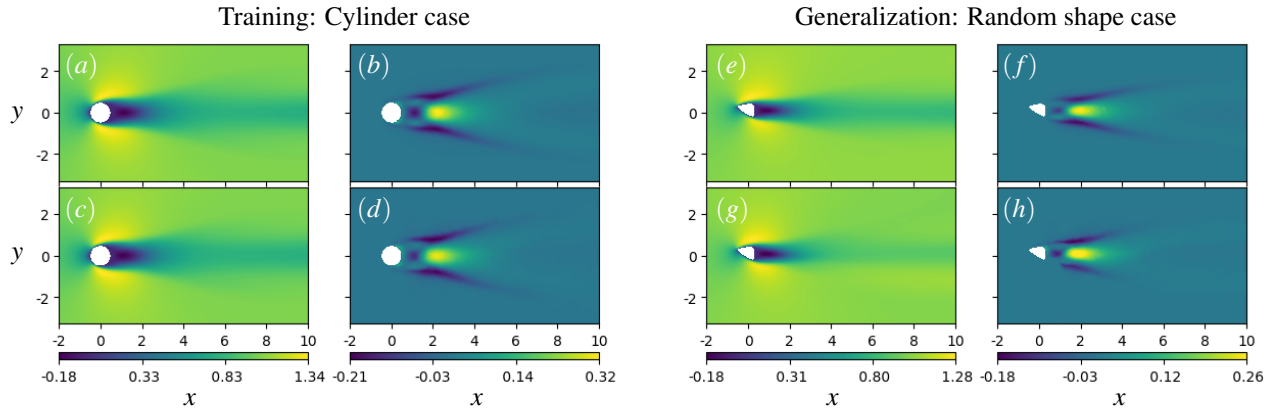


Figure 4: Comparison of the streamwise component between DNS results and DA-GNN predictions; (a, e) $\bar{\mathbf{u}}$ from DNS; (c, g) $\hat{\mathbf{u}}$ from DA-GNN; (b, f) \mathbf{f} from DNS; (d, h) $\hat{\mathbf{f}}$ from DA-GNN;

4 Conclusions

In this paper, we introduced a novel algorithm of data assimilation combining the versatility of GNN with a FEM solver. An interface between the GNN model and the FEM solver enables the communication between the two computational environments for a robust back-propagation of gradients during the training phase. The resulting data-assimilation scheme provides an appropriate reconstruction of the meanflow without any features selection, relying on a pre-trained GNN model and keeping physical coherence of the prediction through the constraint provided by the RANS equations. We are currently moving to corrupted or incomplete meanflow reconstruction and refining the technique for enhancing the generalization properties over unseen cases, including different geometries and flow parameters.

Acknowledgements. The Ph.D. fellowship of M. Quattromini is supported by the Italian Ministry of University. A part of the research was funded by the grant PRIN2017-LUBRI-SMOOTH of the Italian Ministry of Research and ANR-21-REASON from the French Agency for National Research.

References

- K. Duraisamy, G. Iaccarino, and H. Xiao. Turbulence modeling in the age of data. *Annu. Rev. Fluid Mech.*, 51: 357–377, 2019.
- S. L. Brunton, B. R. Noack, and P. Koumoutsakos. Machine learning for fluid mechanics. *Annu. Rev. Fluid Mech.*, 52: 477–508, 2020.
- D. P. G. Foures, N. Dovetta, D. Sipp, and P. J. Schmid. A data-assimilation method for Reynolds-averaged Navier–Stokes-driven mean flow reconstruction. *J. Fluid Mech.*, 759:404–431, 2014.
- B. Donon, Z. Liu, W. Liu, I. Guyon, A. Marot, and M. Schoenauer. Deep Statistical Solvers. In *NeurIPS*, Vancouver, Canada, 2020.
- C. A. Ströfer and H. Xiao. End-to-end differentiable learning of turbulence models from indirect observations. *Theor. App. Mech.*, 11(4):100280, 2021.
- M. Lino, S. Fotiadis, A.A. Bharath, and C.D. Cantwell. Current and emerging deep-learning methods for the simulation of fluid dynamics. *Proc. R. Soc. A*, 479(2275):20230058, 2023.
- W. L. Hamilton. *Graph representation learning*. Morgan & Claypool Publishers, 2020.
- D. Coscia, L. Meneghetti, N. Demo, G. Stabile, and G. Rozza. A continuous convolutional trainable filter for modelling unstructured data. *Comput. Mech.*, 72(2):253–265, 2023.
- F. A. Belbute-Peres, T. Economou, and Z. Kolter. Combining differentiable PDE solvers and graph neural networks for fluid flow prediction. In *ICML*, pages 2402–2411. PMLR, 2020.
- M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E Rognes, and G.N. Wells. The FEniCS project version 1.5. *Archive of numerical software*, 3(100), 2015.
- M. Quattromini, M. A. Bucci, S. Cherubini, and O. Semeraro. Operator learning of rans equations: a graph neural network closure model. *arXiv:2303.03806*, 2023.