Scuola IMT Alti Studi Lucca

A hybrid reduced-order model for segregated fluid-structure interaction solvers in an ALE approach at high Reynolds number

Questa è la versione preprint della seguente opera:

Original
A hybrid reduced-order model for segregated fluid-structure interaction solvers in an ALE approach at high Reynolds number / Nkana Ngan, V., Stabile, G., Mola, A., Rozza, G.. - In: COMPUTERS & MATHEMATICS WITH APPLICATIONS. - ISSN 0898-1221. - 180:(2025), pp. 299-321. [10.1016/j.camwa.2025.01.004]

Availability:
This version is available at: 20.500.11771/37079

Publisher:

Published
DOI:10.1016/j.camwa.2025.01.004

Terms of use:

This publication is made accessible in accordance with the terms for deposit in the institutional repository, as defined by the IMT School for Advanced Studies Lucca's Open Access Policy. (https://library.imtlucca.it/sites/default/files/regolamento-policy-open-access-imtlib_0.pdf).

Si prega di consultare le pagine informative dell'editore relative alle politiche di autoarchiviazione.

(Article begins on next page)

14 September 2026

arXiv:2406.12701v1 [physics.flu-dyn] 18 Jun 2024

Contents

1 Motivation and state-of-the-art **2**

2 Governing equations of the fluid-structure interaction problem and turbulence modeling **4**
2.1 Structural dynamics equations . . . 4
2.2 Fluid dynamics equations and turbulence modeling . . . 5
2.3 Coupling conditions at the interface . . . 5
2.4 Mesh motion strategy . . . 5

3 Numerical Methodology **6**
3.1 Numerical discretization of the full-order model . . . 6
3.1.1 The pressure gradient term . . . 6
3.1.2 The convective term . . . 6
3.1.3 The diffusion term . . . 7
3.1.4 The PIMPLE algorithm . . . 7
3.2 The reduced problem . . . 9
3.2.1 The proper orthogonal decomposition . . . 9
3.2.2 POD-Galerkin projection for velocity and pressure equations . . . 11
3.2.3 Machine learning for eddy viscosity prediction . . . 11
3.2.4 POD with interpolation for mesh motion prediction . . . 13

4 Definition of test case, simulation results, and discussion **14**
4.1 Definition of the test case . . . 15
4.2 Simulation results . . . 15
4.2.1 Prediction quality . . . 16
4.2.2 Machine learning of the temporal eddy viscosity coefficients . . . 17
4.2.3 ROM online resolution time . . . 17
4.2.4 ROM online resolution quality . . . 18

5 Conclusions and perspectives **24**

A Appendix **26**
A.1 Machine learning of the temporal eddy viscosity coefficients . . . 26

A hybrid reduced-order model for segregated fluid-structure interaction solvers in an ALE approach at high Reynolds number

Valentin Nkana Ngan*[1], Giovanni Stabile†[2], Andrea Mola‡[3], and Gianluigi Rozza§[1]

[1]Mathematics Area, mathLab, SISSA, via Bonomea 265, I-34136 Trieste, Italy
[2]The Biorobotics Institute, Sant'Anna School of Advacended Studies, V.le R. Piaggio 34, 56025, Pontedera, Pisa - Italy
[3]MUSAM Continuum Mechanics Laboratory, Scuola IMT Alti Studi Lucca - Piazza S. Ponziano, 6 - 55100 Lucca, LU, Italy

October 20, 2024

Abstract

This study introduces a first step for constructing a hybrid reduced-order models (ROMs) for segregated fluid-structure interaction in an Arbitrary Lagrangian-Eulerian (ALE) approach at a high Reynolds number using the Finite Volume Method (FVM). The ROM is driven by proper orthogonal decomposition (POD) with hybrid techniques that combines the classical Galerkin projection and two data-driven methods (radial basis networks , and neural networks/ long short term memory). Results demonstrate the ROM's ability to accurately capture the physics of fluid-structure interaction phenomena. This approach is validated through a case study focusing on flow-induced vibration (FIV) of a pitch-plunge airfoil at a high Reynolds number ($Re = 10^7$).

Keywords: Fluid-structure interaction, reduced-order model, Finite Volume Method, Proper orthogonal decomposition, Galerkin projection, radial basis network, mesh motion, turbulence, long-short term memory, neural networks, and flow-induced vibration.

1 Motivation and state-of-the-art

Interactions between fluid and moving boundaries are among the most essential issues of fluid dynamics. They arise in wind turbines, civil engineering (e.g., the influence of wind on bridges and buildings), and the aerospace industry [59]. In the last case, fluid-structure interaction (FSI) plays an essential role in the design process of an aircraft. A significant amount of effort has been put into understanding and solving problems related to FSI using computational fluid dynamics (CFD). CFD has become an essential tool in the analysis and design of aerospace vehicles thanks to the advances in both computational algorithms and hardware. With these advances in CFD, the number of wings tested experimentally in the design of a typical commercial aircraft has decreased by an order of magnitude from the last four decades so an effective use of CFD is a key ingredient in the successful design of modern commercial aircraft [27]. Nowadays, CFD tools can accurately predict aircraft aerodynamics in cruise conditions and complement wind tunnel and flight tests in the aircraft design process. However, traditional high-fidelity aerodynamics simulations can be computationally too expensive for scenarios requiring real-time responses (e.g., flow control) and/or predictions for many different configurations (e.g., design-space exploration and flight-parameter sweep). The design of a new aircraft requires an analysis of a huge number of variants. One has to check different aircraft configurations, mass cases, gusts, and maneuvers giving (even

*vkanang@sissa.it
†giovanni.stabile@santannapisa.it
‡andrea.mola@imtlucca.it
§grozza@sissa.it

with engineering experience for current configurations and technologies) hundreds of thousands of simulations [50].

Reduced-order models (ROMs) come into play to accelerate the solution of unsteady and/or parameterized aerodynamics problems in real-time and/or many-query scenarios [3]. The ROM approach in general and particularly based on the Proper Orthogonal Decomposition (POD) method has been extensively adopted for various engineering applications [8]. "ROMs are nowadays probably the most important mathematical technique for realizing a digital twin during the life cycle of a product. By using them, simulation models can be more interactive, reliable, continuous, accessible, and distributable" [28] i.e. ROM can be seen as a key enabler for a new generation of digital twins. "As a concrete example, one could consider the typical ROM problem: given a dynamical system, find a reduced dynamical system that approximates the original system with a controllable trade-off between error and speed, and preservation of key properties like stability."

The model-order reduction literature for aerodynamics problems is growing, with contributions from both engineering and applied mathematics communities. The study of Anttonen et al. [1, 2] outlined that an additional difficulty is reached when considering aeroelastic non-linear dynamical systems. In FSI cases, especially when adopting high-fidelity aerodynamics, some theoretical limits arise from the POD theory formulated in the deformable domain. The POD is based on a definition of a spatial correlation of the system. However, the snapshots resulting from an aeroelastic CFD solver cannot ensure this point as, notoriously, the mesh is moving and deforming during the simulation. The loss of the spatial correlation and the increasingly important stability and accuracy problems make the aeroelastic ROM study widely challenging. In their numerical examples involving a moving airfoil, Freno et al. [14] showed that when an index-based domain is used to build the ROM, similar to the one considered by Anttonen et al. [2], numerical simulations do not suffer from the mesh deformation limitation discussed above. In the case of a rigid body motion as considered in this study, an interesting manner to avoid issues associated to mesh deformation are presented by Lewin et al. [33] and Placzek [48]. They performed the projection of the governing equations in a non-inertial reference frame to preserve the POD formulation's consistency. However, in their case, stability problems appear when considering highly non-linear flows. Troshin et al. [62] outlined an alternative POD methodology for a flow field in a domain with moving boundaries. The moving domain is mapped to a stationary domain by combining a transfinite interpolation and an algorithm for volume adjustment. Liberge et al. [34] implemented a multi-phase method that allows the performance of the POD on a moving domain using characteristic functions to follow the fluid-structure interface. Falaize et al. [12] extended such formulation for flows induced by rigid bodies in forced rotation. Also, they included parametric changes in the proposed model. Longatte et al. [35] explored the behavior of POD-multiphase ROM presented in [34] when the parameter values are different from those used to build the POD basis. Stankiewicz et al. [58, 60] deepen the study of Anttonen et al. [2] with test cases of increasing complexity also considering parametric changes. Freno et al. [13, 14] dealt with general non-linear systems in an aeroelastic context. They used dynamics basis functions to consider the domain deformation dynamics by defining dynamics related to the instantaneous deformed configuration for the projection basis. In addition, they considered a fully non-linear system, but since it is impossible to make it explicit, they used the FOM to evaluate the non-linear term at each time step. As a result, the computational efficiency of the ROM is reduced significantly. Also, Shinde et al . [53] extended the Galerkin-free approach to FSI problems by interpolating POD basis including mesh deformations. Alternatively, Thomas et al. [61] developed a nonlinear ROM for a dynamically nonlinear solver for limit cycle oscillation analysis. They used a nonlinear frequency domain harmonic balance method [18] in conjunction with a Taylor series expansion and POD to create a frequency-domain nonlinear ROM. In the context of system identification approaches, Chen et al. [7] developed a support vector machine (SVM) based ROM for predicting the limit cycle oscillation induced by the nonlinear aerodynamics of an aeroelastic system. Mannarino et al. [36, 37] developed a recurrent neural networks-based ROM technique in the discrete-time domain to deal with nonlinearities in FSI problems. Kou et al. [31] derived a ROM for the investigation of limit cycle oscillations and flutter behaviors of an airfoil by combining linear auto-regressive with exogenous input (ARX) model with radial basis neural networks (RBFNNs) model for the nonlinear approximation. The current work is aligned in the same directions as the following studies [9, 17, 39–41, 66]. This work differs from the works mentioned above by considering the motion of the mesh in the Arbitrary Lagrangian-Eulerian (ALE) sense. Moreover, it proposes an intrusive hybrid approach where the main partial differential equations (PDEs) are treated using a standard POD-Galerkin projection approach and radial

basis functions networks for the grid motion interpolation. More importantly, the current research expands upon our previous work done in [43], by providing two data-driven approaches, for predicting the eddy viscosity at the reduced-order level. This recipe results in a hybrid data-driven reduced-order model for FSI for any segregated solvers in the Finite Volumes Method (FVM). The proposed recipe combines the strengths of the POD-based reduced-order modelling and machine learning. In fact, by incorporating data-driven techniques, the ROM could achieve high accuracy and efficiency, making it a powerful tool for simulating complex FSI problems which could be useful in industrial applications in the development of digital twins systems.

The structure of the manuscript is as follows: The section 2 begins with the formulation of the fluid-structure interaction with turbulence modeling. The section contains four subsections. The first Subsection 2.1 presents the structure motion. The following Subsection 2.2 deals with the mathematical formulation of the fluid's motion in the ALE setting, followed by the coupling strategy at the interface in Subsection 2.3. The section 3 presents the methodology carried out in this study. The section is divided into five subsections. The Subsection 3.1 addresses the numerical discretization of the full-order model. The Subsection 3.2 discusses the reduced-order model concept followed by the discussion of POD for turbulence incompressible flows in Subsection 3.2.2 by insisting on its main properties in terms of model reduction while Subsection 3.2.3 discusses the machine learning algorithms for predicting the eddy viscosity. In the last Subsection 3.2.4, the POD-RBF for interpolating the point cloud motion is introduced. The section 4 presents and discusses the numerical results obtained in this study. Finally, a few considerations and possible avenues for future developments for this study are presented in section 5.

2 Governing equations of the fluid-structure interaction problem and turbulence modeling

This section presents the mathematical formulation of the FSI model. The following assumptions are considered:

- The fluid is viscous, incompressible and Newtonian;
- The geometry and flow field considered are two-dimensional;
- The airfoil structure is rigid;
- The elastic connection between the airfoil and the ground is represented by a set of linear and angular springs and dampers;
- The airfoil undergoes a free translation (vertical direction) and rotational pitch motion.

Therefore, based on the aforementioned assumptions, the present part briefly describes the high-dimensional full-order model to simulate the coupled fluid-body interaction using the Navier-Stokes equations in the ALE setting, considering the rigid body dynamics and the coupling conditions at the interface.

2.1 Structural dynamics equations

In this work, the aeroelastic structural model as depicted in fig. 1 is governed by a two-degree freedom pitch and plunge system. The equations of the structure's motion are:

$$m\ddot{h} + c_h\dot{h} + k_h h - mb\ddot{\theta}\cos\theta + mb\dot{\theta}^2\sin\theta = F_h(t). \tag{1}$$

$$\boldsymbol{I}_\theta\ddot{\theta} + c_\theta\dot{\theta} + k_\theta\theta - mb\ddot{h}\cos\theta = \boldsymbol{M}_\theta(t). \tag{2}$$

m being the mass of the airfoil per unit span, $F_h(t)$ the sectional lift per unit span, $\boldsymbol{I}_\theta$ the sectional moment of inertia of the airfoil, $\boldsymbol{M}_\theta(t)$ is the pitching moment, $\theta(t)$ the pitch rotation, h(t) the plunge displacement, b the distance between the pivot location and the center of mass. The structural stiffness of the plunge and pitch is designated by k_h and k_θ; the related damping coefficients are c_h and c_θ. The structural frequencies are $f_\theta = (2\pi)^{-1}\omega_\theta$, and $f_h = (2\pi)^{-1}\omega_h$ with $\omega_\theta = \sqrt{k_\theta/I_{zz}}$ and $\omega_h = \sqrt{k_h/m}$. I_{zz} being the moment of inertia $\boldsymbol{I}_\theta$ in the z-direction. For a thorough introduction to structural dynamics and aero-elasticity refer to [21].

2.2 Fluid dynamics equations and turbulence modeling

The unsteady Reynolds Average Navier-Stokes Equations in the ALE framework for a *Newtonian fluid* are written as follows:

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

$$\frac{\delta \bar{\boldsymbol{u}}}{\delta t} + \nabla \cdot (\bar{\boldsymbol{u}} \otimes (\bar{\boldsymbol{u}} - \boldsymbol{u}^g)) = \frac{1}{\rho}\nabla \cdot [(\nu + \nu_t)\nabla \bar{\boldsymbol{u}})] - \frac{1}{\rho}\nabla \bar{p}. \tag{4}$$

$$\frac{\delta \Omega(t)}{\delta t} + \nabla \cdot \boldsymbol{u}^g = 0. \tag{5}$$

$\bar{\boldsymbol{u}}$ is the average velocity field, $\bar{p}$ the average pressure field, and $\boldsymbol{u}^g$ the grid velocity. With ν being the kinematic viscosity and ν_t is the so-called turbulent viscosity. The quantity ν_t is suitably a viscosity only from the dimensional point of view and it is called viscosity considering the analogy of the Boussineq approximation and with the shear stress relations in a Newtonian fluid. The molecular viscosity is a property of the fluid and not its motion. A variety of methodologies are available in the literature to solve the eddy viscosity. This work uses the k-ω SST (shear stress transport) introduced in [38]. The time derivative in the ALE framework is given by:

$$\frac{\delta}{\delta t} = \frac{\partial}{\partial t} + \boldsymbol{u}^g \nabla. \tag{6}$$

The grid which moves in space must also obey the conservation law [63], which is stated as "the change in volume (area) of each control volume between time t^n and t^{n+1} must equal the volume (area) swept by the cell's boundary during $\Delta t = t^{n+1} - t^n$" which may be expressed as:

$$\frac{\delta}{\delta t}\int_{\Omega_i} d\Omega_i + \int_{S_i} \boldsymbol{u}^g \cdot \boldsymbol{n} dS_i = 0 \quad \equiv \quad \frac{\delta \Omega_i}{\delta t} + \nabla \cdot \boldsymbol{u}^g = 0, \tag{7}$$

for every control volume $\Omega_i = \Omega_i(t)$. $\boldsymbol{n}$ being the outward unit normal vector on the boundary surface, and $S_i = \partial\Omega_i$. By multiplying eq. (7) by ρ and using the incompressibility constraint leads to:

$$\int_{\partial\Omega_i} \boldsymbol{u}^g \cdot \boldsymbol{n} dS_i = 0. \tag{8}$$

This means there is no need to consider the grid velocity in the continuity equation. Additionally, there are *initial and boundaries conditions*. This work uses URANS as it is the workhorse of turbulence modeling in industrial applications [20].

2.3 Coupling conditions at the interface

The coupling between fluid and structure is achieved at the boundary conditions on the common interface $\Gamma(t)$ which all stem from simple physical principles: *kinematic condition* (the fluid velocity, gird velocity,and structure's velocity are continuous at the interface), *dynamic condition* (the normal stresses of the fluid and structure are continuous on the interface), and the *geometric condition* (the fluid and structure domain should always match.

$$\boldsymbol{u} \cdot \boldsymbol{e}_y = \boldsymbol{u}_g \cdot \boldsymbol{e}_y = \dot{h} \quad \text{and} \quad \int_{\Gamma(t)} (\boldsymbol{\sigma}(\boldsymbol{x}, t) \cdot \boldsymbol{n}) \cdot \boldsymbol{n}_y d\Gamma + F_h(t) = 0, \tag{9}$$

with $\boldsymbol{\sigma}(\boldsymbol{x}, t) = -p(\boldsymbol{x}, t)\mathbf{I} + \mu\left(\nabla \cdot \boldsymbol{u}(\boldsymbol{x}, t) + (\nabla \cdot \boldsymbol{u}(\boldsymbol{x}, t))^T\right)$ as the fluid is assumed to be Newtonian. $F_h(t)$ is the time dependent lift force.

2.4 Mesh motion strategy

Translation and rotational motion of the centre of gravity (COG) are accounted by solving Newton's second law eqs. (1) and (2) in the global inertial reference frame. After the linear and angular accelerations have been computed using eqs. (1) and (2), translation and rotational kinematics are

used to update the body linear and angular velocities. After calculating the motion of the rigid body it is necessary to move the boundary as well as the mesh surrounding the body in order to maintain a good quality mesh. In this work, the mesh deformation technique used is the so-called Slerp (Spherical Linear Interpolation) as it is better at handling translation and rotational (of a solid body in the three arises XYZ) mesh deformation when it comes to cell shearing [26] and has great applications in computer vision.

3 Numerical Methodology

The present section opens with a brief outline of the details of the standard FVM used for the FOM discretization. Relevant details of the reduced model are then presented, along with a description of the projection algorithm for the fully discrete equations. Finally, it describes Machine Learning algorithms (neural networks and recurrent neural networks) used for the online computation of the eddy viscosity, and of the radial basis interpolation combined with proper orthogonal decomposition used for the online computation of the grid nodal displacement field.

3.1 Numerical discretization of the full-order model

The standard FVM aims to discretize the system of partial differential equations written in integral form following [42]. The present uses a 2-dimensional tessellation. N_h represents the dimension of the full-order model (FOM) which is the number of control volumes in the discretized problem. The following addresses the discretization methodology of the momentum and continuity equations. In particular, a segregated approach is used to solve the momentum and continuity equations inspired by Rhie-Chow interpolation [49].

The referenced domain $\Omega(t)$ is divided into a tessellation $\mathcal{T}(t) = \{\Omega_i(t)\}_{i=1}^{N_h}$ so that every cell $\Omega_i(t)$ is a non-convex polygon and $\bigcup_{i=1}^{N_h} \Omega_i(t) = \Omega(t)$ and $\Omega_i(t) \cap \Omega_j(t) = \emptyset \quad \forall i \neq j$. In the following, to simplify the notation $\Omega_i = \Omega_i(t)$ and $S_i = \partial\Omega_i(t)$. S_i being the total surface related to cell Ω_i. The unsteady-state momentum equation written in its integral form for every cell of the tessellation reads as follows:

$$\int_{\Omega_i} \frac{\delta \bar{\boldsymbol{u}}}{\delta t} d\Omega_i + \int_{\Omega_i} \nabla \cdot [\bar{\boldsymbol{u}} \otimes (\bar{\boldsymbol{u}} - \boldsymbol{u}^g)] d\Omega_i - \int_{\Omega_i} \nu_{eff} \nabla^2 \bar{\boldsymbol{u}} d\Omega_i + \int_{\Omega_i} \nabla \bar{p} d\Omega_i = 0. \tag{10}$$

$\nu_{eff} = \nu_l + \nu_t$ being the effective viscosity and the sum of the ν_l (molecular viscosity) and turbulent viscosity. In the sequel, the full-order model is analyzed term by term.

3.1.1 The pressure gradient term

The pressure gradient term is discretized using Gauss's theorem.

$$\int_{\Omega_i} \nabla \bar{p} d\Omega_i = \int_{S_i} \bar{p} d\boldsymbol{S} \approx \sum_j \boldsymbol{S}_{ij} \bar{p}_{ij}, \tag{11}$$

where $\boldsymbol{S}_{ij}$ is the oriented surface dividing the two neighbor cells Ω_i and Ω_j and $\bar{p}_{ij}$ is the pressure evaluated at the center of the face $\boldsymbol{S}_{ij}$.

3.1.2 The convective term

The convective term can be discretized as follows using Gauss's theorem.

$$\int_{\Omega_i} \nabla \cdot [\bar{\boldsymbol{u}} \otimes (\bar{\boldsymbol{u}} - \boldsymbol{u}^g)] d\Omega_i = \int_{\Omega_i} \nabla \cdot (\bar{\boldsymbol{u}} \otimes \bar{\boldsymbol{u}}) d\Omega_i - \int_{\Omega_i} \nabla \cdot (\bar{\boldsymbol{u}} \otimes \boldsymbol{u}^g) d\Omega_i \tag{12}$$

$$= \int_{S_i} d\boldsymbol{S} \cdot (\bar{\boldsymbol{u}} \otimes \bar{\boldsymbol{u}}) - \int_{S_i} d\boldsymbol{S} \cdot (\bar{\boldsymbol{u}} \otimes \boldsymbol{u}^g) \tag{13}$$

$$= \sum_{j \in S_i} \bar{\boldsymbol{u}}_{ij} \boldsymbol{F}_{ij} - \sum_{j \in S_i} \bar{\boldsymbol{u}}_{ij} ({\boldsymbol{u}^g}_{ij} \cdot \boldsymbol{S}_{ij}). \tag{14}$$

Here, $\bar{\boldsymbol{u}}_{ij}$ is the velocity evaluated at the center of the face $\boldsymbol{S}_{ij}$, and $\boldsymbol{F}_{ij} = \bar{\boldsymbol{u}}_{ij} \cdot \boldsymbol{S}_{ij}$ is the flux of the velocity through the face $\boldsymbol{S}_{ij}$. This procedure underlines two considerations. The first one is that $\bar{\boldsymbol{u}}_{ij}$ is not straightly available in the sense that all the variables of the problem are evaluated at the center of the cells. At the same time, the velocity is evaluated at the center of the face. Many different techniques are available to obtain it. However, the basic idea behind them all is that the face value is obtained by interpolating the values at the center of the cells. The second clarification is about fluxes: during an iterative process for the resolution of the equations, they are calculated using the velocity obtained at the previous step so that the non-linearity is easily resolved.

3.1.3 The diffusion term

The diffusion term is discretized as follows:

$$\int_{\Omega_i} \nu_{eff} \nabla^2 \bar{\boldsymbol{u}} d\Omega_i = (\nu_{eff})_i \int_{S_i} d\boldsymbol{S} \cdot (\nabla \bar{\boldsymbol{u}})\, d\Omega_i \approx \sum_j (\nu_{eff})_{ij} \boldsymbol{S}_{ij} \cdot (\nabla \bar{\boldsymbol{u}})_{ij}, \tag{15}$$

where $(\nu_{eff})_i$ is the effective viscosity of the i-th cell, $(\nu_{eff})_{ij}$ is the effective viscosity evaluated at the center of the face S_{ij}, and $(\nabla \bar{\boldsymbol{u}})_{ij}$ is the gradient of $\bar{\boldsymbol{u}}_{ij}$ evaluated at the center of the face $\boldsymbol{S}_{ij}$. As for the evaluation of the term $\boldsymbol{S}_{ij} \cdot (\nabla \bar{\boldsymbol{u}})_{ij}$ in eq. (15), its value depends on whether the mesh is orthogonal or non-orthogonal. Notice that the gradient of the velocity is not known at the face of the cell. The mesh is orthogonal if the line that connects two cell centers is orthogonal to the face that divides these two cells. For orthogonal meshes, the term $\boldsymbol{S}_{ij} \cdot (\nabla \bar{\boldsymbol{u}})_{ij}$ is evaluated as follows:

$$\boldsymbol{S}_{ij} \cdot (\nabla \bar{\boldsymbol{u}})_{ij} \approx \|\boldsymbol{S}_{ij}\| \frac{\bar{\boldsymbol{u}}_i - \bar{\boldsymbol{u}}_j}{\|\boldsymbol{d}_{ij}\|}, \tag{16}$$

where $\boldsymbol{d}_{ij}$ represents the vector connecting the centers of cells of index i and j. If the mesh is non-orthogonal, then a correction term has to be added to eq. (16). In that case, one has to consider computing a non-orthogonal term to account for the non-orthogonality of the mesh as given by the following relation [24]:

$$\boldsymbol{S}_{ij} \cdot (\nabla \bar{\boldsymbol{u}})_{ij} = \|\boldsymbol{\pi}_{ij}\| \frac{\bar{\boldsymbol{u}}_i - \bar{\boldsymbol{u}}_j}{\|\boldsymbol{d}_{ij}\|} + \boldsymbol{k}_{ij} \cdot (\nabla \bar{\boldsymbol{u}})_{ij}. \tag{17}$$

Herein, $\boldsymbol{S}_{ij} = \boldsymbol{\pi}_{ij} + \boldsymbol{k}_{ij}$ and $\boldsymbol{\pi}_{ij}$ is chose to be parallel to $\boldsymbol{S}_{ij}$ and $\boldsymbol{k}_{ij}$ to be orthogonal to $\boldsymbol{d}_{ij}$. The term $(\nabla \bar{\boldsymbol{u}})_{ij}$ is obtained through interpolation of the values of the gradient at the cell centers $(\nabla \bar{\boldsymbol{u}}_i)$ and $(\nabla \bar{\boldsymbol{u}}_j)$. The discretized forms of eqs. (3) to (5) are written in a compact as follows:

$$\begin{bmatrix} \mathbf{A}_u & \mathbf{B}_p \\ \nabla(\cdot) & \mathbf{0} \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{u}}_h \\ \bar{\boldsymbol{p}}_h \end{bmatrix} = \mathbf{0}. \tag{18}$$

The above system matrix has a saddle point structure which is usually difficult to solve using a coupled approach. For this reason, a segregated approach is used in this work where the momentum equation is solved with a tentative pressure and later corrected by exploiting the divergence-free constraint. Next, the main traits of PIMPLE algorithm is recalled.

3.1.4 The PIMPLE algorithm

The PIMPLE algorithm is a mix of SIMPLE [46], and PISO [23] algorithms. This algorithm is mostly used for unsteady problems requiring a high Courant number or a dynamic mesh such as the one considered in this study. To better understand the procedure of the PIMPLE algorithm, some crucial points about both algorithms are reported in the following, as they will be useful later during the online phase. Note that in this subsection, the quantities $\boldsymbol{u}_h^{n*}$, and $\boldsymbol{p}_h^{n-1}$ are the averaged terms coming from the Reynolds decomposition for velocity and pressure.

Starting with the SIMPLE algorithm the first step is to solve the discretized momentum equation considering the pressure field of the previous iterations. The momentum matrix is divided into diagonal and extra-diagonal parts so that the following holds:

$$\mathbf{A}_u \boldsymbol{u}_h^{n*} = \mathbf{A} \boldsymbol{u}_h^{n*} - \mathbf{H}(\boldsymbol{u}_h^{n*}), \tag{19}$$

with n being an index to identify a generic iteration and $\mathbf{A}_u$ satisfying the following relation:

$$\mathbf{A}_u \boldsymbol{u}_h^{n*} = -\mathbf{B}_p \boldsymbol{p}_h^{n-1}. \tag{20}$$

By using eq. (18), the momentum equation can be reshaped as follows:

$$\mathbf{A}\boldsymbol{u}_h^{n*} = \mathbf{H}(\boldsymbol{u}_h^{n*}) - \mathbf{B}_p \boldsymbol{p}_h^{n*} \Rightarrow \boldsymbol{u}_h^{n*} = \mathbf{A}^{-1}\mathbf{H}(\boldsymbol{u}_h^{n*}) - \mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}_h^{n-1}. \tag{21}$$

In an iterative algorithm, the next step is introducing a small correction to the velocity and pressure field inside the *inner loop*. Then, one can define the following relations:

$$\boldsymbol{u}_h^n = \boldsymbol{u}_h^{n*} + \boldsymbol{u}' \qquad \boldsymbol{p}_h^n = \boldsymbol{p}_h^{n-1} + \boldsymbol{p}'. \tag{22}$$

where $\boldsymbol{u}_h^{n*}$ does not satisfy the continuity equation, and $\boldsymbol{u}_h^n$ does. $\square'$ are the corrections for both terms. By inserting eq. (22) in eq. (21), and rearranging terms give:

$$\boldsymbol{u}_h^n - \boldsymbol{u}' = \mathbf{A}^{-1}[\mathbf{H}(\boldsymbol{u}_h^n) - \mathbf{H}(\boldsymbol{u}') - \mathbf{B}_p \boldsymbol{p}_h^n + \mathbf{B}_p \boldsymbol{p}'] \tag{23}$$

From eq. (23), one deduces a relation between $\boldsymbol{u}'$ and $\boldsymbol{p}'$:

$$\boldsymbol{u}' = \tilde{\boldsymbol{u}}' - \mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}', \tag{24}$$

with

$$\tilde{\boldsymbol{u}}' = \mathbf{A}^{-1}\mathbf{H}(\boldsymbol{u}'). \tag{25}$$

As the following relation holds thanks to eq. (20) :

$$\boldsymbol{u}_h^n = \mathbf{A}^{-1}[\mathbf{H}(\boldsymbol{u}_h^n) - \mathbf{B}_p \boldsymbol{p}_h^n] \tag{26}$$

With the use of eq. (24) and the divergence operator $\nabla(\cdot)$ applied to $\boldsymbol{u}_h^n$ in eq. (22) knowing $\boldsymbol{u}'$ from eq. (24), one obtain an equation that directly relates $\boldsymbol{p}'$ and $\boldsymbol{u}_h^{n*}$:

$$[\nabla(\cdot)]\left(\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}'\right) = [\nabla(\cdot)]\boldsymbol{u}_h^{n*} + [\nabla(\cdot)]\tilde{\boldsymbol{u}}'. \tag{27}$$

Which is basically the discretized Poisson equation for pressure (PPE) expressed in terms of the velocity and pressure corrections. In the SIMPLE algorithm, the velocity corrections $\tilde{\boldsymbol{u}}'$ are unknown as $\mathbf{H}(\boldsymbol{u}')$ are not too and hence neglected, implying the following relation:

$$[\nabla(\cdot)]\left(\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}'\right) = [\nabla(\cdot)]\boldsymbol{u}_h^{n*}. \tag{28}$$

Therefore, $\boldsymbol{p}'$ is expressed as the only function of $\boldsymbol{u}_h^{n*}$ in eq. (28). Then the corrected pressure is entered again in eq. (21) in order to obtain a new velocity field $\boldsymbol{u}_h^{n*}$ and repeat the procedure until the pressure correction falls below a given *tolerance* and the velocity satisfy both the continuity and momentum equation.

As the $\tilde{\boldsymbol{u}}'$ is neglected, the SIMPLE algorithm converges slowly and is used mainly for steady-state simulations. Furthermore, to avoid instabilities, relaxation factor α_p and α_u are introduced in the computation of $\boldsymbol{p}_h^n$ and $\boldsymbol{u}_h^{n*}$ as follows:

$$\boldsymbol{p}_h^n = \boldsymbol{p}_h^{n-1} + \alpha_p \boldsymbol{p}'. \tag{29}$$

$$\boldsymbol{u}_h^{n*} = \mathbf{A}^{-1}\mathbf{H}(\boldsymbol{u}_h^{n*}) - \alpha_u \mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}_h^{n-1}. \tag{30}$$

The PISO algorithm comes to play to speed up the convergence after neglecting $\tilde{\boldsymbol{u}}'$ and *computing the pressure correction* $\boldsymbol{p}'$ using eq. (24). $\boldsymbol{u}'$ is computed as follows:

$$\boldsymbol{u}' = -\mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}'. \tag{31}$$

Allowing the computation of $\tilde{\boldsymbol{u}}'$ using eq. (25). One defines a second velocity correction equation mirroring eq. (24) as follows:

$$\boldsymbol{u}'' = \tilde{\boldsymbol{u}}' - \mathbf{A}^{-1}\mathbf{B}_p \boldsymbol{p}''. \tag{32}$$

As $\boldsymbol{u}''$ in eq. (32) satisfy the continuity equation, one define also a second pressure correction

equation as:

$$[\nabla(\cdot)]\left(\mathbf{A}^{-1}\mathbf{B}_p\boldsymbol{p}''\right) = [\nabla(\cdot)]\tilde{\boldsymbol{u}}'. \tag{33}$$

To sum up, what the PISO algorithm does more than the SIMPLE algorithm is to add a second inner loop to correct pressure and velocity. This speeds up the convergence, allowing this algorithm to be used in a transient simulation. Following the procedure described by eqs. (31) to (33) further corrections steps can be added, increasing both the algorithm's convergence and computational cost. The essential steps of the PIMPLE algorithm involving mesh motion is reported in algorithm 1. In algorithm 1, the iterations within one time-steps are called *outer iterations*, they are performed in an *outer loop* in which the coefficients and the source matrix of the discretized equations are updated. The operations performed on linear systems with fixed coefficients are called instead *inner iterations* and they occur in the so called *inner loop*.

Algorithm 1: PIMPLE algorithm with dynamic mesh.

Input : Initial fields $\boldsymbol{u}_h^{n*}$, $\boldsymbol{p}_h^{n-1}$, ν_t^0, and $\boldsymbol{\delta}^0$ ▷ $\boldsymbol{\delta}^0$ initial displacement;
Output: $\boldsymbol{u}_h^n$, $\boldsymbol{p}_h^n$, ν_t^n, and $\boldsymbol{\delta}^n$;

```
while t ≤ t_end do
    while No. outer corrections ≥ 2 and Tol ≥ maxTol do
        Compute the forces;                                   ▷ Using u_h^{n*}, p_h^{n-1};
        Solve the rigid body problem eqs. (1) and (2)         ▷ To obtain the new COG;
        Solve the mesh motion problem                         ▷ To obtain δ^n see section 2.4;
        A_u u_h^{n*}                                          ▷ Assembling the momentum matrix eq. (19);
        Solve A_u u_h^{n*} = −B_p p_h^{n-1}                   ▷ Momentum predictor eq. (20) to obtain u_h^{n*};
        [∇(·)](A^{-1}B_p p') = [∇(·)]u_h^{n*}                 ▷ Assembling the matrix of PPE eq. (28);
        Solve [∇(·)](A^{-1}B_p p') = [∇(·)]u_h^{n*}           ▷ PPE to obtain p' ;
        u' ← −A^{-1}B_p p'                                    ▷ Momentum corrector eq. (31) ;
        while No. inner corrections do
            [∇(·)](A^{-1}B_p p'') = [∇(·)]ũ'                  ▷ Assembling the matrix for PPE eq. (33);
            Solve [∇(·)](A^{-1}B_p p'') = [∇(·)]ũ'            ▷ Recursively to obtain p'';
            u' ← ũ' − A^{-1}B_p p'':                          ▷ Momentum corrector eq. (32) ;
        Solve turbulence and other transport quantities to obtain ν_t^n;
        Update tolerance;
        u_h^{n*} ← u' ;
        p_h^{n-1} ← p_h^{n-1} + p' ;
```

3.2 The reduced problem

The first subsection recalls the proper orthogonal decomposition for determining the modal basis functions, the next two subsections discuss Galerking projection and the machine learning algorithms to predict the eddy viscosity. The last subsection discusses the radial basis concept for predicting the mesh motion.

3.2.1 The proper orthogonal decomposition

The Proper Orthogonal Decomposition (POD) is used to construct the low-dimensional space. The POD is a compression technique where a set of numerical realizations (in time or parameter space) is reduced into a number of orthogonal basis (spatial modes) that capture the essential information suitably combined from previously acquired system data [1]. In the sequel, the POD is formulated only in time space because in this work the parameter dependency is implicit.

This work applies the POD to a group of realizations called snapshots. It consists of computing a certain number of full-order solutions $\boldsymbol{s}_i = \boldsymbol{s}(t_i)$ where $t_i \in \boldsymbol{T}$ for $i = 1, \cdots, N$. $\boldsymbol{T}$ being the training collection of a certain number N of the time values, to obtain a maximum amount of information from this costly stage to be employed later on for a cheaper resolution of the problem. Those snapshots can be resumed at the end of the resolution all together into a matrix $\boldsymbol{S} \in \mathbb{R}^{N_h \times N}$.

As already mentioned, N_h is the number of control volumes in the discretized domain.

$$\boldsymbol{S} = [\boldsymbol{s}(\boldsymbol{x}, t_1), \ldots, \boldsymbol{s}(\boldsymbol{x}, t_N)]\,. \tag{34}$$

The idea is to compute the ROM solution that can minimize the error denoted here by E see eq. (37) between the obtained realization of the problem and its high-fidelity counterpart. In the POD-Galerkin scheme, the reduced order solution is represented as follows:

$$\boldsymbol{s}(\boldsymbol{x}, t) \approx \boldsymbol{s}^{ROM}(\boldsymbol{x}, t) = \sum_{i=1}^{N_r} a_i(t)\boldsymbol{\phi}_i(\boldsymbol{x}). \tag{35}$$

Where $N_r \ll N_h$ (N_h is the number of cells in the computational domain) is a predefined number, namely the dimension of the reduced solution manifold, $\boldsymbol{\phi}_i$ is a generic pre-calculated ortho-normal function depending only on the space while $a_i(t)$ is the temporal modal coefficients satisfying the following conditions:

$$a_j(t) = (\boldsymbol{\phi}_j, \boldsymbol{s}(\boldsymbol{x}, t))_{L^2(\Omega)}\,, \qquad \boldsymbol{\phi}_j^T \boldsymbol{M} \boldsymbol{\phi}_i = \delta_{ij}. \tag{36}$$

$\boldsymbol{M}$ being the mass matrix defined by the chosen inner product. In the case of L_2-norm and FVM $\boldsymbol{M}$ is a diagonal matrix containing the cell volumes. The best performing functions $\boldsymbol{\phi}_i$ in this case, are the ones minimizing the L^2-norm error E between all the reduced-order solutions $\boldsymbol{s}_i^{ROM}$, $i = 1, \cdots, N$ and their high fidelity counterparts:

$$E = \sum_{i=1}^{N} \|\boldsymbol{s}_i^{ROM} - \boldsymbol{s}_i\|_{L^2(\Omega)} = \sum_{i=1}^{N} \|\boldsymbol{s}_i - \sum_{i=1}^{N_r} (\boldsymbol{s}_i, \boldsymbol{\phi}_i)_{L^2(\Omega)}\, \boldsymbol{\phi}_i\|_{L^2(\Omega(t_0))}. \tag{37}$$

It can be shown that solving a minimization problem based on eq. (37) is equivalent to solving the following eigenvalue problem [32]:

$$\boldsymbol{C}\mathbf{V} = \mathbf{V}\boldsymbol{\lambda}. \tag{38}$$

$\boldsymbol{C} \in \mathbb{R}^{N \times N}$ being the correlation matrix between all the different training solutions of the snapshot matrix $\boldsymbol{S}$, $\mathbf{V} \in \mathbb{R}^{N \times N}$ is the matrix whose columns are the eigenvectors, and $\boldsymbol{\lambda} \in \mathbb{R}^{N \times N}$ is a diagonal matrix whose diagonal entries are the eigenvalues. The entries of the correlation matrix are defined as follows:

$$\boldsymbol{C}_{ij} = (\boldsymbol{s}_i, \boldsymbol{s}_j)_{L^2\Omega(t_0))}\,. \tag{39}$$

$\Omega(t_0)$ being the reference configuration of the computational domain in the case of grid motion. Note that the projection is performed with respect to $L_2(\Omega(t))$ while POD is computed with respect to $L_2(\Omega(t_0))$. Using a POD strategy, the required basis functions are obtained through the resolution of the eigenproblem mentioned in eq. (38), obtained with the method of snapshots by solving eq. (37). One can compute the required basis functions as follows:

$$\boldsymbol{\phi}_i = \frac{1}{N\sqrt{\lambda_i}} \sum_{j=1}^{N} \boldsymbol{s}_j V_{ji} \qquad \forall i = 1, \cdots, N. \tag{40}$$

All the basis functions are collected into a single matrix:

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \cdots, \boldsymbol{\phi}_{N_r}] \in \mathbb{R}^{N_h \times N_r}. \tag{41}$$

Which is used to project the high fidelity problem onto the reduced subspace so that the final system dimension is N_r. This procedure leads to a problem requiring a computational cost that is much lower than the original problem. Next, the Galerkin projection, adapted in this study is presented.

3.2.2 POD-Galerkin projection for velocity and pressure equations

All the high-fidelity solutions are obtained by employing a segregated algorithm iterating the momentum and pressure equations until convergence is reached. The full-order model for both the velocity and pressure in the discretized form is given as follows:

$$\mathbf{A}_u \boldsymbol{u}_h = \boldsymbol{b}_u, \tag{42}$$

$$\mathbf{B}_p \boldsymbol{p}_h = \boldsymbol{b}_p. \tag{43}$$

With $\mathbf{A}_u \in \mathbb{R}^{dN_h \times dN_h}$, $\boldsymbol{u}_h \in \mathbb{R}^{dN_h}$, $\mathbf{B}_p \in \mathbb{R}^{N_h \times N_h}$ is the matrix operator for Poisson equation pressure (PPE) discussed in subsection 3.1, $\boldsymbol{p}_h \in \mathbb{R}^{N_h}$, and $d = 2$ is the dimension of the computational domain. N_h being the number of control volumes (cells) in the mesh, $\boldsymbol{b}_u$ and $\boldsymbol{b}_p$ are the respective source terms in the discretized form of the momentum equation and PPE. In this section, Galerkin projection (on the fully discrete equations) is used for the construction of the reduced-order method. Next, the reduced expansions of the velocity and pressure fields is introduced: $\boldsymbol{u}_h(\boldsymbol{x}, t) \approx \boldsymbol{u}_r(\boldsymbol{x}, t)$ and $\boldsymbol{p}_h(\boldsymbol{x}, t) \approx p_r(\boldsymbol{x}, t)$, with

$$\boldsymbol{u}_r(\boldsymbol{x}, t) = \sum_{i=1}^{N_u} a_i(t)\boldsymbol{\phi}_i(\boldsymbol{x}) = \boldsymbol{\Phi}\boldsymbol{a}^T, \tag{44}$$

$$p_r(\boldsymbol{x}, t) = \sum_{i=1}^{N_p} b_i(t)\boldsymbol{\xi}_i(\boldsymbol{x}) = \boldsymbol{\Xi}\boldsymbol{b}^T. \tag{45}$$

Herein, $a_i(t)$ and $b_i(t)$ are modal coefficients; $\boldsymbol{\phi}_i$ and $\boldsymbol{\xi}_i$ are the basis functions corresponding to the POD modes of the velocity and pressure fields stored respectively in $\boldsymbol{\Phi} \in \mathbb{R}^{dN_h \times N_u}$ and $\boldsymbol{\Xi} \in \mathbb{R}^{N_h \times N_p}$ with N_u and N_p being the numbers of basis functions selected for the predicted velocity and pressure solutions respectively, $\boldsymbol{a} \in \mathbb{R}^{N_u}$ is the vector containing the coefficients for the velocity expansion while the same reads for pressure with respect to $\boldsymbol{b} \in \mathbb{R}^{N_p}$. For the construction of the reduced basis spaces, the POD strategy mentioned in subsection 3.2 is used on the snapshot matrices of the velocity and pressure fields to obtain two separate families of reduced basis functions.

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_{N_u}] \in \mathbb{R}^{dN_h \times N_u}, \tag{46}$$

$$\boldsymbol{\Xi} = [\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{N_p}] \in \mathbb{R}^{N_h \times N_p}. \tag{47}$$

The eqs. (42) and (43) are projected using in eqs. (46) and (47) respectively leading to:

$$\boldsymbol{A}_u^r \boldsymbol{a} = \boldsymbol{b}_u^r, \tag{48}$$

$$\boldsymbol{A}_p^r \boldsymbol{b} = \boldsymbol{b}_p^r. \tag{49}$$

Where $\boldsymbol{A}_u^r = \boldsymbol{\Phi}^T \mathbf{A}_u \boldsymbol{\Phi} \in \mathbb{R}^{N_r^u \times N_r^u}$, $\boldsymbol{A}_p^r = \boldsymbol{\Xi}^T \mathbf{A}_p \boldsymbol{\Xi} \in \mathbb{R}^{N_r^p \times N_r^p}$, $\boldsymbol{b}_u^r = \boldsymbol{\Phi}^T \boldsymbol{b}_u \in \mathbb{R}^{N_r^u}$, and $\boldsymbol{b}_p^r = \boldsymbol{\Xi}^T \boldsymbol{b}_p \in \mathbb{R}^{N_r^p}$. The resulting eqs. (48) and (49) can be solved using any method for dense matrices. In this work, the Householder rank-revealing QR decomposition of a matrix with full pivoting is used and it is available in the Eigen library [16]. As the main idea here is to rely on a method capable of being as coherent as possible concerning the high-fidelity problem (algorithm 1), in the following the main steps for the reduced algorithm related to incompressible turbulent flows with mesh motion are reported in algorithm 2.

3.2.3 Machine learning for eddy viscosity prediction

In this work, a deep neural network is used for the prediction of the eddy viscosity mimicking the works done in [9, 20, 67]. In [20] radial basis functions were used to predict the temporal coefficients of the eddy viscosity based on the temporal coefficients of the velocity. The study carried out in [67] used a fully connected neural network to predict the parameterized coefficients of the eddy viscosity in a steady simulation. Scholars in [9] studied the effects of the effective viscosity in projection-based ROM simulations and employed a simple spline interpolation for the prediction of the first dominant mode of the temporal coefficient of the eddy viscosity from known values.

Algorithm 2: Reduced-PIMPLE algorithm with dynamic mesh.

Input : Initial fields $\boldsymbol{u}_h^{n*}$, $\boldsymbol{p}_h^{n-1}$, ν_t^0, and $\boldsymbol{\delta}^0$ ▷ $\boldsymbol{\delta}^0$ is the initial node displacement;
Output: $\boldsymbol{u}_h^n$, $\boldsymbol{p}_h^n$, ν_t^n, and $\boldsymbol{\delta}^n$;

while $t \leq t_{end}$ **do**
  while *No. outer corrections* ≥ 2 *and* $Tol \geq maxTol$ **do**
    Compute the forces; ▷ Using $\boldsymbol{u}_h^{n*}$, $\boldsymbol{p}_h^{n-1}$;
    Solve the rigid body problem eqs. (1) and (2) ▷ To obtain the new COG $\boldsymbol{\vartheta}_{new}$;
    Compute $\boldsymbol{c} = RBF(\boldsymbol{\vartheta}_{new})$ eq. (59);
    Reconstruct $\boldsymbol{\delta}^n = \Psi \boldsymbol{c}^T$ eq. (60);
    $\mathbf{A}_u \boldsymbol{u}_h^{n*}$ ▷ Assembling the momentum matrix eq. (19);
    Solve $\boldsymbol{\Phi}^T \mathbf{A}_u \boldsymbol{\Phi} \boldsymbol{a}^* = \boldsymbol{\Phi}^T \mathbf{b}_u$ ▷ To obtain $\boldsymbol{a}^*$ with $\mathbf{b}_u = -\mathbf{B}_p \boldsymbol{p}_h^{n-1}$;
    Reconstruct $\boldsymbol{u}_h^{n*}$ ▷ Using $\boldsymbol{a}^*$;
    $[\nabla(\cdot)] \left(\mathbf{A}^{-1} \mathbf{B}_p \boldsymbol{p}'\right) = [\nabla(\cdot)] \boldsymbol{u}_h^{n*}$ ▷ Assembling the matrix of PPE eq. (28);
    Solve $\boldsymbol{\Xi}^T \mathbf{A}_p \boldsymbol{\Xi} \boldsymbol{b}' = \boldsymbol{\Xi}^T \boldsymbol{b}_p$ ▷ To obtain $\boldsymbol{b}'$;
    Reconstruct $\boldsymbol{p}'$ ▷ Using $\boldsymbol{b}'$;
    $\boldsymbol{u}' \leftarrow -\mathbf{A}^{-1} \mathbf{B}_p \boldsymbol{p}'$ ▷ Momentum corrector eq. (31) ;
    while *No. inner corrections* **do**
      $[\nabla(\cdot)] \left(\mathbf{A}^{-1} \mathbf{B}_p \boldsymbol{p}''\right) = [\nabla(\cdot)] \tilde{\boldsymbol{u}}'$ ▷ Assembling the matrix for PPE eq. (33);
      Solve $\boldsymbol{\Xi}^T \mathbf{A}_p \boldsymbol{\Xi} \boldsymbol{b}'' = \boldsymbol{\Xi}^T \boldsymbol{b}_p$ ▷ Recursively to obtain $\boldsymbol{b}''$ where $\boldsymbol{b}_p = [\nabla(\cdot)] \tilde{\boldsymbol{u}}'$;
      Reconstruct $\boldsymbol{p}''$ ▷ Using $\boldsymbol{b}''$;
      $\boldsymbol{u}' \leftarrow \tilde{\boldsymbol{u}}' - \mathbf{A}^{-1} \mathbf{B}_p \boldsymbol{p}''$: ▷ Momentum corrector eq. (32) ;
    Evaluate the networks using NNs or LSTM ;
    Reconstruct the new turbulent viscosity ν_t^n;
    $\boldsymbol{u}_h^{n*} \leftarrow \boldsymbol{u}'$;
    $\boldsymbol{p}_h^{n-1} \leftarrow \boldsymbol{p}_h^{n-1} + \boldsymbol{p}'$;

A fully connected neural network and a recurrent neural network based on the LSTM are presented here to predict temporal coefficients of eddy viscosity. The choice of the aforementioned methodologies relies on the idea of building a reduced problem independent of the turbulent technique ($k-\epsilon$, $k-\omega$, etc.) that might be used in the original problem to calculate the eddy viscosity. The first method uses the physical relation between the velocity field and eddy viscosity thanks to the Boussinesq hypothesis. The second method assume a one-to-one dynamical mapping between the low-dimensional states $\boldsymbol{n}^i$ and $\boldsymbol{n}^{i-1}$ defined in eq. (50). The low-dimensional eddy viscosity is approximated first using the POD decomposition on the snapshot matrix $\boldsymbol{S}_{\nu_t} \in \mathbb{R}^{N_h \times N_s}$ as :

$$\nu_t(\boldsymbol{x}, t) \approx \sum_{i=0}^{N_{\nu_t}} n_i(t) \boldsymbol{\psi}_i(\boldsymbol{x}) = \boldsymbol{\Psi} \boldsymbol{n}^T. \tag{50}$$

$\psi_i(\boldsymbol{x})$ and $n_i(t)$ are the POD spatial and temporal coefficients modes for eddy viscosity respectively. $N_{\nu_t} \ll N_s$ denotes the selected number of modes to predict the eddy viscosity. In contrast to the temporal coefficients modes of the velocity and pressure obtained by projecting the FOM onto the respective POD spatial modes and subsequently solving the reduced problem, the predicted temporal coefficients modes for the eddy viscosity are modeled via a multi-layer feed-forward neural network or a recurrent network.

- **Neural networks:** Within this method, the neural networks are fed with temporal coefficients of the velocity field $\boldsymbol{a}$ and mapped to the temporal coefficient of the turbulent viscosity $\tilde{\boldsymbol{n}}$. $\tilde{\boldsymbol{n}}$ being the prediction from the neural network. For a comprehensive description, the reader could refer to Goodfellow et al. [19]. In a feed-forward neural network, the output of a single layer of a given input $\mathbf{A} \in \mathbb{R}^N$ given by $\mathbf{Y} = f(\mathbf{W}\mathbf{A} + \mathbf{b})$. $\mathbf{W} \in \mathbb{R}^{M \times N}$ being the weight matrix, $\mathbf{b} \in \mathbb{R}^M$ is a bias term and f (.) is a *nonlinear function* that acts element-wise on its inputs. The Multi-layer neural networks are generated by feeding the output $\mathbf{h}_l = f_{l+1}(\mathbf{W}_l \mathbf{A}_l + \mathbf{b}_l)$ of a layer l as the input of the next layer. The vector $\mathbf{h}_l$ is often referred to as the hidden state or feature vector at the l-th layer. Generally, training a network involves finding the parameters $\boldsymbol{\theta} = \{\mathbf{W}_l, \mathbf{b}_l\}_{l=0}^{l=L-1}$ such that the expected loss

$\mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y})$ between the output $\mathbf{Y}$ and the target value $\hat{\mathbf{Y}}$ is minimized i.e.

$$\boldsymbol{\theta}_* = \arg\min_{\boldsymbol{\theta}}[\mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y})], \tag{51}$$

where $\mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y})$ is some measure of discrepancy between the predicted and target outputs given in some cases by:

$$\mathcal{L}(\hat{\mathbf{Y}}, \mathbf{Y}) = \left[\frac{1}{n-1}\sum_{i=1}^{n}\frac{||\boldsymbol{Y}_i - \boldsymbol{g}(\boldsymbol{Y}_{i-1};\boldsymbol{\theta})||_2^2}{||\boldsymbol{Y}_i||_2^2}\right]. \tag{52}$$

For example in the case of recurrent neural networks with $\boldsymbol{g}$ a one-to-one mapping giving by: $\hat{\boldsymbol{Y}}_i = \boldsymbol{g}(\boldsymbol{Y}_{i-1};\boldsymbol{\theta})$.

- **Recurrent neural networks:** This method assumes the following relationships:

$$\nu_t^{i+1} = \boldsymbol{g}(\nu_t^i) \Rightarrow \tilde{\boldsymbol{n}}^{i+1} = \boldsymbol{g}(\tilde{\boldsymbol{n}}^i). \tag{53}$$

$\boldsymbol{g}$ being an unknown mapping between the flow fields of adjacent time steps. The goal consists of learning a dynamical operator $\boldsymbol{g}$ by finding the parameters $\boldsymbol{\theta}$ (weights and biases) that allow to recurrently predict finite time series of the low-dimensional states. Recurrent neural networks (RNNs) are designed specifically to deal with sequential data. Data coming from transient CFD simulations or any nonlinear dynamical system is a set of time series that is sequential. This makes the application of RNNs for non-linear dynamical systems quite relevant [6]. RNNs differ from the traditional feed-forward neural network with the presence of a recurrent connection. This particular recurrent connection is responsible for storing the history of previous inputs ($\boldsymbol{X}_i$) via hidden states. The basic mathematical structure of simple RNNs cell for an input $\mathbf{A}_i \in \mathbb{R}^N$ and output $\mathbf{Y}_i \in \mathbb{R}^M$ given as follows:

$$\mathbf{X}_i = f(\mathbf{W}_x\mathbf{X}_{i-1} + \mathbf{W}_a\mathbf{A}_i + \mathbf{b}) \quad \text{and} \quad \mathbf{Y}_i = g(\mathbf{W}_y\mathbf{X}_i). \tag{54}$$

$\mathbf{W}_x \in \mathbb{R}^{N_x \times N_x}$, $\mathbf{W}_a \in N_x \times N$, and $\mathbf{W}_y \in \mathbb{R}^{N_x \times M}$ being the hidden, input and output weight matrices respectively. $\mathbf{b} \in \mathbb{R}^M$ represents the bias term, and $\boldsymbol{X}_{i-1}$ the cell state at time $i-1$. RNNs are typically trained using stochastic gradient descent (SGD), or some variant, but the gradients are calculated using the backpropagation through time (BPTT) algorithm [65]. For a detailed discussion on BPTT the reader might refer to [4]. This work considers RNNs equipped with LSTM (long short-term memory) units [51]. The straightforward prediction of $\tilde{\nu}_t$ at the online level is given using eq. (50) i.e.

$$\tilde{\nu}_t = \Psi\tilde{\boldsymbol{n}}^T. \tag{55}$$

3.2.4 POD with interpolation for mesh motion prediction

This section presents a method to reduce the computational cost associated with the mesh motion part in the system. The advantage of this methodology is to make the online part independent of the mesh motion technique used at the offline stage. The methodology combines proper orthogonal decomposition with radial basis functions (RBF) networks on the point displacement field. The interpolation using RBF is given by the following formula:

$$f(\boldsymbol{x}_j) = \sum_{j=1}^{N}\boldsymbol{w}_j\rho(||\boldsymbol{x}_j - \boldsymbol{x}_k||), \tag{56}$$

where $f : \mathbb{R} \mapsto \mathbb{R}$ a known map at some finite number of points $f(\boldsymbol{x}_k) = y_k$, $k = 1, \cdots, N$, ρ is a radial basis function, N is the number of neurons in the hidden layer, and $\boldsymbol{w}_j$ being the weight of neuron j in the linear output neuron. eq. (56) can be rewritten as a linear system, namely:

$$\boldsymbol{y} = \mathbf{G}\boldsymbol{w}^T, \tag{57}$$

where $\mathbf{G} = (g_{kj}) = \rho(||\boldsymbol{x}_k - \boldsymbol{x}_j||)$ being the Gram matrix. The weights are given by: $\boldsymbol{w} = \mathbf{G}^{-1}\boldsymbol{y}$. The rationale behind POD-RBF regards the evaluation of the new temporal or parameterized

coefficients computed at the parameter points $\boldsymbol{\vartheta}_k$; in this work, it will be a two-dimensional vector of the pitch and plunge at a given time t. Next, the separability's assumption on the point displacement field is given as follows:

$$\boldsymbol{\delta}(\boldsymbol{x}, t_k) \approx \sum_{i=1}^{N_\delta} c_i(\boldsymbol{\vartheta}(t_k))\boldsymbol{\psi}_i(\boldsymbol{x}), \quad \forall \boldsymbol{\vartheta}(t_k) = \vartheta_k \in \boldsymbol{\Theta}. \tag{58}$$

$\boldsymbol{\Theta}$ being the training set of the airfoil motion at the offline stage, and N_δ the require number of basis to approximate the $\boldsymbol{\delta}$. In the online stage, as input, a new time-parameter value $\boldsymbol{\vartheta}_{new}$ is given, and using the trained weights obtained from eq. (57), the coefficient $c_i = c_i(\vartheta_{new})$ is obtained by interpolation with RBF and $\boldsymbol{\vartheta}_{new}$ is obtained by solving eqs. (1) and (2) at the online stage. Then, the new coefficient is computed by:

$$c(\boldsymbol{\vartheta}_{new}) = \sum_{j=1}^{N_\delta} \boldsymbol{w}_j \rho(||\boldsymbol{\vartheta}_{new} - \boldsymbol{\vartheta}_j||). \tag{59}$$

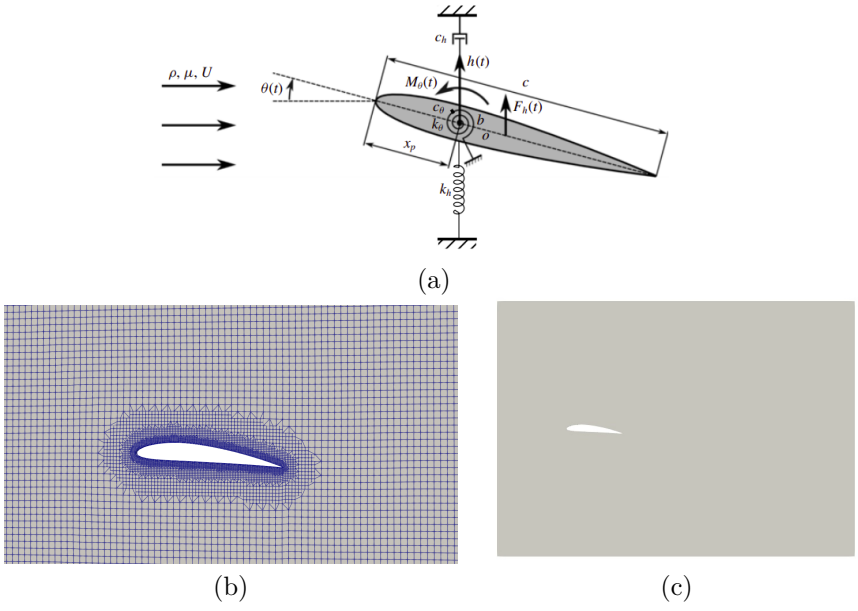
The new point displacement is predicted as follows:

$$\boldsymbol{\delta}(\boldsymbol{x}, t_{new}) = \sum_{i=1}^{N_\delta} c_i(\boldsymbol{\vartheta})\boldsymbol{\psi}_i(\boldsymbol{x}) = \Psi \boldsymbol{c}^T. \tag{60}$$

4 Definition of test case, simulation results, and discussion

This section shows the results obtained for our reference test case which represents two-dimensional turbulent flow past a plunging and pitching airfoil. The simulation is carried out for a total time of 500 flow through times (FTTs). In the present case, such time unit is defined as the time $\text{FTT} = \dfrac{L}{||\boldsymbol{U}_\infty||} = 0.01$ required by a fluid particle to travel a distance equivalent to the airfoil chord L at the speed of the undisturbed stream $||\boldsymbol{U}_\infty||$ [30].

The 500 FTTs duration choice allows for the flow to fully develop also in the wake region. The test case of the study is the analysis of a two-degree freedom flutter as shown in fig. 1.



(a)

(b) (c)

Figure 1: (a) Schematic of the fluid-structure system considered: a foil allowed to undergo 2 degrees of freedom fully passive plunging and pitching motion with spring constraints [64], (b) a picture of the zoomed mesh with 12 556 cells (control volumes) and 26 316 node points near an airfoil of chord length 1.0 m, (c) picture showing the position of the foil in the computational domain

4.1 Definition of the test case

Fig. 1 (c) shows the 2D computational domain used in this work. The grid features 12 556 cells (control volumes) and 26 316 node points. It also features an extrusion layer around the airfoil to better capture the physical boundary layer. The molecular viscosity $\nu = 10^{-5}\,\mathrm{m}^2/\mathrm{s} \approx \nu_{air}$.

The boundary conditions prescribed for the velocity field at the inflow boundary are non-homogeneous Dirichlet. The velocity value imposed is that of a uniform and constant horizontal velocity $\boldsymbol{U}_\infty = (U_{in}, 0)$ with $U_{in} = 10^2\,\mathrm{m\,s}^{-1}$. Given the airfoil's chord length $L = 1.0\,\mathrm{m}$, the resulting Reynolds number is 10^7. On the — moving — airfoil boundary a Dirichlet boundary condition is applied, imposing that the fluid's velocity is equal to the airfoil surface one. On the top and bottom boundaries, we made use of symmetry boundary conditions, while, zero pressure value and zero normal velocity gradient are prescribed at the outflow boundary. The full-order simulations are carried out using the PIMPLE algorithm, as described in section 3.1. The PIMPLE algorithm can adapt the time step in a way that assures the maximum Courant number does not exceed a prescribed value which in this case, has been set to 0.5. As Reynolds average Navier-Stokes (RANS) is used in this work, the time step Δt is chosen based on the following *rule of thumb* [30]: $\Delta t \approx 100\Delta t_{DNS}$.

$$\text{with} \quad \Delta t_{DNS} \approx \frac{Co \times \eta}{U_{in}} \quad \text{and} \quad \eta \equiv L \times Re^{-\frac{3}{4}}, \tag{61}$$

where L is the size of the largest eddies (in this work, $L = 1m$), η is the Kolmogorov length scale and it is the smallest hydrodynamic scale in turbulent flows.

The turbulence model used in this test is the $k-\omega$ SST model, which in several works (see for instance [47]) proved capable of simulating turbulent flows associated with vortex induced vibrations. The Implicit Euler scheme is used for time discretization. As for the spatial gradients, a Gauss linear scheme is employed. The convective and diffusive terms have been approximated with the first-order Upwind scheme [54] for more stability. The reason is that, in the transport-dominated turbulent regime here under study, the local Peclet number can reach peak values greater than 2. The values of the relaxation factors α_u, and α_p are fixed at 0.7 and 0.3 respectively. One non-orthogonal correction at each PIMPLE iteration is used to deal with the mesh's non-orthogonality. In addition, one pressure correction (inner correctors) and two momentum corrections (outer correctors) are used in the simulations. The linear solver selected combines a smoother Gauss-Seidel solver used for the pressure equation, and a symmetric Gauss-Seidel solver for the momentum equation. The structural motion is computed by means of the `sixDoFRigidBodyMotionSolver` OpenFOAM solver. Given the external fluid dynamic forces acting on a rigid body, such a solver is able to compute the linear and angular displacements in three dimensions. However, the airfoil here considered is only free to translate along the vertical direction (plunge displacement) and rotate along the axis perpendicular to the planar domain Ω (pitch displacement). The resulting system of two second-order differential eqs. (1) and (2) is solved using the Symplectic second-order explicit time-integrator for solid-body motion [10]. The Arbitrary Lagrangian-Eulerian (ALE) method deals with the motion of the grid nodes resulting from the fluid-structure coupling. In particular, the plunging displacement h, and the pitching displacement θ of the airfoil are used to deform the mesh (in the transverse and rotational directions), as described in Subsection 3.2.4. Table 1 reports a comprehensive summary of all the modeling and numerical parameter values used for the simulation setup.

4.2 Simulation results

This section presents the results obtained with the reduced model developed on the airfoil test case described earlier. As mentioned, the ROM is based on the POD-Galerkin approach for the momentum and continuity equations (velocity and pressure fields), on POD-LSTM or POD-NNs for the online eddy viscosity computation, and POD-RBF for the mesh displacement update.

Table 1: Summary of simulation settings of the flow passing pitch-plunge airfoil

Flow settings		**Structure settings**	
Re	10^7	$f_{sh} = f_h = f_\theta$	20 Hz
Time scheme	Implicit Euler	Time scheme	Sympletic
Gradient scheme	cellLimited Gauss linear 1	m	22.9 g
Convective scheme	Gauss upwind	g_z	$-9.81\,\mathrm{m/s^2}$
Laplacian scheme	Gauss linear limited 0.5	L_z	-2
$St = \frac{f_{sh} c \sin\alpha}{U_\infty}$	0.2	k_h	$3.6262 \times 10^5\,\mathrm{N\,m^{-1}}$
U_{in}	$10^2\,\mathrm{m\,s^{-1}}$	k_θ	$3.25 \times 10^4\,\mathrm{N\,m^{-1}}$
Co	0.5	c_h	$2\,\mathrm{N\,m^{-1}}$
Δt_{DNS}	$\frac{C_0 \times Re^{-\frac{3}{4}}}{U_{in}}$	c_θ	$5 \times 10^{-1}\,\mathrm{N\,m^{-1}}$
Turbulence model	$k-\omega$ SST	I_{zz}	2.057121362

In Table 1, St is the Strouhal number, f_{sh} the frequency of the vortex shedding. L_z is the angular momentum in Z-direction, and g_z the gravity in the Z-direction. The following quantities: m, f_h, f_θ, k_h, c_θ, k_θ, c_h, and I_{zz} are defined in section 2.1.

Note that the FVM C++ library *OpenFOAM* version 2106 [26] has been used for data collection at the full-order model level. Such a numerical solver, widely used in industrial applications [25, 52] exploits the fact that FVM locally respects the balance of momentum and mass. At the reduced level, the reduction and resolution of the reduced system are carried out using the C++-based library ITHACA-FV (In real Time Highly Advanced Computational Applications for Finite Volumes) [56, 57]. ITHACA-FV is designed to carry out Galerkin projection of PDE problems that, at the full-order level, are solved making use of FV discretization based on OpenFOAM. The interpolation using RBF in this work has been carried out using the C++ library SPLINTER [15]. The radial basis used for interpolation is the *thin plate spline* with the radial basis's radius set to 1. Finally, the RNNs/ NNs are built using the PyTorch library [44]. The next Subsection assesses the qualitative prediction of the reduced model.

4.2.1 Prediction quality

The point of this Subsection is that of establishing the accuracy of the modal decomposition upon which the reduced model is based. Table 2 presents the eigenvalues associated with the first five dominant modes of all the fields of interest. The values reported also suggest that for the grid nodes motion (pointDisplacement field), one single mode could be enough to predict the mesh motion with acceptable accuracy. Hence, in this study, 1 mode will be used to predict the airfoil motion. A more comprehensive view of the modes eigenvalues magnitude is presented in fig. 2.

# Modes	Eigenvalues U	Eigenvalues p	Eigenvalues ν_t	Eigenvalues pointDisplacement
1	1	1	1	1
2	0.001480623	0.07835354	0.007290981	0.027573111
3	0.001133095	0.008968419	0.001695786	0.00201184
4	0.000992664	0.001758634	0.000879206	0.000000351
5	0.000356768	0.000941899	0.000580963	0.00000000018

Table 2: Normalized eigenvalues of the POD modes of the fields of interest

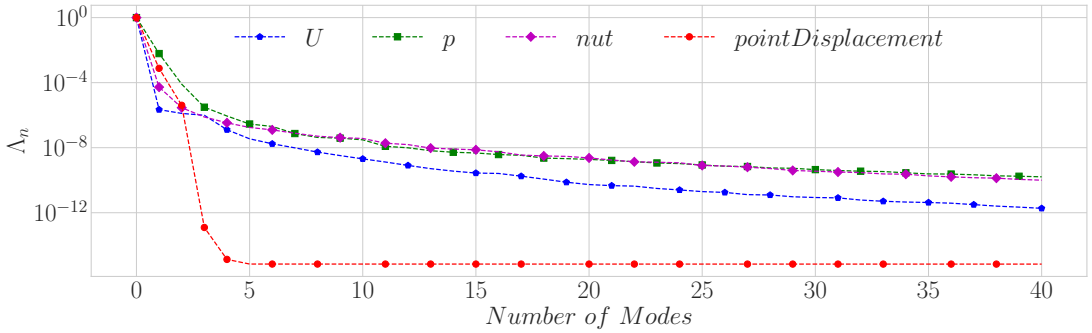


Figure 2: The decay of the POD modes eigenvalues for velocity, pressure, pointDisplacement, and Eddy viscosity fields. Color code: blue – velocity, green –pressure, red – pointDisplacement, magenta – eddy viscosity

4.2.2 Machine learning of the temporal eddy viscosity coefficients

For all the training runs in this work, a variant of the stochastic gradient descent algorithm called adaptive moment estimation ADAM [29] is used as an optimizer. ADAM has an adaptive learning rate method which is commonly used to train deep networks. The optimization is based on a scaled version of the modal coefficients, given by

$$\hat{\boldsymbol{a}}_j(t) = \frac{\boldsymbol{a}_j(t) - \langle \boldsymbol{a}_j(t) \rangle}{\sigma[\boldsymbol{a}_j(t)]}, \tag{62}$$

where $\langle \boldsymbol{a}_j(t) \rangle$ is the mean value of the modal coefficient time series considered and $\sigma[\boldsymbol{a}_j(t)]$ is the corresponding variance. The dataset scaling is necessary to avoid that the gradients that enter the computations of the cost function are too small. In such a case, it would be impossible to generate significant updates of the parameters of the network [5]. The model parameters (weights and bias) are trained with the PyTorch library [45] and later imported in the C++ solver to generate the transient predicted solution for the eddy viscosity during the online computations. The full details on training, validation, and testing are reported in A.1. The accuracy of the resulting feed-forward NN and LSTM-RNNs model results are illustrated in Fig. 16. The four diagrams represent the time series of the modal coefficients corresponding to the four energetically dominant modes of the full-order model eddy viscosity, as well as their data driven approximations. In the diagrams, the two dashed vertical lines divide the time axis into the training window, on the left, the validation window, located between the red and blue dashed lines, and the testing window, on the right. The picture confirms that both models are able to capture the overall trend of the reference FOM solution, not only in the training time window, but also in the and validation and testing ones. By a quantitative perspective, both data driven models appear accurate in the reproduction of frequency, amplitude and phase of the first three modal coefficients of the eddy viscosity field. The plot corresponding to the fourth modal coefficient shows instead a drop of the accuracy of the feed-forward NN prediction, while, on the other hand, the LSTM-RNN prediction remains as accurate as for the previous modes. Thus, this preliminary analysis suggests that both feed-forward NN and LSTM-RNN data driven algorithms used are in principle capable of approximating with good accuracy the eddy viscosity modal coefficients. This is of course crucial for a correct closure of the turbulent problem at the reduced level. The next sections will then assess if the quality of the eddy viscosity approximation — which appears high except for higher order modal coefficients obtained with feed-forward NN — will translate into accurate ROM results.

4.2.3 ROM online resolution time

The data were generated by transient simulations running for one second and saving snapshots of the flow field every 0.0005 seconds for a total of 2001 snapshots. All simulations were run on an HP Pavilion laptop with AMD Ryzen 7 5700u with Radeon graphics ×16, 16GB RAM, AMD Renoir graphics card, and Ubuntu 20.04 operating system. Table 3 reports a comparison analysis of the the full-order and reduced-order models execution times as the number of modes for the prediction

Stages	*# of modes*	Time [s]
Offline	-	3.441516e+4
POD-NNs	$N_u = N_p = 5$, $N_{nut} = 3$, $N_{pD} = 1$	1.770725259e+4
	$N_u = N_p = 10$,$N_{nut} = 5$,$N_{pD} = 1$	1.957406359e+4
	$N_u = 15, N_p = 5$, $N_{nut} = 2$, $N_{pD} = 3$	2.159144848e+4
	$N_u = 10, N_p = 5, N_{nut} = 3$, *and* $N_{pD} = 1$	1.942751287e+4
POD-LSTM	$N_u = N_p = N_{nut} = 5$ *and* $N_{pD} = 3$	1.766056714e+4
	$N_u = 15, N_p = 5$, $N_{nut} = 3$ *and* $N_{pD} = 3$	2.136608204e+4

Table 3: Offline and Online times comparison varying the number of modes

of velocity, pressure, pointDisplacement, and eddy viscosity is varied. This allows for evaluating the effect of the number of modes variation on the computational cost of the online phase.

The offline stage comprises four steps: the computation of the snapshot (computed by a numerical approximation of the original high-dimensional system), computation of the POD basis, projection of the dynamics on the low-rank subspace, and the Machine Learning training of the neural networks (including for the radial basis networks). But only the computational coast of the first step is reported in Table 3 as it is the most expensive one. The online coast is the computational time needed to compute the solutions of the surrogate model. In Table 3, one can observe a small speed-up as this work does not employ hyper-reduction technique.

However, the results in Table 3 suggest that the speed-up obtained by the online solution of the reduced system is not proportional to the reduction of the unknowns obtained at the reduced-order level. This is due to the fact that in the presence of a deforming domain such as the one characterizing our FSI simulations, the entries of the matrices of the ROM system must be computed at each time step through integrals on the updated full-order grid. Clearly, this is at the moment representing a major bottleneck towards a ROM which grants significant computational cost reduction with respect to its FOM counterpart, and work is being carried out towards lowering the computational cost associated with the reduced model assembling. Nonetheless, the main goal of the present work is that of assessing the accuracy of the ROM approach taken. In particular, it is important establishing whether the interaction between the physics based reduction of the fluid dynamic balance equations, and the data driven reduction of the turbulence and grid displacement equations, results in an accurate solver.

4.2.4 ROM online resolution quality

The present Subsection aims at analyzing how close the predicted ROM solutions are with respect to the FOM ones. To this end, figs. 3 and 4 shows a qualitative comparison between the solution fields contour plots corresponding to time $t = 0.1$ s obtained with both the FOM and ROM solvers. The plots in the figure confirm that, to the eyeball test, the ROM solutions obtained using both POD-NN and POD-LSTM appear by all means similar to the high-fidelity ones.

More quantitative considerations can be driven from fig. 5, fig. 6, and fig. 7 which depict the time evolution of the ROM L^2 error of the velocity, pressure, and eddy viscosity obtained with different combinations of modal truncation orders for the velocity field. Note that the L^2 relative error for a given quantity q is computed as follows:

$$\epsilon_q = \frac{||q_{FOM} - q_{ROM}||_{L^2(\Omega(t))}}{||q_{FOM}||_{L^2(\Omega(t))}} \times 100\%. \tag{63}$$

The results in fig. 5 confirm the qualitative impression of accuracy given by figs. 3 and 4, as the velocity field errors plotted remain well below the 1% threshold for the entire simulation. The plot also clearly indicates that the velocity field accuracy obtained making use of POD-NN is higher than that obtained with the POD-LSTM approach. This is further confirmed by fig. 6, in

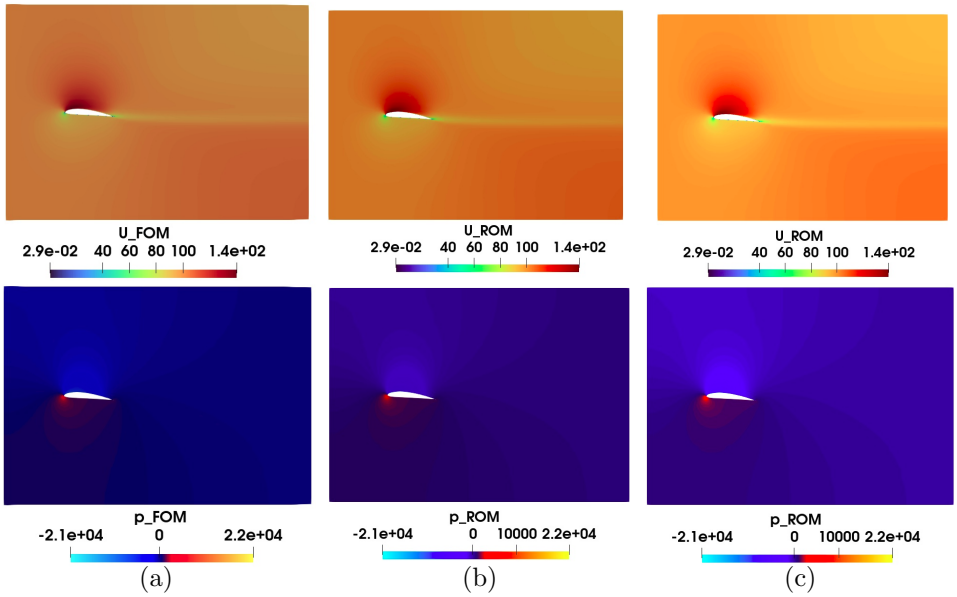


Figure 3: Comparison of the velocity and pressure fields. First row velocities comparison and second row pressure comparison. Column (a) FOM fields, column (b) reduced solution with POD-NNs and column (c) reduced solution with POD-LSTM. The snapshots are captured in the second period i.e. $t = T = 0.1$ s

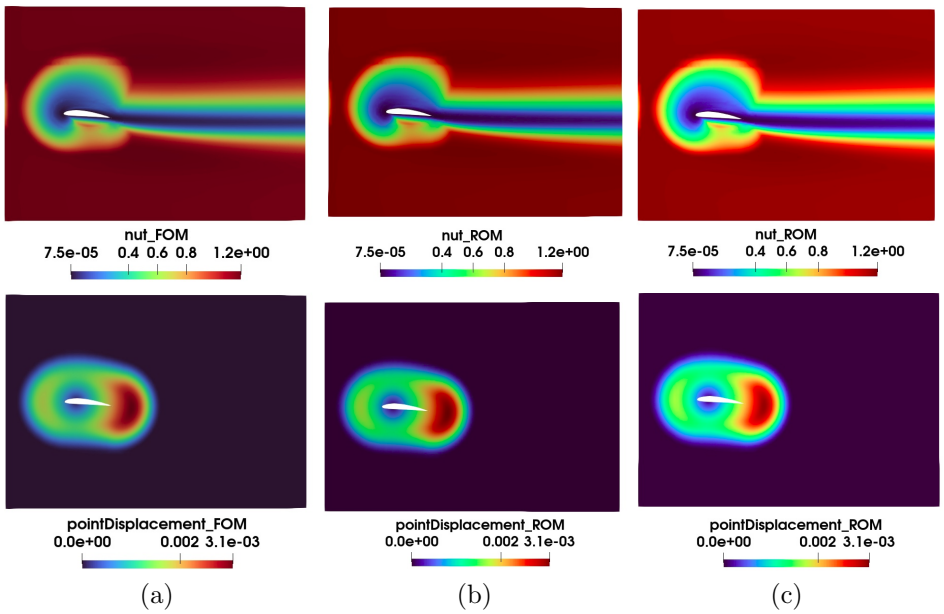


Figure 4: Comparison of the eddy viscosity and grid node displacement fields. First row eddy viscosity comparison and second row grid node displacement comparison. Column (a) FOM fields, column (b) reduced solution with POD-NNs and column (c) reduced solution with POD-LSTM. The snapshots are captured in the second period i.e. $t = T = 0.1$ s

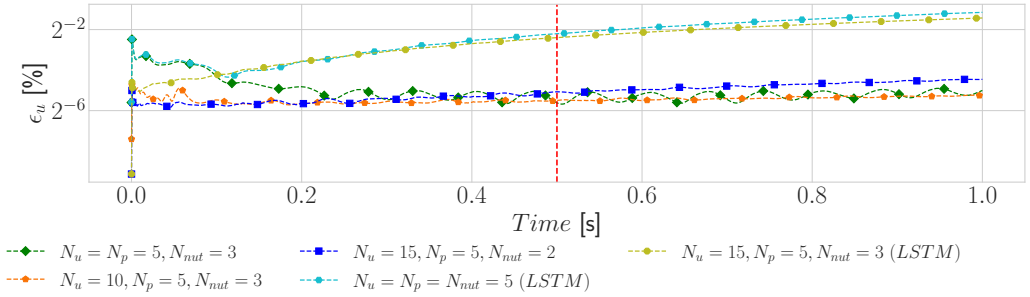


Figure 5: Sensitivity study of the error (log-scale) in the L^2-norm versus the time evolution of the velocity field. The red line shows the results obtained inside and outside the time window

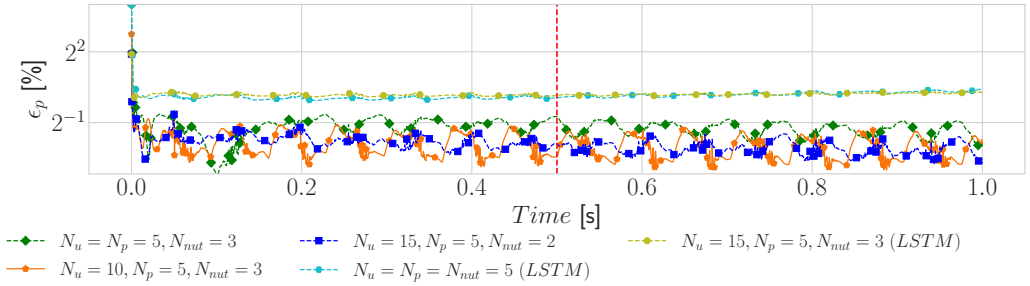


Figure 6: Sensitivity study of the error (log-scale) in the L^2-norm versus the time evolution of the pressure field. The red line shows the results obtained inside and outside the time window

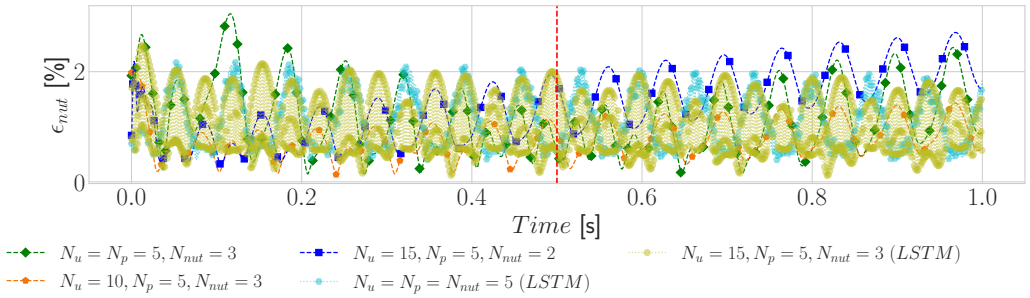


Figure 7: Sensitivity study of the error in the L^2-norm in log-scale versus the time evolution of the eddy viscosity field. The red line shows the results obtained inside and outside the time window

which the error associated with POD-LSTM is approximately as high as 1%, while the POD-NN approach results in appreciably lower error levels. The satisfactory results shown in these plots depend on the NN / LSTM algorithm effectiveness in the calculation of the eddy viscosity field POD coefficients time evolution. In this regard, it is worth pointing out that the fig. 7 error magnitude suggests that a 1-2% error level for the eddy viscosity field approximation, leads to consistently lower error values on the velocity and pressure fields. These observations seem to further validate the data driven approach taken for the eddy viscosity field, as it appears to lead to small errors of pressure and velocity fields, which are the main fields of interest for our simulations. Fig. 7 also shows that the error associated with the POD-LSTM approach presents a pattern characterized high frequency oscillations, which might be the culprit for the higher error levels observed in the velocity and pressure fields. This is consistent with what other researchers have observed in [5, 11, 55] in the prediction of the velocity field in a channel flow. In their study, they reported that the phenomenon was due to an insufficient number of snapshots of the training dataset or when the number of cells in the hidden layer was not enough. In addition to that, the lifetime of a transient turbulent state is highly sensitive to the initial conditions and if only one component of the initial conditions differs by 10^{-12}, the resulting trajectory will diverge from the truth one. It is also possible that the memory in the sequence can affect the LSTM accuracy as reported in [41] because signals originating from chaotic dynamic systems are known to have quite short correlated events and memory does not typically persist over long periods. In this case, the Hurst exponent is a prominent solution [22] for further investigations. However, the mentioned suggestions were not the emphasis of this paper. Instead, the emphasis was concentrated on the design of the overall solver to generate a reduced model for segregated FSI solvers for turbulent regime in the FV context.

The plots in fig. 5, fig. 6, and fig. 7 also visualize the effect of the number of velocity modes on the accuracy of the ROM solutions. In general, increasing the number of velocity modes considered at the online level increases the accuracy, especially in the initial transient part of the time integration. However, an increase to values higher than 10 modes does not appear to result in significant gains. It is finally important to point out that the snapshots for the POD have been collected only in a training window corresponding to the first half of the time history plotted — the one on the left of the red dashed vertical line in each plot. So, the plots also indicate that, in the case of a periodic problem as the one analyzed, the ROM errors are not significantly growing if time extrapolation is carried out.

A further step in the ROM results analysis is represented by the evaluation of the fluid dynamic forces and airfoil displacement accuracy. In fact, the L_2 errors discussed in the previous plots provide information on the average discrepancy of the most relevant fluid dynamic quantities in the fluid domain. However, the previous plots provide little information on the local distribution of such errors, which might have relevant impact on our FSI simulations. In fact, a low overall error in the pressure and velocity fields might still be in principle associated with high local error in small regions, for instance surrounding the airfoil. In such a case, both the fluid dynamic forces and the airfoil displacement might be computed with low accuracy.

Fig. 8 depicts the time history of the lift force exerted by the fluid on the airfoil. In the plot, the FOM solution is compared to the ones obtained with ROMs making use of different number of pressure and velocity modes. The plot suggests that both the ROM methodologies proposed are able to obtain qualitatively good approximation of the lift force throughout the time integration window considered, even including the initial transient. Also in this case, the plot indicates with a red dashed vertical line the separation between the training window, on the left, and the time window, on the right, in which the solution is extrapolating over the time variable. An inspection of the lift curves suggests that, when POD-NN is considered, no significant error increase is associated to time extrapolation. As for POD-LSTM, a slight degradation of the prediction quality is observed in the extrapolation region. Further confirmation of this is given by the corresponding error plots presented in fig. 9, in which it is possible to observe that for all the combination of pressure and velocity modes considered, the error remains below the 1% threshold, except for the initial transient, in which the $N_u = N_p = 5$ solution for both POD-NN and POD-LSTM presents a slightly higher error.

Similar plots relative to the airfoil drag are presented in fig. 10 and fig. 11. Also in this case, the figure presents a comparison between the FOM drag curve and the corresponding curves obtained with ROM models making use of different modal truncation orders. The plot suggests that the qualitative behavior of the airfoil resistance is well captured across all time steps of the

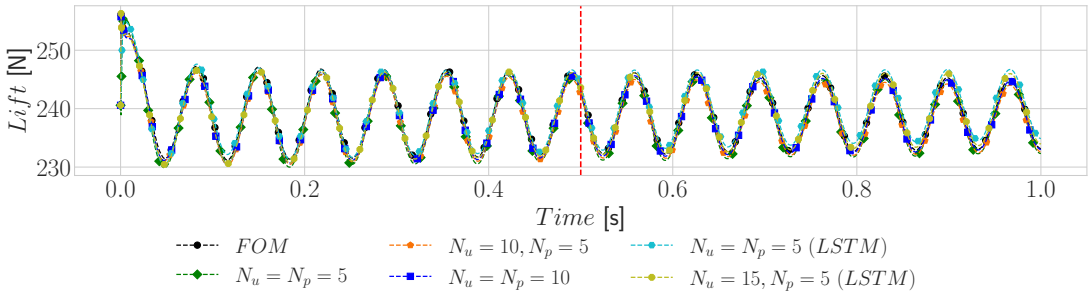


Figure 8: Times series comparison between the reference signal of the lift force acting on the foil in Newton unit with predicted signals. The vertical line in red divides the signal in two: left prediction in the time window and right prediction outside the time window (extrapolation)

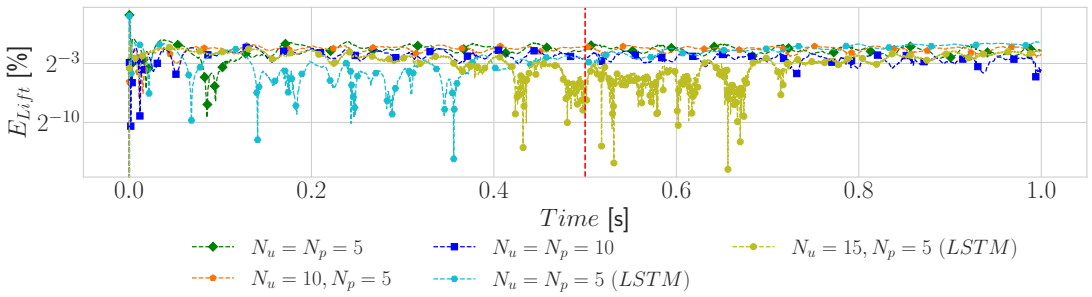


Figure 9: Times series of the error analysis of the lift force (original and predicted signals) from fig. 8. The vertical line in red divides the error plots in two. Left: interpolation error and right: extrapolation error

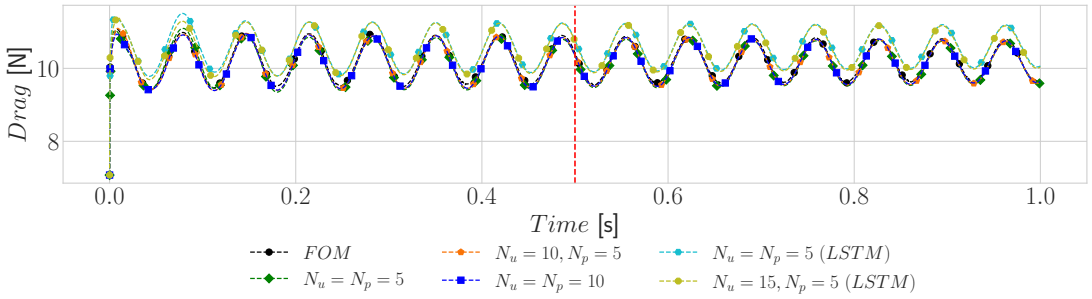


Figure 10: Times series comparison between the reference signal of the drag force acting on the foil in Newton unit with predicted signals. The vertical line in red divides the signal in two: left prediction in the time window and right prediction outside the time window (extrapolation)

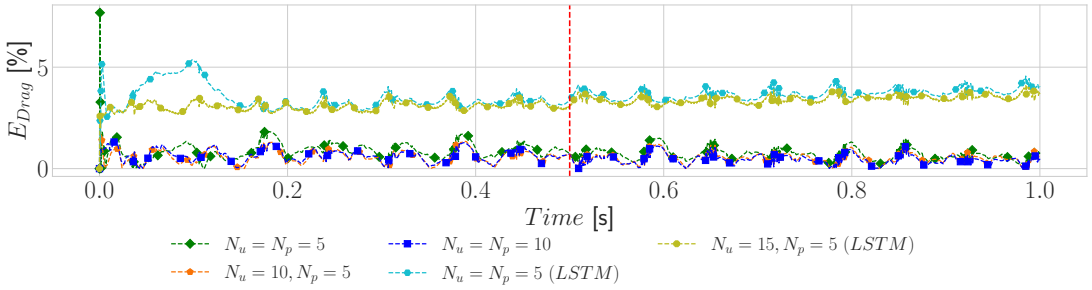


Figure 11: Times series of the error analysis of the drag force (original and predicted signals) from fig. 10. The vertical line in red divides the error plots in two. Left: interpolation error and right: extrapolation error

flow simulation, including the extrapolation time window. The value of the drag error obtained with POD-NN is again lower than 1% for the most part of the overall time history, although non-negligible portions of the error curve pass such a threshold. Such a higher relative error is just a product of the lower absolute value of the drag force with respect to the lift values. Thus, it can be said that the accuracy shown by the plots is quite remarkable, and is not degrading in the time extrapolation window. On the other hand, the error obtained with POD-LSTM appears to settle for higher values, although it remains below 5% for the time window analyzed, including the extrapolation region. Also, in this case, the increased error scale with respect to the lift should be associated with the smaller magnitude of the drag force absolute value and amplitude.

As a final confirmation of the proposed ROM results quality, it is important to also consider the time history of the airfoil displacements computed during the simulations. Fig. 12 depicts the airfoil center of gravity position, as computed at each time step by the FOM and by the ROM models proposed. The plot clearly indicates that all the reduced-order model solutions closely track the full-order one. For the most part of the time window — which as usual is divided into a training part and an extrapolation part by the red dashed line in the plot — the plunge coordinates curves obtained with all the POD-NN models considered appear in fact overlapped to the FOM one. The POD-LSTM results obtained making use of $N_u = 15$ velocity modes and $N_p = 5$ pressure modes have accuracy comparable with the POD-NN ones. Instead, the POD-LSTM curve associated with $N_u = N_p = 5$ visually appears less accurate, especially in the extrapolation region. The corresponding error plot is presented in fig. 13. The values reported in the diagram substantially confirm that the POD-NN error obtained with all the modal truncation combination considered, is in average as low as 0.1%, and always below the 1% threshold. Again, it has to be remarked that this satisfactory result does not appear to be negatively affected by time extrapolation, as the values in the second half of the plot remain generally low.

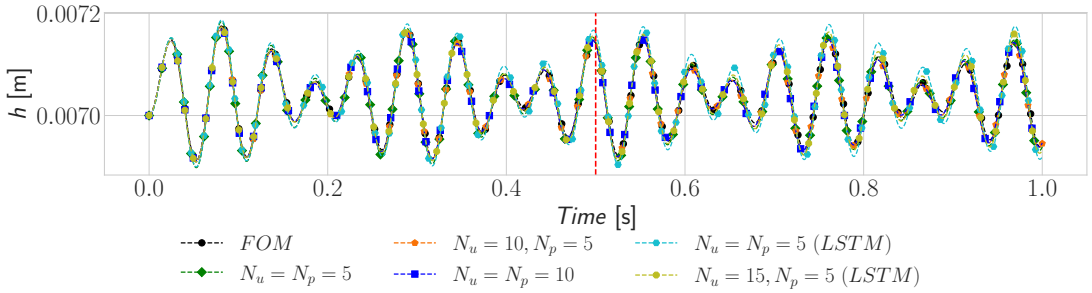


Figure 12: Time series comparison between the reference signal of the plunge with different predicted signals. The vertical line divides the signal in two: left prediction in the time window and right prediction outside the time window (extrapolation)

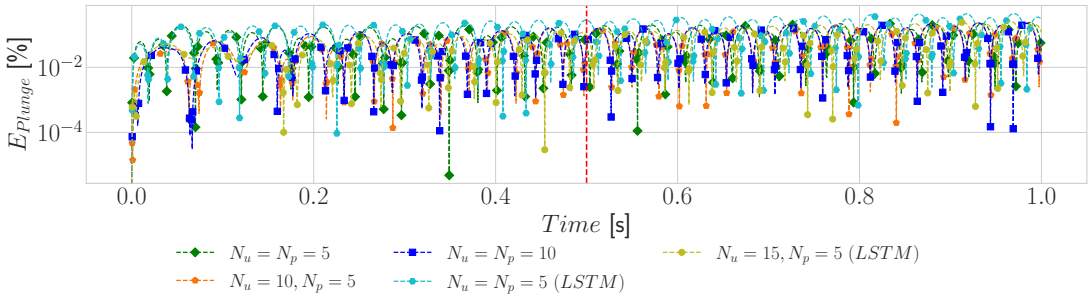


Figure 13: Plunge's error analysis versus time

Fig. 14 shows the time history of the airfoil pitch angle, as simulated with the FOM solver, and with ROM solvers making use of different eddy viscosity approximation strategies and different modal truncation orders. Here, all the ROM solvers tested — even the ones with lower truncation orders — lead to airfoil pitch curve approximation that are barely distinguishable from the FOM one. This is suggesting that not only the aerodynamic force, but also its point of application are reproduced with accuracy at the reduced order level. As a consequence, the pitch angle percentage error plot, presented in fig. 15 displays errors that are consistently below the 0.1% value throughout the entire simulation, for all the ROM models considered.

5 Conclusions and perspectives

This paper has proposed a hybrid projection-based ROM for segregated fluid-structure interaction (FSI) solvers in an ALE approach at a very high Reynolds number. The Finite Volume Method (FVM) has been used as discretization technique at the full-order level as the method ensures conversation properties and it is a preferred choice in industry for solving complex engineering problems as well as its ability to handle real-world geometries. This method is designed to work effectively with segregated solvers within the ALE framework at very high Reynolds numbers. As traditional methods can be computationally expensive, a reduced method has been introduced that combines a POD-Galerkin projection, POD-RBF, POD-NNs, and POD-LSTM as these methodologies have demonstrated immense potential in other research's fields. The resulting ROM framework proposed here is modular, hybrid, and data-driven; thus, it aligns with the development of a digital twin including fluid-structure interaction effects. The ROM is tested against a benchmark FSI problem at high Reynolds numbers to validate its accuracy and efficiency. By comparing the proposed ROM to the full-order model, the results indicated that the proposed hybrid ROM achieved acceptable accuracy, stability, and convergence. The proposed hybrid projection-based ROM offers a promising solution for efficiently solving segregated FSI problems in an ALE framework at high

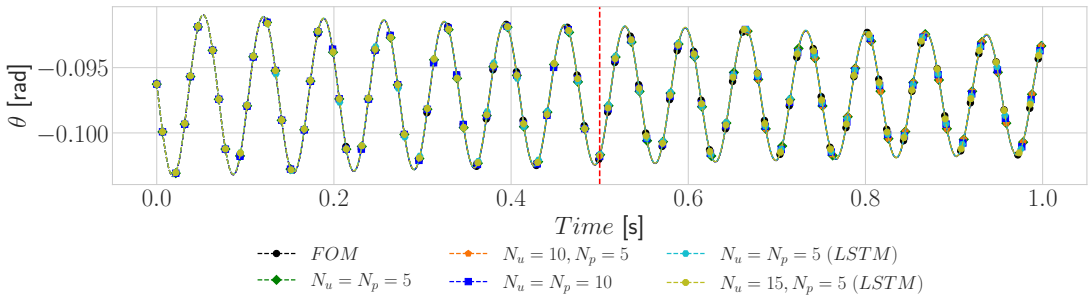


Figure 14: Time series comparison between the reference signal of the pitch (angle of attack) with different predicted signals. The vertical line in red divides the signal in two: left prediction in the time window and right prediction outside the time window (extrapolation)

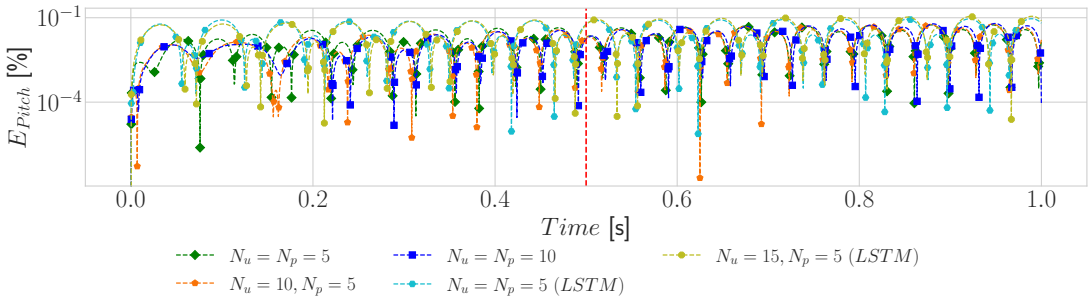


Figure 15: Pitch's error analysis versus time

Reynolds numbers without compromising accuracy. This approach could be particularly valuable for industries dealing with complex FSI problems, such as aerospace and automotive engineering, where computational resources are a critical concern. A natural direction for future work will include the construction of a robust LSTM encoder-decoder model with an attention layer to keep track of coefficient relationships within the inputs and output coefficients. Another avenue for further improvement is to equip the standard Galerkin projection with a hyper-reduction algorithm to boil down the computation time.

Acknowledgements

This work was partially funded by European Union Funding for Research and Innovation — Horizon 2020 Program — in the framework of European Research Council Executive Agency: H2020 ERC CoG 2015 AROMA-CFD project 681447 "Advanced Reduced Order Methods with Applications in Computational Fluid Dynamics" P.I. Professor Gianluigi Rozza, by PRIN "Numerical Analysis for Full and Reduced Order Methods for Partial Differential Equations" (NA-FROM-PDEs) project, by PRIN "Reduced Order Models for Environmental and Urban flows" (ROMEU), and by INdAM GNCS.

A Appendix

A.1 Machine learning of the temporal eddy viscosity coefficients

The main building blocks of the feed-forward NNs contain four layers: an input layer which is the *temporal vector coefficients of the velocity*, two hidden layers of 10 units each followed by an ELU (exponential linear unit) activation function, and the output layer which is the *temporal vector coefficients of the eddy viscosity*. The first 50% of the data in the time series is used for training (30%) and validation (20%) and the remaining for testing. The model is trained over 500 epochs.

For LSTM-RNNs, an LSTM Encoder-Decoder model is used for training, validation, and test on the data set. The architecture of such a LSTM-Encoder-Decoder model has one LSTM layer in the encoder part and another LSTM layer — in addition to one hidden layer of 5 units — in the decoder part. The reason for the choice is that this model has been used extensively for sequence-to-sequence prediction in the literature for NLP (natural language processing). The setup parameters of the LSTM are reported in table 4.

Parameter	Hidden size	learning rate	optimizer	LSTM layers	Batch size	Epochs	input dim	output dim
Value	1	1e-4	ADAM	1	5	500	5	5

Table 4: LSTM hyper-parameters

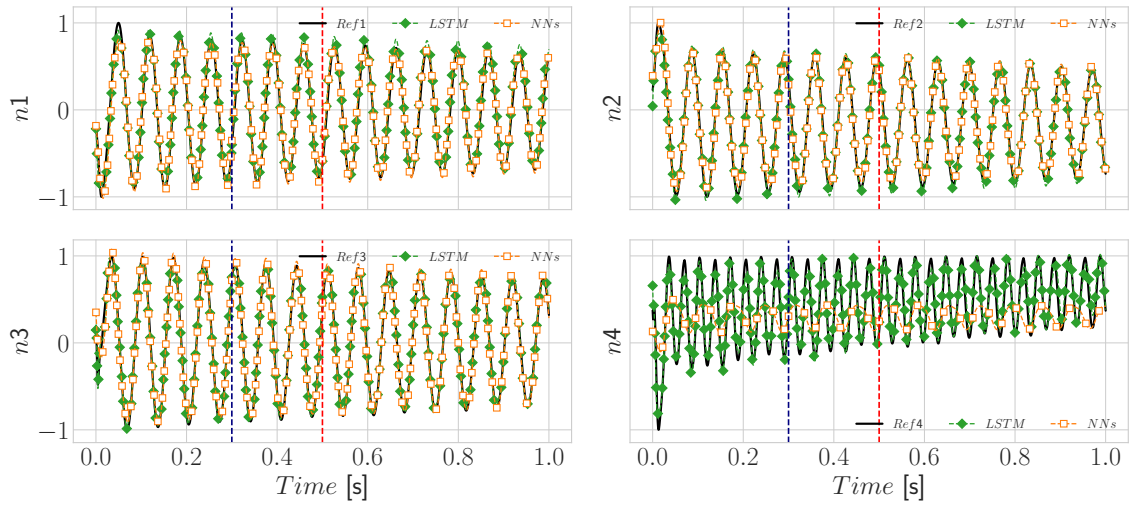


Figure 16: Training, validation, and test of the first four POD temporal coefficients of the eddy viscosity at the offline level using LSTM and NNs. The first half of the datasets (scaled between -1 and 1) is divided in two parts (30% for training and 20% for validation) for training and validation on both models. The remaining half is used for testing on both architectures

References

[1] J. S. Anttonen, P. I. King, and P. S. Beran. Applications of multi-POD to a pitching and plunging airfoil. *Mathematical and Computer Modelling*, 42(3-4):245–259, 2005.

[2] J. S. R. Anttonen. *Techniques for reduced order modeling of aeroelastic structures with deforming grids*. Air Force Institute of Technology, 2001.

[3] V. . Applications. *Volume3 Applications*. De Gruyter, Berlin, Boston, 2021.

[4] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[5] G. Borrelli, L. Guastoni, H. Eivazi, P. Schlatter, and R. Vinuesa. Predicting the temporal dynamics of turbulent channels through deep learning, 2022.

[6] S. R. Bukka. *Data-driven computing for the stability analysis and prediction of fluid-structure interaction*. PhD thesis, National University of Singapore (Singapore), 2020.

[7] G. Chen, Y. Zuo, J. Sun, and Y. Li. Support-vector-machine-based reduced-order model for limit cycle oscillation prediction of nonlinear aeroelastic system. *Mathematical Problems in Engineering*, 2012:1–12, 2012.

[8] M. H. Dao. Projection-based reduced order model for simulations of nonlinear flows with multiple moving objects. *arXiv preprint arXiv:2106.02338*, 2021.

[9] M. H. Dao and H. H. Nguyen. Projection-Based Reduced Order Model and Machine Learning Closure for Transient Simulations of High-Re Flows. *arXiv preprint arXiv:2105.10854*, 2021.

[10] A. Dullweber, B. Leimkuhler, and R. McLachlan. Symplectic splitting methods for rigid body molecular dynamics. *The Journal of chemical physics*, 107(15):5840–5851, 1997.

[11] H. Eivazi, L. Guastoni, P. Schlatter, H. Azizpour, and R. Vinuesa. Recurrent neural networks and koopman-based frameworks for temporal predictions in a low-order model of turbulence. *International Journal of Heat and Fluid Flow*, 90:108816, Aug. 2021.

[12] A. Falaize, E. Liberge, and A. Hamdouni. POD-based reduced order model for flows induced by rigid bodies in forced rotation. *Journal of Fluids and Structures*, 91:102593, Nov. 2019.

[13] B. A. Freno and P. G. Cizmas. A proper orthogonal decomposition method for nonlinear flows with deforming meshes. *International Journal of Heat and Fluid Flow*, 50:145–159, Dec. 2014.

[14] B. A. Freno, N. R. Matula, R. L. Fontenot, and P. G. Cizmas. The use of dynamic basis functions in proper orthogonal decomposition. *Journal of Fluids and Structures*, 54:332–360, Apr. 2015.

[15] B. Grimstad et al. SPLINTER: a library for multivariate function approximation with splines. http://github.com/bgrimstad/splinter, 2015. Accessed: 2015-05-16.

[16] G. Guennebaud, B. Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[17] R. Gupta and R. Jaiman. A hybrid partitioned deep learning methodology for moving interface and fluid-structure interaction. *Computers & Fluids*, 233:105239, 2022.

[18] K. C. Hall, J. P. Thomas, and W. S. Clark. Computation of unsteady nonlinear flows in cascades using a harmonic balance technique. *AIAA Journal*, 40(5):879–886, May 2002.

[19] J. Heaton. Ian goodfellow, yoshua bengio, and aaron courville: Deep learning. *Genetic Programming and Evolvable Machines*, 19(1-2):305–307, Oct. 2017.

[20] S. Hijazi, G. Stabile, A. Mola, and G. Rozza. Data-driven POD-Galerkin reduced order model for turbulent flows. *Journal of Computational Physics*, 416:109513, 2020.

[21] D. H. Hodges and G. A. Pierce. *Introduction to structural dynamics and aeroelasticity*, volume 15. cambridge university press, 2011.

[22] H. E. Hurst. Long-term storage capacity of reservoirs. *Transactions of the American society of civil engineers*, 116(1):770–799, 1951.

[23] R. I. Issa. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of computational physics*, 62(1):40–65, 1986.

[24] H. Jasak. *Error analysis and estimation for the finite volume method with applications to fluid flows.* PhD thesis, Imperial College London (University of London), 1996.

[25] H. Jasak. OpenFOAM: Open source CFD in research and industry. *International Journal of Naval Architecture and Ocean Engineering*, 1(2):89–94, Dec. 2009.

[26] H. Jasak, A. Jemcov, Z. Tukovic, et al. OpenFOAM: A C++ library for complex physics simulations. In *International workshop on coupled methods in numerical dynamics*, volume 1000, pages 1–20. IUC Dubrovnik Croatia, 2007.

[27] F. T. Johnson, E. N. Tinoco, and N. J. Yu. Thirty years of development and application of CFD at Boeing Commercial Airplanes, Seattle. *Computers & Fluids*, 34(10):1115–1151, Dec. 2005.

[28] W. Keiper, A. Milde, and S. Volkwein. *Reduced-order modeling (ROM) for simulation and optimization: powerful algorithms as key enablers for scientific computing.* Springer, 2018.

[29] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization, 2017.

[30] E. Komen, A. Shams, L. Camilo, and B. Koren. Quasi-DNS capabilities of OpenFOAM for different mesh types. *Computers & Fluids*, 96:87–104, 2014.

[31] J. Kou and W. Zhang. Layered reduced-order models for nonlinear aerodynamics and aeroelasticity. *Journal of Fluids and Structures*, 68:174–193, Jan. 2017.

[32] K. Kunisch and S. Volkwein. Galerkin proper orthogonal decomposition methods for a general equation in fluid dynamics. *SIAM Journal on Numerical Analysis*, 40(2):492–515, Jan. 2002.

[33] G. C. Lewin and H. Haj-Hariri. Reduced-order modeling of a heaving airfoil. *AIAA Journal*, 43(2):270–283, Feb. 2005.

[34] E. Liberge, M. Pomarede, and A. Hamdouni. Reduced-order modelling by pod-multiphase approach for fluid-structure interaction. *European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique*, 19(1-3):41–52, 2010.

[35] E. Longatte, E. Liberge, M. Pomarede, J.-F. Sigrist, and A. Hamdouni. Parametric study of flow-induced vibrations in cylinder arrays under single-phase fluid cross flows using POD-ROM. *Journal of Fluids and Structures*, 78:314–330, Apr. 2018.

[36] A. Mannarino and E. H. Dowell. Reduced-order models for computational-fluid-dynamics-based nonlinear aeroelastic problems. *AIAA Journal*, 53(9):2671–2685, Sept. 2015.

[37] A. Mannarino and P. Mantegazza. Nonlinear aerodynamic reduced order modeling by discrete time recurrent neural networks. *Aerospace Science and Technology*, 47:406–419, Dec. 2015.

[38] F. R. Menter, M. Kuntz, R. Langtry, et al. Ten years of industrial experience with the SST turbulence model. *Turbulence, heat and mass transfer*, 4(1):625–632, 2003.

[39] T. Miyanawala and R. Jaiman. A hybrid data-driven deep learning technique for fluid-structure interaction. *arXiv preprint arXiv:1807.09591*, 2018.

[40] T. Miyanawala and R. K. Jaiman. A hybrid data-driven deep learning technique for fluid-structure interaction. In *International Conference on Offshore Mechanics and Arctic Engineering*, volume 58776, page V002T08A004. American Society of Mechanical Engineers, 2019.

[41] A. T. Mohan and D. V. Gaitonde. A deep learning based approach to reduced order modeling for turbulent flow control using LSTM neural networks. *arXiv preprint arXiv:1804.09269*, 2018.

[42] F. Moukalled, L. Mangani, M. Darwish, F. Moukalled, L. Mangani, and M. Darwish. *The finite volume method*. Springer, 2016.

[43] V. N. Ngan, G. Stabile, A. Mola, and G. Rozza. A reduced-order model for segregated fluid-structure interaction solvers based on an ale approach, 2023.

[44] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[45] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library, 2019.

[46] S. V. Patankar and D. B. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. In *Numerical prediction of flow, heat transfer, turbulence and combustion*, pages 54–73. Elsevier, 1983.

[47] R. Pigazzini, G. Contento, S. Martini, T. Puzzer, M. Morgut, and A. Mola. Viv analysis of a single elastically-mounted 2d cylinder: Parameter identification of a single-degree-of-freedom multi-frequency model. *Journal of Fluids and Structures*, 78:299–313, 2018.

[48] A. Placzek. *Construction de modèles d'ordre réduit non-linéaires basés sur la décomposition orthogonale propre pour l'aéroélasticité*. PhD thesis, Conservatoire national des arts et metiers-CNAM, 2009.

[49] C. M. Rhie and W. L. Chow. Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA Journal*, 21(11):1525–1532, Nov. 1983.

[50] C.-C. Rossow and N. Kroll. High performance computing serves aerospace engineering: Opportunities for next generation product development, 2008. Last accessed 10.01.2008.

[51] H. Sak, A. Senior, and F. Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition, 2014.

[52] M. Shademan, R. Barron, and R. Balachandar. Evaluation of openfoam in academic research and industrial applications. In *21st Conference of the CFD Society of Canada*, page 7, 2013.

[53] V. Shinde, E. Longatte, F. Baj, Y. Hoarau, and M. Braza. Galerkin-free model reduction for fluid-structure interaction using proper orthogonal decomposition. *Journal of Computational Physics*, 396:579–595, 2019.

[54] D. B. Spalding. A novel finite difference formulation for differential expressions involving both first and second derivatives. *International Journal for Numerical Methods in Engineering*, 4(4):551–559, July 1972.

[55] A. Srinivasan and P. Anand. Deep Learning models for turbulent shear flow, 2018.

[56] G. Stabile, S. Hijazi, A. Mola, S. Lorenzi, and G. Rozza. POD-Galerkin reduced order methods for CFD using Finite Volume Discretisation: vortex shedding around a circular cylinder. *Communications in Applied and Industrial Mathematics*, 8(1):210–236, 2017.

[57] G. Stabile and G. Rozza. Finite volume POD-Galerkin stabilised reduced order methods for the parametrised incompressible Navier–Stokes equations. *Computers & Fluids*, 173:273–284, 2018.

[58] W. Stankiewicz, M. Morzyński, R. Roszak, B. Noack, and G. Tadmor. Reduced order modeling of a flow around an airfoil with a changing angle of attack. *Archives of Mechanics*, 60(6):509–526, 2008.

[59] W. Stankiewicz, R. Roszak, and M. Morzyński. Arbitrary Lagrangian-Eulerian approach in reduced order modeling of a flow with a moving boundary. *Progress in Flight Physics*, 5:109–124, 2013.

[60] W. Stankiewicz, R. Roszak, and M. Morzyński. Arbitrary lagrangian-eulerian approach in reduced order modeling of a flow with a moving boundary. In P. Reijasse, D. Knight, M. Ivanov, and I. Lipatov, editors, *Progress in Flight Physics*. EDP Sciences, June 2013.

[61] J. P. Thomas, E. H. Dowell, and K. C. Hall. Using automatic differentiation to create a nonlinear reduced-order-model aerodynamic solver. *AIAA Journal*, 48(1):19–24, Jan. 2010.

[62] V. Troshin, A. Seifert, D. Sidilkover, and G. Tadmor. Proper orthogonal decomposition of flow-field in non-stationary geometry. *Journal of Computational Physics*, 311:329–337, Apr. 2016.

[63] Y.-Y. Tsui, Y.-C. Huang, C.-L. Huang, and S.-W. Lin. A finite-volume-based approach for dynamic fluid-structure interaction. *Numerical Heat Transfer, Part B: Fundamentals*, 64(4):326–349, 2013.

[64] Z. Wang, L. Du, J. Zhao, M. C. Thompson, and X. Sun. Flow-induced vibrations of a pitching and plunging airfoil. *Journal of Fluid Mechanics*, 885, Jan. 2020.

[65] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[66] M. J. Whisenant and K. Ekici. Galerkin-Free Technique for the Reduced-Order Modeling of Fluid-Structure Interaction via Machine Learning. In *AIAA Scitech 2020 Forum*, page 1637, 2020.

[67] M. Zancanaro, M. Mrosek, G. Stabile, C. Othmer, and G. Rozza. Hybrid neural network reduced order modelling for turbulent flows with geometric parameters. *Fluids*, 6(8):296, 2021.