



POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Computational Fluid Dynamics by Artificial Intelligence: a quasi- unsupervised deep learning ap- proach

TESI DI LAUREA MAGISTRALE IN
MATHEMATICAL ENGINEERING - COMPUTATIONAL SCIENCE
AND COMPUTATIONAL LEARNING

Author: **Nicole Baggio**

Student ID: 245645

Advisor: Prof. Stefano Mariani

Co-advisors: Paolo Marcandelli

Academic Year: 2024-25

Abstract

Computational Fluid Dynamics provides accurate descriptions of complex flow phenomena, yet its computational cost may significantly constrain applications requiring repeated simulations or real-time predictions. This thesis investigates a quasi-unsupervised deep learning strategy for incompressible flows, with particular attention to simulations on irregular meshes. The methodology is based on a graph neural network and incorporates the discrete Navier–Stokes residuals into the loss function. Spatial differential operators are approximated on unstructured node sets through Radial Basis Function Finite Difference (RBF-FD) techniques. The methodology is evaluated on laminar channel flow, obstacle flows characterized by wake formation, and a representative configuration with moving boundaries. In laminar conditions, hybrid supervision combined with residual constraints recovers velocity and pressure fields with accuracy comparable to fully supervised learning while reducing dependence on labelled data. In unsteady regimes arising from obstacle flows, the quasi-unsupervised configuration remains competitive with supervised training and generalizes across geometrically different configurations within the same physical setting. More strongly coupled dynamics can be captured under full supervision, whereas configurations with reduced supervision fail to reproduce the correct qualitative interaction, despite maintaining bounded numerical errors. A realistic assessment of computational cost highlights that surrogate modeling remains dependent on CFD-generated datasets, and that the required training effort may approach or even exceed the cost of the underlying simulations. While inference is computationally efficient and a single trained model can be applied across multiple configurations without retraining, the framework cannot yet be regarded as a definitive replacement for traditional CFD workflows. Overall, the proposed quasi-unsupervised deep learning approach enhances robustness and geometric adaptability in controlled incompressible flow regimes, although further improvements in physical generalization and computational efficiency are necessary for practical deployment.

Keywords: computational fluid dynamics, graph neural networks, physics-informed learning, Navier-Stokes equations, incompressible flows.

Abstract in lingua italiana

La Fluidodinamica Computazionale fornisce descrizioni accurate di flussi complessi; tuttavia, il suo costo computazionale può limitare applicazioni che richiedono simulazioni ripetute o previsioni in tempo reale. Questa tesi studia una strategia di deep learning quasi non supervisionato per flussi incomprimibili, con particolare attenzione a simulazioni su mesh non strutturate. La metodologia si basa su un'architettura neurale su grafi e introduce un addestramento informato dalla fisica, in cui i residui discreti delle equazioni di Navier–Stokes sono inclusi nella funzione di perdita. Gli operatori differenziali spaziali sono approssimati mediante tecniche Radial Basis Function Finite Difference (RBF-FD). Il metodo è valutato su flusso laminare in canale, su configurazioni con ostacolo caratterizzate da formazione di scia e su un caso con contorni mobili. In regime laminare, una supervisione ibrida con vincoli fisici ricostruisce i campi di velocità e pressione con accuratezza comparabile al pieno apprendimento supervisionato, riducendo la dipendenza da dati etichettati. Nei regimi non stazionari con ostacolo, la configurazione quasi non supervisionata resta competitiva e generalizza tra geometrie differenti nello stesso contesto fisico. Dinamiche più fortemente accoppiate sono catturate solo con piena supervisione, mentre una supervisione ridotta non riproduce correttamente l'interazione qualitativa, pur mantenendo errori numerici limitati. Una valutazione realistica del costo computazionale mostra che la modellazione surrogata dipende ancora da dataset CFD e che l'addestramento può eguagliare o superare il costo delle simulazioni di riferimento. Sebbene l'inferenza sia efficiente e un singolo modello possa essere applicato a più configurazioni senza riaddestramento, il framework non rappresenta ancora un sostituto definitivo dei tradizionali flussi di lavoro CFD. Nel complesso, l'approccio proposto migliora robustezza e adattabilità geometrica in regimi controllati di flusso incomprimibile, ma richiede ulteriori progressi in termini di generalizzazione fisica ed efficienza computazionale per un impiego pratico.

Parole chiave: fluidodinamica computazionale , graph neural networks, physics-informed learning, equazioni di Navier-Stokes, fluidi incomprimibili.

Contents

Abstract	i
Abstract in lingua italiana	iii
Contents	v
1 Introduction	1
2 Mathematical and Numerical Preliminaries	7
2.1 Incompressible Navier-Stokes Equations	8
2.2 OpenFOAM and the Numerical Solution of Navier-Stokes Equations	12
2.3 Numerical setup of the channel flow in OpenFOAM	14
3 Neural Models for PDE-Based Simulations	21
3.1 Neural Networks	21
3.2 MeshGraphNet	23
3.2.1 Node and Edge Encoders	25
3.2.2 Processor: Message Passing Layers	26
3.2.3 Decoder	29
3.3 Physics-Informed Neural Networks	30
3.3.1 Discrete Physics-Informed Losses	32
3.4 Radial Basis Functions and RBF-FD Discretization	34
3.4.1 Geometric Quality: Fill Distance and Separation Distance	37
3.4.2 Differential Operators Used in This Work	38
3.5 Conclusions and next steps	40
4 Channel Flow	41
4.1 Setup	42
4.2 OpenFOAM simulation	43
4.3 Results	45

4.3.1	Supervised Case	45
4.3.2	Generalization across meshes	53
4.3.3	Partially unsupervised Case	56
4.3.4	Fully Unsupervised Case	60
5	Channel with an obstacle	67
5.1	OpenFOAM setup and boundary conditions	68
5.2	Training methodology and learning strategies	72
5.2.1	Multi-phase training	72
5.2.2	Temporal rollout	76
5.3	Laminar and steady state square obstacle flow ($Re = 20$)	77
5.4	Transitional cyrcular obstacle flow ($Re = 200$)	83
5.4.1	Comparison supervised vs unsupervised	89
5.5	Generalization for a larger dataset	90
5.5.1	Limits	93
5.6	Transitional airfoil obstacle flow ($Re = 2000$)	96
5.7	Turbulent square obstacle flow ($Re = 2 \times 10^6$)	99
6	Extension to Fluid-Solid Interaction	103
6.1	Fluid-Structure Interaction: mathematical framework	104
6.1.1	Arbitrary Lagrangian–Eulerian formulation	105
6.1.2	Hron-Turek FSI3 benchmark	106
6.2	Supervised case	108
6.2.1	Modeling strategy and learning setup	108
6.2.2	Results	110
6.3	Partially unsupervised case	112
6.3.1	Modeling strategy and geometric treatment	112
6.3.2	Results	114
6.4	Discussion and concluding remarks	118
7	Scalability Limits of Message Passing Graph Neural Networks	119
8	Conclusions	125
	Bibliography	127

List of Figures	133
List of Tables	141
List of Symbols	143
Acknowledgements	145

1 | Introduction

Computational Fluid Dynamics (CFD) plays a fundamental role in a wide range of scientific and engineering applications [1], spanning aerospace and automotive design, energy production, environmental modeling and biomedical engineering. Across these domains, CFD is routinely employed to analyze complex flow phenomena and to support tasks such as aerodynamic shape optimization, performance assessment of energy systems, prediction of pollutant dispersion in urban environments, simulation of flow through porous media and infiltration processes, and the investigation of blood flow and fluid-structure interaction in patient-specific arterial geometries [2–6].

Many practically relevant problems can be abstracted into canonical flow configurations, such as flows in channels or flows around obstacles. These settings serve as standard benchmarks in fluid dynamics, as they capture key physical mechanisms, including boundary layer development, flow separation, vortex shedding and the transition to turbulence, while remaining sufficiently controlled to allow systematic analysis. For this reason, canonical configurations are widely used in parametric studies, optimization loops and uncertainty quantification frameworks, where large ensembles of simulations are required to estimate mean quantities, fluctuations and sensitivity measures [7, 8]. Insights gained from such canonical configurations are routinely used as building blocks for the interpretation and modeling of more complex, real-world flows.

Despite its widespread success, computational fluid dynamics (CFD) remains computationally expensive. At its core, CFD relies on the numerical solution of the Navier-Stokes equations, which describe the conservation of linear momentum of the fluid and are coupled with the continuity equation enforcing mass conservation.

The nonlinear inertial term, corresponding to the advection of momentum (i.e. the velocity field multiplying its own spatial gradient), governs the transfer of kinetic energy across scales and is responsible for complex flow phenomena such as separation, vortex shedding, and turbulence.

At high Reynolds numbers, fluid flows exhibit a multiscale structure characterized by the coexistence of large energy-containing eddies and much smaller dissipative structures [9].

The presence of widely separated interacting scales increases the numerical complexity of the problem: accurately capturing the smallest dynamically relevant features requires sufficiently fine spatial and temporal resolution, which can substantially raise the computational cost. Although multiscale modeling strategies such as Large Eddy Simulation or adaptive refinement techniques [10] can alleviate this burden by selectively resolving only part of the flow spectrum, the intrinsic scale separation remains a major source of computational difficulty.

These requirements lead to substantial computational costs, which can become prohibitive in applications that require repeated flow evaluations, such as aerodynamic shape optimization, parameter calibration in biomedical flows, or uncertainty quantification studies.

In recent decades, various strategies have been developed to alleviate the computational burden of high-fidelity CFD simulations, including reduced-order modeling, multiscale discretization techniques, and projection-based model reduction [11, 12]. More recently, machine learning approaches have emerged as a promising complementary direction to further accelerate flow prediction and surrogate modeling [13, 14]. Data-driven surrogate models have shown the ability to approximate complex flow dynamics with significantly reduced computational effort, achieving good accuracy across a range of canonical and turbulent flow configurations [15, 16]. This has sparked growing interest in using machine learning to accelerate CFD simulations or to replace selected components of classical numerical pipelines.

Despite these advances, purely data-driven approaches suffer from notable limitations. Their performance depends strongly on large, representative training datasets, which are typically generated through costly CFD simulations, thereby shifting rather than removing the computational burden. In addition, supervised models often exhibit limited generalization to unseen geometries, mesh resolutions, or flow regimes [17, 18]. These challenges highlight a fundamental limitation of purely data-driven models: while they may accurately interpolate observed solutions, they do not inherently encode the physical principles governing fluid motion. In the absence of explicit physical constraints, purely data-driven neural networks may produce predictions that match training data locally but fail to preserve fundamental physical principles such as conservation laws or stability properties, especially when evaluated outside the training distribution [19].

To address these issues, physics-informed machine learning [20] has emerged as a paradigm that integrates the governing physical laws directly into the learning process [21, 22]. Rather than relying exclusively on labeled data, physics-informed approaches constrain neural networks using the underlying partial differential equations, boundary conditions

and conservation principles. This physics-based regularization enables training without explicit supervision and improves generalization when extrapolating in space or time [23]. In fluid dynamics, this typically corresponds to embedding the Navier-Stokes equations into the training objective through residual-based loss terms.

While physics-informed neural networks (PINNs) have shown promising results in data-scarce settings, they often struggle to achieve high accuracy in complex flow scenarios, particularly in turbulent regimes or large-scale domains. These limitations are commonly attributed to training instabilities and to the stiffness of the underlying PDE constraints when embedded into the training objective, which can lead to poorly conditioned loss landscapes and imbalanced gradients [24, 25].

Many existing machine-learning approaches for CFD, including both purely data-driven models and several physics-informed formulations, are based on convolutional neural networks and grid-based representations of the flow fields [26, 27]. While these architectures have proven effective on structured domains, they inherently assume regular grids and fixed local connectivity patterns. As a consequence, their application to unstructured meshes or geometrically complex domains typically requires interpolation or remeshing onto Cartesian grids, which may introduce geometric distortions, reduce resolution near curved boundaries, and limit physical consistency, especially in the vicinity of solid obstacles [28].

These limitations become particularly relevant in fluid dynamics problems characterized by complex geometries and topological variations, where an accurate and smooth representation of curved or circular obstacles is essential to correctly capture boundary layers, flow separation, and wake dynamics.

In this context, graph neural networks (GNNs) have emerged as a powerful framework for learning surrogate models of physical systems defined on unstructured domains [29, 30]. By operating directly on mesh-based or point-cloud representations, graph-based models naturally adapt to irregular discretizations and complex boundaries, without the need for interpolation onto regular grids. By explicitly encoding the computational mesh and its connectivity, GNNs allow the learning process to respect the geometric structure of the domain and to generalize across different discretizations and geometries [31].

In this thesis, we focus on the modeling of two-dimensional, incompressible flows in canonical configurations, with particular emphasis on channel flows and flows around solid obstacles. These configurations provide a controlled yet physically rich setting, in which essential flow features such as boundary layer development, flow separation, vortex shedding and wake dynamics naturally arise.

The concrete objective of this thesis is to develop a data-efficient and physically consistent surrogate model capable of predicting both stationary solutions and the time evolution of incompressible fluid flows on irregular meshes, while remaining stable over long temporal horizons in the transient regime. Rather than replacing traditional solvers entirely, the goal is to investigate whether embedding physical constraints into a graph-based learning framework can improve robustness, generalization and long-term stability compared to purely supervised approaches.

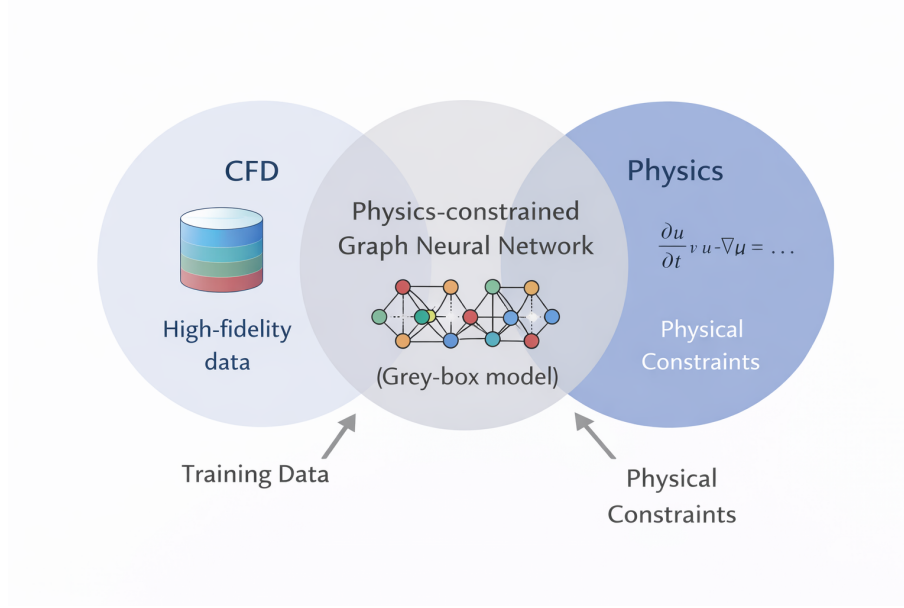


Figure 1.1: Schematic representation of a grey-box modeling approach combining high-fidelity CFD data and physical constraints within a graph neural network framework.

To this end, we propose a quasi-unsupervised graph neural network framework based on a MeshGraphNet architecture [29]. Figure 1.1 provides a conceptual overview of this intermediate modeling strategy, in which data-driven learning is complemented by physical constraints in order to achieve more physically consistent and robust flow predictions.

The evaluation of physical residuals requires the approximation of spatial differential operators, which is non-trivial on unstructured meshes represented as graphs. In this work, spatial derivatives are computed using Radial Basis Function–Finite Difference (RBF-FD) methods [32], which allow differential operators to be evaluated directly on irregular point clouds. This choice ensures consistency between the numerical operators used to enforce physical constraints and the graph-based representation employed by the neural network.

In the following chapters, we first introduce the mathematical background of incompressible flow modeling, together with a brief overview of the numerical methods used to solve

the Navier–Stokes equations and to generate the reference datasets using the OpenFOAM framework (see Chapter 2). We then present the neural and graph-based models adopted in this work, with particular emphasis on the MeshGraphNet architecture and on the use of Radial Basis Function–Finite Difference (RBF-FD) methods to approximate spatial differential operators on unstructured domains (see Chapter 3). The proposed methodology is subsequently assessed on a range of flow configurations of increasing complexity, starting from simple channel flows (see Chapter 4) and progressively moving toward more challenging scenarios involving obstacles and vortex-dominated dynamics, in order to evaluate both predictive accuracy and generalization capabilities (see Chapter 5). The framework is then extended to coupled multi-physics problems through the consideration of fluid–structure interaction (FSI), where the ability of the model to capture the interplay between fluid and structural dynamics is investigated (see Chapter 6). Finally, we examine the scalability of the proposed approach with respect to increasing spatial resolution, highlighting structural limitations of message passing graph neural networks and discussing their implications for PDE learning (see Chapter 7). The thesis concludes with a critical summary of the main findings, limitations, and perspectives for future research (see Chapter 8).

Beyond benchmarking predictive accuracy, particular attention is devoted to the assessment of temporal rollouts, in order to evaluate whether the proposed framework can autonomously propagate flow dynamics over extended horizons rather than relying solely on one-step supervised predictions. This aspect represents a key step toward greater dynamical consistency compared to purely snapshot-based approaches. Overall, the proposed grey-box graph based framework improves robustness on irregular meshes compared to purely supervised models, but still exhibits limitations in long horizon rollouts, where progressive error accumulation can degrade the preservation of coherent flow structures.

2 | Mathematical and Numerical Preliminaries

In this chapter, we introduce the mathematical model and the numerical framework employed to generate the reference datasets used throughout this work. These datasets are an essential component of the proposed methodology, as they provide both a quantitative benchmark for the evaluation of the learned models and the supervised information required during specific training phases.

The physical problem considered as a baseline in this study is the simulation of incompressible fluid flow in a confined channel configuration. While such geometries naturally arise in applications including microfluidic and bioengineering devices, where the flow is bounded by rigid walls and controlled boundary conditions are imposed at the inlet and outlet, they are employed here primarily as a canonical benchmark to assess the predictive and dynamical performance of the proposed framework in a controlled setting. An example of a microfluidic channel in glass device is shown in Fig. 2.1.

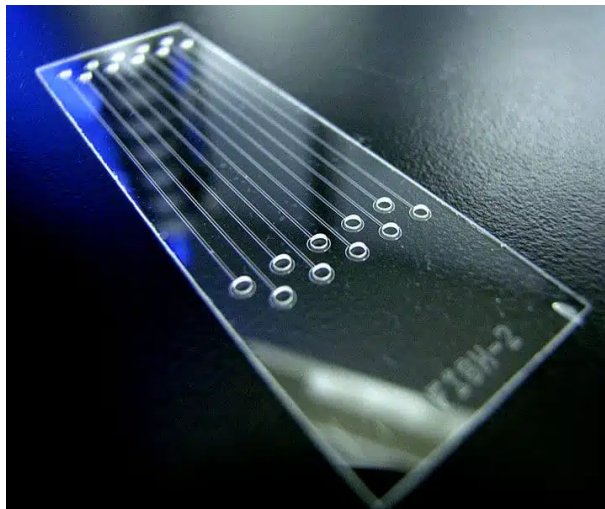


Figure 2.1: Example of a microfluidic channel device with rigid confining walls. The present study considers an idealised two dimensional approximation of a confined channel, abstracting from three dimensional effects and microscale specific phenomena.

In the present work, however, the geometry is idealised as two dimensional, assuming negligible variations in the out of plane direction and focusing on the governing incompressible Navier-Stokes dynamics. This choice allows the investigation of flow regimes ranging from steady laminar conditions to unsteady and transitional dynamics.

Although the governing dynamics of the problem are fully described by the incompressible Navier-Stokes equations, explicit analytical solutions are only available for a very limited class of highly idealized configurations. In finite domains with prescribed inlet and outlet conditions, and in the presence of flow instabilities, the resulting system of equations generally cannot be solved in closed form. As a consequence, the study of realistic channel flows necessarily relies on numerical approximation techniques. For this reason, the reference flow fields used in this work are obtained by means of a numerical solver. In particular, the open-source computational fluid dynamics framework OpenFOAM is employed to compute accurate numerical solutions of the incompressible Navier-Stokes equations, which serve as high-fidelity approximations of the underlying physical problem.

The chapter is organized as follows. In Section 2.1 we introduce the incompressible Navier-Stokes equations and discuss their physical and mathematical properties, with particular emphasis on the limitations of analytical approaches. In Section 2.2 we present the numerical discretization strategy and the solver adopted in this work, namely OpenFOAM. Finally, in Section 2.3 we describe how the solver is employed to generate datasets for the specific channel flow configurations considered in this study, detailing the numerical setup and simulation parameters used for the different flow regimes.

2.1. Incompressible Navier-Stokes Equations

Fluid motion is governed by the fundamental Navier-Stokes equations, formalized in the early to mid-19th century, and named after the two mathematicians Claude-Louis Navier and George Gabriel Stokes, who independently contributed to the formulation [33]. The incompressible Navier-Stokes (NS) equations express the fundamental principles of conservation of mass and momentum governing fluid motion. They provide the mathematical framework for describing the temporal evolution of a viscous incompressible fluid occupying a spatial domain $\Omega \subset \mathbb{R}^2$, where the two-dimensional assumption is adopted in the present work.

The primary unknowns of the problem are the velocity field $\mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^2$ and the scalar pressure field $p(\mathbf{x}, t) \in \mathbb{R}$, where $\mathbf{x} = (x, y) \in \Omega$ denotes the spatial coordinate and $t \geq 0$ represents time:

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \mu \nabla^2 \mathbf{u}, \quad (2.1)$$

$$\nabla \cdot \mathbf{u} = 0 \quad (2.2)$$

where ρ denotes the fluid density and μ the dynamic viscosity. We adopt the standard notation ∇ for the gradient in $\mathbb{R}^2$, with divergence $\nabla \cdot$ and Laplacian ∇^2 , the latter acting component-wise on vector fields.

The first equation represents the conservation of linear momentum, while the second enforces the incompressibility constraint by requiring the velocity field to be divergence-free.

From a physical standpoint, incompressibility corresponds to the assumption that the density of each fluid particle remains constant in time, independently of the applied state of stress [34]. This assumption implies that mass conservation does not allow for local volume changes of fluid elements and leads directly to the divergence-free condition imposed on the velocity field.

The conservation of mass is expressed by the continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \quad (2.3)$$

For incompressible flows, the density is constant, $\rho = \text{const.}$ Under this assumption, the continuity equation reduces to the kinematic constraint

$$\nabla \cdot \mathbf{u} = 0 \quad (2.4)$$

which enforces local conservation of mass without volumetric expansion or compression.

In Eq.(2.1), the nonlinear convective term $\mathbf{u} \cdot \nabla \mathbf{u}$ makes the NS equations complex and potentially turbulent. This term represents inertial effects, while the viscous forces are associated with the diffusive term $\mu \nabla^2 \mathbf{u}$.

A nondimensional quantity used to characterize the balance between inertial and viscous forces is the Reynolds number:

$$Re = \frac{\rho U L}{\mu}, \quad (2.5)$$

where U is a characteristic velocity magnitude of the flow and L a characteristic length

scale. The Reynolds number provides a compact measure of the balance between inertial forces, which tend to amplify velocity fluctuations in space and promote flow instabilities, and viscous forces, which act to damp velocity gradients and suppress disturbances. As a result, variations in the Reynolds number have a direct impact on the qualitative structure of the flow and on the dynamical regimes that can be observed [9].

At sufficiently low Reynolds numbers, viscous effects dominate and the flow remains laminar. In this regime, fluid motion is smooth and orderly, velocity fluctuations are absent, and the flow field is typically steady or weakly time-dependent. For simple geometries such as straight channels, laminar flows often admit analytical or semi-analytical solutions, such as the plane Poiseuille flow, discussed later in this section.

As the Reynolds number increases, inertial effects become increasingly important and the flow transitions from a laminar regime to more complex, unsteady dynamics. The transitional regime is characterized by the onset of flow instabilities, the appearance of localized vortical structures and an increased sensitivity to perturbations and boundary conditions. In this range the flow dynamics become more complex and strongly dependent on the geometry of the domain.

At high Reynolds numbers, inertial forces dominate over viscous dissipation and the flow typically becomes turbulent. Turbulent flows are characterized by broadband velocity fluctuations, strong nonlinear interactions across a wide range of spatial and temporal scales, and enhanced mixing and momentum transport. In this regime, no exact analytical solutions are available and the flow must be described using statistical quantities or resolved numerically.

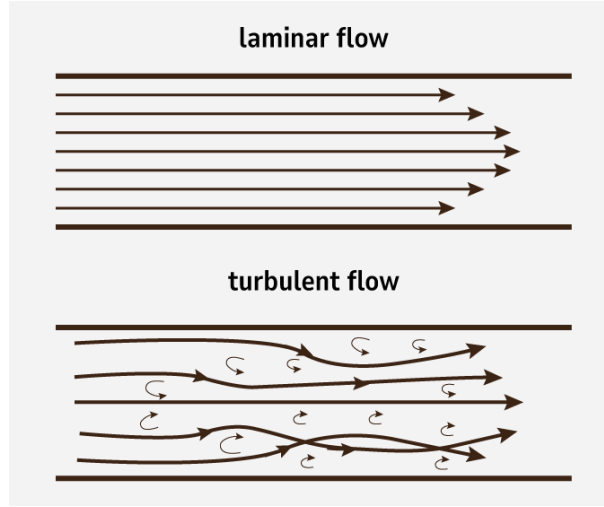


Figure 2.2: Qualitative comparison between laminar and turbulent flow regimes: laminar flow (top) and turbulent flow (bottom).

Figure 2.2 illustrates the qualitative differences between laminar and turbulent flow regimes.

For internal flows, such as channel configurations, typical ranges of Reynolds numbers associated with these regimes are summarized in Table 2.1.

Reynolds number range	Flow regime
$Re \lesssim 2 \times 10^3$	Laminar
$2 \times 10^3 \lesssim Re \lesssim 4 \times 10^3$	Transitional
$Re \gtrsim 4 \times 10^3$	Turbulent

Table 2.1: Typical flow regimes as a function of the Reynolds number for internal flows.

In the present work, the incompressible Navier-Stokes equations are employed to model the dynamics of viscous fluid flow in a two-dimensional channel, both in the absence and in the presence of solid obstacles. The channel walls are treated as rigid boundaries and are subject to no-slip boundary conditions, which impose a vanishing fluid velocity at the solid surfaces. Physically, the no-slip condition enforces the adherence of the fluid to the walls due to viscous effects.

In the absence of obstacles and for sufficiently low Reynolds numbers, the channel flow admits a well-known analytical solution, known as plane Poiseuille flow. In this regime, the flow is steady and fully developed, and the velocity profile across the channel exhibits a parabolic shape, with maximum velocity at the centerline and zero velocity at the walls.

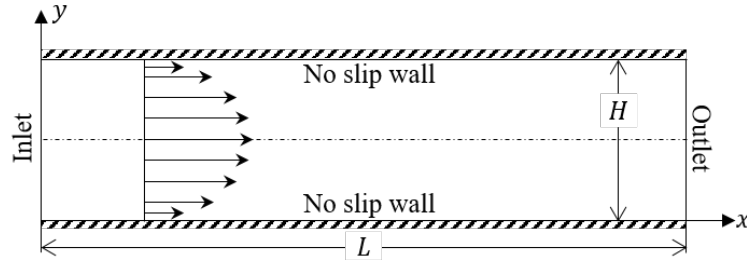


Figure 2.3: Schematic velocity profile of plane Poiseuille flow between two parallel plates.

Figure 2.3 illustrates the characteristic parabolic velocity profile associated with plane Poiseuille flow.

Except for a restricted class of highly symmetric and steady configurations, the incompressible Navier-Stokes equations do not admit explicit analytical solutions. This limitation is not solely due to the presence of nonlinear terms, but rather to the combined effect of nonlinear convective coupling, the incompressibility constraint, and the strong dependence of the solution structure on the geometry of the domain and boundary conditions [35, 36]. In particular, analytical solutions are only available in special cases where the nonlinear convective term vanishes or can be significantly simplified, as in plane Poiseuille flow. Outside these idealized configurations, the Navier-Stokes equations define a coupled nonlinear system of partial differential equations whose solutions cannot be expressed in closed form and must instead be approximated using numerical methods or studied through qualitative analytical techniques [37]. As a consequence, the investigation of channel flows involving finite domains, solid obstacles, and unsteady dynamics necessarily relies on numerical approximation techniques. These methods provide discrete representations of the continuous problem, enabling the computation of flow fields in configurations that are analytically intractable.

In the present work, such numerical solutions are obtained using the open-source computational fluid dynamics framework OpenFOAM, which is introduced in the following section.

2.2. OpenFOAM and the Numerical Solution of Navier-Stokes Equations

As discussed in the previous section, the incompressible Navier-Stokes equations generally do not admit explicit analytical solutions in realistic flow configurations. As a result, their study in bounded domains and complex geometries necessarily relies on numerical

approximation techniques.

In the present work, the numerical approximation of the incompressible Navier-Stokes equations is performed using the open-source computational fluid dynamics framework OpenFOAM [38]. OpenFOAM is employed as a high-fidelity numerical solver to generate reference solutions.

OpenFOAM is based on the finite-volume method (FVM) [39], in which the domain is discretized into control volumes and the integral form of the conservation laws is enforced locally. The finite-volume formulation is particularly well suited for fluid dynamics problems for several reasons. In fact, it guarantees strict local conservation of mass and momentum at the discrete level, since flux balances are imposed on each control volume. Moreover, its flux-based structure provides robust handling of convection-dominated flows through the explicit treatment of face fluxes and upwind-type schemes.

Fields such as velocity and pressure are represented as C++ objects. The coupling between velocity and pressure is typically handled through the SIMPLE, PISO, or PIMPLE algorithms, depending on whether the problem is steady or transient. These algorithms belong to the class of pressure-velocity coupling methods widely used in incompressible flow solvers [40]. SIMPLE is primarily designed for steady-state simulations and relies on iterative under-relaxation to achieve convergence. PISO is tailored for transient problems and performs multiple pressure corrections within each time step to improve temporal consistency. PIMPLE combines features of both approaches, enhancing robustness for unsteady simulations with larger time steps. In the present work, the transient incompressible solver icoFoam is adopted for laminar and transitional regimes; this solver is based on the PISO algorithm. For the highest Reynolds number case, the pimpleFoam solver is employed, which relies on the PIMPLE algorithm to ensure stable pressure-velocity coupling in the presence of turbulence modeling.

In the following section, we show how the numerical framework introduced above is employed to solve the flow configurations of interest in this work, namely the two-dimensional channel flow and the channel flow in the presence of solid obstacles. As discussed previously, these configurations constitute the reference problems used throughout the thesis.

The numerical solutions obtained with OpenFOAM play a crucial role in the present study, as they are used to generate high-fidelity datasets that serve as ground-truth for the data-driven and machine learning models investigated in the subsequent chapters. While OpenFOAM provides an accurate and reliable approach for solving the Navier-Stokes equations in complex geometries, its computational cost can become significant. For instance, in the relatively simple two-dimensional channel with obstacle configura-

tion considered here, a single transient simulation required a few hours of wall-clock time on a standard 4-core machine. In more complex geometries, three-dimensional settings, or higher-resolution simulations, the computational effort increases substantially. Consequently, parametric studies or repeated evaluations under varying boundary conditions can easily result in cumulative runtimes of several days [14].

2.3. Numerical setup of the channel flow in OpenFOAM

This work focuses on a set of baseline flow configurations, namely two-dimensional channel flows in the absence and in the presence of a solid obstacle. These canonical problems are chosen as fundamental reference cases.

Owing to their simplicity, interpretability and well-understood physical behavior, channel flows constitute a standard benchmark in computational fluid dynamics and provide a suitable foundation for the development and validation of data-driven and machine learning models.

In this section, we describe how these baseline configurations are solved numerically using OpenFOAM in order to generate the reference datasets employed throughout this work.

Let us start by considering a laminar flow in a two-dimensional channel. The inflow boundary prescribes a uniform velocity profile. Although this condition does not correspond to a fully developed channel profile, it provides a simple and controlled way to set the volumetric flow rate in the domain. As a consequence, a short entrance region is required to allow the flow to adjust and develop spatial velocity gradients before reaching the region of interest. The outflow boundary is treated as a free-exit condition, enabling the fluid to leave the domain without imposing artificial constraints on the pressure distribution.

The computational domain is schematically illustrated in Figure 2.4.

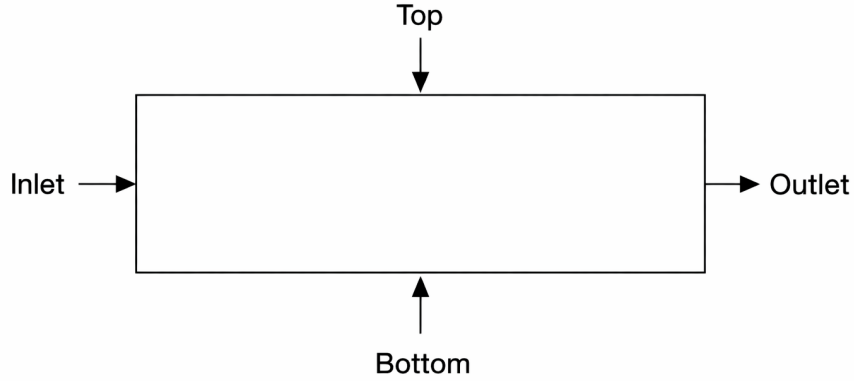


Figure 2.4: The configuration of the channel flow

In particular, the physical problem under consideration is two-dimensional. However, OpenFOAM operates on three-dimensional meshes and does not natively support purely two-dimensional discretizations. For this reason, the two-dimensional domain is extruded into a thin three-dimensional domain with a small thickness in the third spatial direction. On the front and back faces, the empty boundary condition is imposed, which enforces invariance of the solution along the thickness direction and removes any flux contribution across these faces. As a result, the computed flow field corresponds to a two-dimensional solution, while retaining the numerical structure of a three-dimensional finite-volume discretization.

The generation of the computational mesh plays a crucial role in both the accuracy of the numerical simulations and the quality of the datasets employed to train the graph-based surrogate models. In this work, different meshing strategies are adopted depending on the geometric complexity of the configuration under consideration.

For the simplest configurations, namely channel flows without obstacles or with a square obstacle aligned with the Cartesian axes, the mesh is generated using the `blockMesh` utility of OpenFOAM. This tool allows the construction of structured, hexahedral meshes by explicitly defining a set of vertices, blocks and grading parameters in the `blockMeshDict` file. Owing to the simplicity of the underlying geometry, this approach provides full control over the mesh resolution and stretching, enabling the refinement of regions of interest such as boundary layers near the walls or around sharp obstacle corners.

The resulting meshes are structured in the streamwise and wall-normal directions, which ensures a regular connectivity pattern and facilitates the interpretation of the numerical results for these baseline cases.

For configurations involving non-Cartesian geometries, such as circular obstacles or airfoil profiles, a fully structured mesh generated via blockMesh is no longer suitable. In these cases, the mesh is generated using the open-source mesh generator Gmsh [41]. This tool allows the construction of unstructured meshes by directly discretizing the computational domain based on its geometric description, enabling an accurate representation of curved boundaries. Using Gmsh, the geometry is defined through a set of points, curves and surfaces, and the mesh is generated by specifying local characteristic lengths that control the element size. This approach provides flexibility in refining the mesh near solid boundaries and regions of interest, while maintaining a coarser resolution in the far field. The resulting meshes are unstructured and originate from a two-dimensional triangulation of the domain. After extrusion in the third spatial direction, the final three-dimensional mesh consists of prismatic cells with a small thickness, consistent with the quasi two-dimensional formulation adopted in this work.

The use of both blockMesh and Gmsh allows us to systematically vary the geometric complexity of the flow configurations, ranging from simple channel flows to cases involving curved obstacles and airfoil-like shapes. Although unstructured meshes could in principle be employed for all configurations, structured meshes generated with blockMesh offer advantages in simple Cartesian geometries. In particular, they ensure orthogonality and alignment with the main flow directions, which reduces discretization errors and improves numerical efficiency. For this reason, structured meshes are adopted for simple channel configurations, while unstructured meshes generated with Gmsh are reserved for geometries involving curved boundaries.

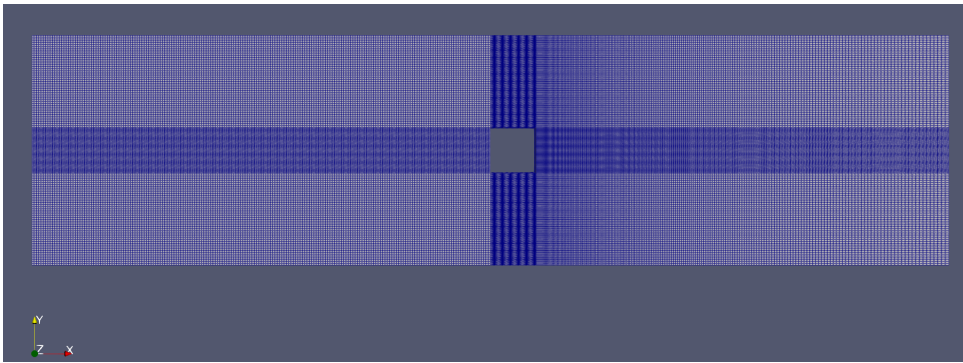


Figure 2.5: Structured mesh generated using blockMesh for a channel flow configuration with Cartesian geometry.

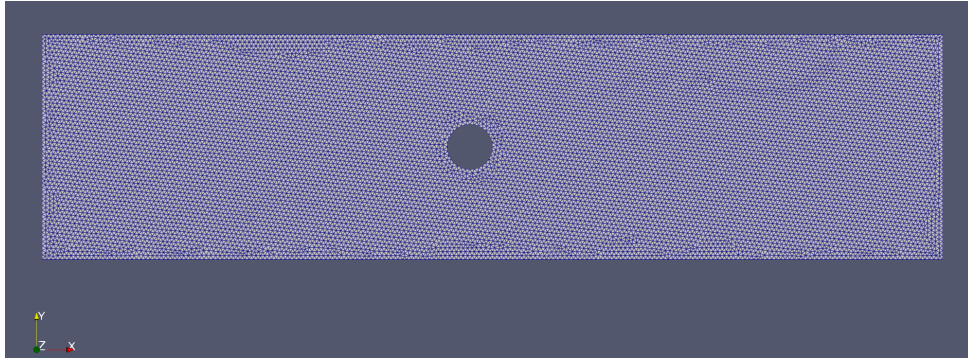


Figure 2.6: Unstructured mesh generated using Gmsh for a channel flow configuration with a curved obstacle.

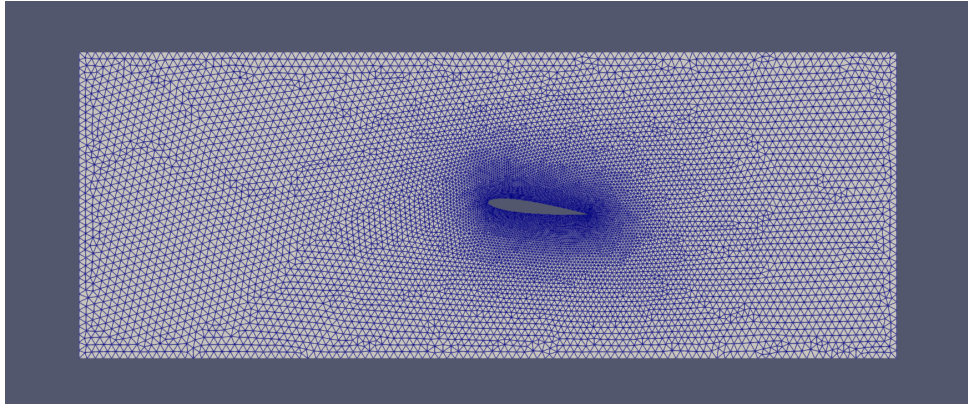


Figure 2.7: Unstructured mesh generated using Gmsh for a channel flow configuration with an airfoil obstacle.

Examples of structured and unstructured meshes generated using blockMesh and Gmsh, respectively, are shown in Figures 2.5, 2.6 and 2.7.

For the simplest test cases, where the flow remains laminar and no obstacles are present, we employed a standard set of boundary conditions commonly used for canonical external-flow configurations. These conditions are summarised in Table 2.2, and are sufficient for problems where no significant flow separation or recirculation occurs.

Boundary	Velocity $\mathbf{u}$	Pressure p
Inlet	fixedValue $\mathbf{u} = (1, 0)$	zeroGradient
Outlet	zeroGradient	fixedValue 0
Top/Bottom	noSlip $\mathbf{u} = (0, 0)$	zeroGradient
Front/Back	empty	empty

Table 2.2: Boundary conditions used for the simplest test cases without obstacles.

At the inlet, a prescribed velocity profile is imposed,

$$\mathbf{u} = (1, 0).$$

At the outlet, a homogeneous Neumann condition is applied to the velocity field

$$\frac{\partial \mathbf{u}}{\partial n} = \mathbf{0}$$

where $\partial/\partial n$ denotes the derivative in the direction normal to the boundary. For a vector field, this condition enforces that the normal derivative of each velocity component vanishes.

On the top and bottom walls, the no-slip condition is enforced:

$$\mathbf{u} = (0, 0).$$

At the outlet, a fixed reference value for the pressure is prescribed,

$$p = 0$$

which removes the additive indeterminacy (gauge freedom) of the pressure field and ensures uniqueness of the solution.

On the inlet and wall boundaries, homogeneous Neumann conditions are imposed,

$$\frac{\partial p}{\partial n} = 0,$$

allowing the pressure field to be determined consistently by the interior flow solution.

In the OpenFOAM implementation, homogeneous Neumann conditions correspond to the zeroGradient boundary type, while Dirichlet conditions are imposed using fixedValue.

However, for more complex configurations, particularly those involving obstacles, vortex shedding and non-trivial wake dynamics, the previous setup may lead to numerical instabilities or physically inconsistent behaviour at the boundaries. In these cases, we adopted a more robust configuration based on mixed outlet conditions (`inletOutlet` for the velocity and `zeroGradient` for the pressure) and a flux-corrected pressure condition (`fixedFluxPressure`) at the inlet. Although these boundary conditions could also be applied to the simplest laminar channel-flow configuration, their use in that setting would be unnecessary. In the absence of flow separation or backflow, the simpler Dirichlet–Neumann combination described previously is sufficient to guarantee well-posedness and numerical stability.

The `inletOutlet` condition switches between a zero-gradient and a fixed-value treatment for the velocity depending on the local flow direction, allowing fluid to exit the domain while correctly handling possible backflow at the outlet. At the inlet, the velocity field is prescribed.

In unsteady incompressible flow, however, the pressure cannot be fixed independently, since it acts as a Lagrange multiplier enforcing the continuity constraint. For this reason, the normal pressure gradient at the inlet is not imposed a priori. Instead, it is dynamically adjusted so that it remains consistent with the prescribed velocity field and with the global mass conservation of the domain. This prevents inconsistencies between the imposed velocity flux and the pressure correction step, which could otherwise generate spurious pressure oscillations, especially in the presence of strong vortical structures. In the OpenFOAM implementation, this boundary treatment corresponds to the `fixedFluxPressure` condition.

These boundary conditions, listed in Table 2.3, allow for possible backflow at the outlet and ensure a better control of the mass flux through the domain, which is essential when dealing with vortical structures. This is particularly important in the presence of unsteady wakes, where flow reversal may occur at the outlet and a purely Neumann or Dirichlet treatment could lead to numerical instabilities or non-physical pressure reflections affecting the upstream solution.

Boundary	Velocity $\mathbf{u}$	Pressure p
Inlet	fixedValue $\mathbf{u} = (1, 0)$	fixedFluxPressure
Outlet	inletOutlet	zeroGradient
Walls/Airfoil	noSlip $\mathbf{u} = (0, 0)$	zeroGradient
Front/Back	empty	empty

Table 2.3: Boundary conditions used for the cases involving obstacles and unsteady wake dynamics.

With the boundary conditions described above, the simulations exhibit stable and consistent numerical behaviour across all configurations considered in this study. At each time step, the velocity residuals typically decrease to the order of 10^{-8} , while the pressure correction converges below 10^{-6} . The continuity errors remain negligible, confirming that the incompressibility constraint is effectively enforced. Moreover, the Courant number remains well below unity, ensuring stable temporal integration of the unsteady problem. Taken together, these indicators confirm that the adopted spatial and temporal discretizations provide reliable and internally consistent reference solutions, which can be used as a trustworthy ground truth for the training and evaluation of the surrogate models. While the OpenFOAM simulations described in this chapter yield accurate and physically consistent results, they are not computationally negligible. The unstructured two-dimensional meshes employed in this work range from approximately 10^3 to 10^4 control volumes, corresponding to several thousand degrees of freedom for the incompressible velocity pressure system. Capturing unsteady wake dynamics requires several thousand time steps, resulting in wall-clock times on the order of hours on a standard four-core machine. Although such computational costs are manageable for isolated simulations, they become increasingly significant when multiple parameter configurations, mesh variations, or repeated evaluations are required. This scalability issue provides the primary motivation for exploring data-driven surrogate approaches capable of approximating flow dynamics at a reduced computational cost.

3 | Neural Models for PDE-Based Simulations

In the Chapter 2, the mathematical formulation of the reference problem and the numerical framework used to generate high-fidelity CFD solutions were introduced. While these solvers provide accurate and physically consistent results, their computational cost and their intrinsically case dependent nature, since any modification of the problem setup entails solving the full system again, make them impractical in scenarios requiring fast predictions or repeated evaluations.

To address these limitations, this work explores a machine learning-based surrogate modeling approach aimed at capturing the essential flow dynamics. In particular, a physics-informed MeshGraphNet framework is considered. This chapter introduces the theoretical foundations of the modeling framework used in this work. We begin by reviewing standard neural network models (Section 3.1), highlighting both their expressive power and their limitations in the context of PDE-based simulations. We then introduce graph-based architectures (Section 3.2), which naturally extend neural networks to irregular and unstructured domains and form the basis of the MeshGraphNet model adopted in this work. Finally, physics-informed loss formulations (Section 3.3) and meshless discretization techniques based on radial basis functions (Section 3.4) are presented as key components for embedding physical constraints into the learning process.

Together, these elements define the neural modeling framework developed in this thesis and used in the subsequent chapters.

3.1. Neural Networks

Neural networks are a class of data-driven models designed to approximate nonlinear mappings between input and output variables. Formally, a neural network represents a parametric function composed of a sequence of linear transformations and nonlinear activation functions, whose parameters are learned from data through optimization pro-

cedures based on gradient descent [42]. A schematic example of a standard feed-forward neural network architecture is shown in Fig. 3.1. In a feed-forward network, information propagates unidirectionally from the input layer to the output layer through a sequence of hidden layers, without feedback loops or recurrent connections.

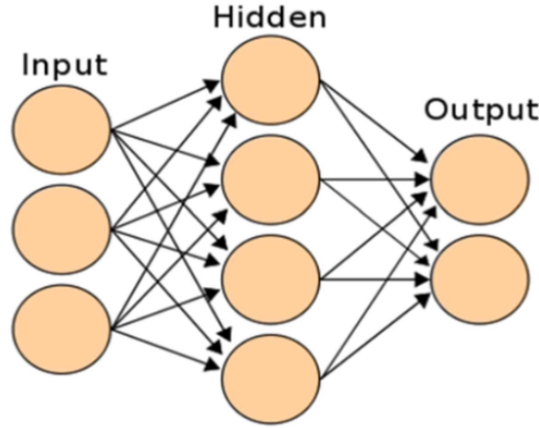


Figure 3.1: Illustration of a typical feed-forward neural network architecture, in which artificial neurons (units) are arranged in successive layers and connected to adjacent layers.

Due to their expressive power, neural networks have been successfully applied to a wide range of tasks, including regression, classification and representation learning [43]. In particular, feed-forward architectures and convolutional neural networks (CNNs) have become standard tools in many scientific and engineering applications, benefiting from efficient implementations on modern hardware [44, 45].

CNNs are designed to operate on data defined over regular grids, where local neighbourhoods are uniformly structured. Convolutional filters are spatially shared, meaning that the same kernel is applied at every location in the domain. This induces translation invariance, i.e., spatial patterns are processed independently of their absolute position. While this property is advantageous for image-like data, it implicitly assumes that the underlying domain is discretized on a regular Cartesian grid, where distances between neighbouring points are uniform and local connectivity remains consistent throughout the domain. In computational fluid dynamics, however, solutions are typically defined on irregular or unstructured meshes, where node spacing, connectivity, and local topology vary significantly. Projecting such data onto a regular grid in order to apply standard convolutions may introduce interpolation errors and artificial pixelation effects, potentially degrading the representation of fine geometric features or boundary-layer dynamics. Furthermore, the assumption of uniform local structure is incompatible with adaptive mesh refinement and variable connectivity. Consequently, the direct use of conventional CNN

architectures in problems involving complex geometries or unstructured discretizations is non-trivial [28].

These limitations motivate the development of neural architectures capable of operating on more general data representations. In the following section, we introduce graph-based neural models, which extend the neural network paradigm to relational and non-Euclidean domains by explicitly encoding interactions between connected entities.

3.2. MeshGraphNet

Mesh-based simulations play a fundamental role in the modeling of complex physical systems across many areas of science and engineering. By discretizing the computational domain into a collection of interconnected elements, a mesh representation provides a flexible geometric framework on which numerical methods can be formulated. The resolution can be locally adapted, allowing a balance between accuracy and computational cost.

Despite their central role in numerical simulation, mesh-based representations have received comparatively limited attention in machine learning. Most data-driven models for physical systems continue to rely on regular grids, largely due to the computational efficiency and widespread adoption of convolutional neural networks [45]. Early efforts toward learning on meshes include geometry processing techniques [28, 46] and hybrid approaches combining numerical PDE solvers with neural networks [47], yet general-purpose learning frameworks for unstructured meshes remain relatively scarce.

MeshGraphNets, introduced by Pfaff et al. [29], address these limitations by representing the simulation domain as a graph, where mesh nodes are treated as graph nodes and connectivity relations are encoded as edges. Interactions between neighbouring nodes are modeled through message-passing operations, allowing information to propagate along the graph structure. Unlike grid-based convolutional approaches, no projection onto a regular lattice is required. Avoiding this intermediate resampling step prevents the introduction of interpolation artefacts and artificial pixelation effects, which may otherwise blur fine-scale flow structures or distort boundary-layer behaviour. This formulation enables learning directly on irregular and unstructured meshes, allowing models to generalize across different resolutions and geometries (see Fig. 3.2).

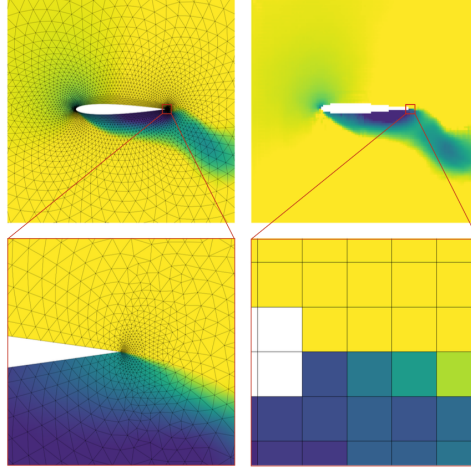


Figure 3.2: Comparison between mesh-based predictions obtained using a graph-based representation (left) and a grid-based convolutional baseline (right). The graph-based approach enables a more localized and adaptive resolution of the flow field, allowing finer-scale structures to be captured more accurately [29].

In this work, the MeshGraphNets architecture is adopted to model two-dimensional, incompressible fluid flows in canonical configurations. The graph-based formulation captures mesh connectivity and geometry, while physics-informed constraints are integrated into the training process to promote physical consistency.

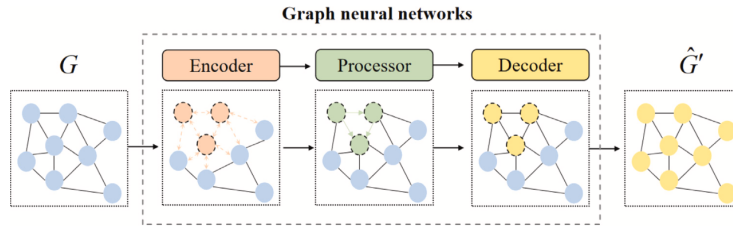


Figure 3.3: Diagram of Graph Neural Network architecture [48].

As shown in Fig. 3.3, the MeshGraphNet architecture follows a modular encoder-processor-decoder structure. Raw node and edge features are first mapped into a latent space through dedicated encoder networks. A sequence of message-passing layers then propagates information across the graph, enabling each node to aggregate contributions from its local neighborhood and to capture both local and longer-range interactions. Finally, a decoder maps the processed node embeddings back to the physical output space, producing node-wise predictions of the target quantities.

In the following subsections, each component of the architecture is described in detail,

with particular emphasis on the encoder design, the message-passing mechanism and the decoder used in this work.

3.2.1. Node and Edge Encoders

Each mesh vertex is represented as a node and is associated with a feature vector combining spatial, topological and dynamic information. Spatial features include the node position within the computational domain and node-type indicators used to distinguish interior nodes from boundary and obstacle nodes. Dynamic features encode the local flow state and its temporal evolution, such as the velocity components at previous time steps and the time increment between successive snapshots.

Edges connect neighboring nodes and store attributes derived from local geometric information. In the present implementation, for each edge connecting nodes i and j , the associated feature vector is defined as

$$\mathbf{e}_{ij} = (x_i - x_j, y_i - y_j, \|\mathbf{x}_i - \mathbf{x}_j\|),$$

where $\mathbf{x}_i = (x_i, y_i)$ and $\mathbf{x}_j = (x_j, y_j)$ denote the spatial coordinates of the two nodes.

To transform these heterogeneous inputs into a uniform latent representation, two encoder multilayer perceptrons are employed:

$$\epsilon_{\text{node}} : \mathbb{R}^{F_n} \rightarrow \mathbb{R}^d, \quad \epsilon_{\text{edge}} : \mathbb{R}^{F_e} \rightarrow \mathbb{R}^d.$$

Here, F_n and F_e denote the dimensionality of the raw node and edge feature vectors, respectively, while d represents the dimension of the latent embedding space shared by both nodes and edges. In other words, the encoder maps the original feature spaces $\mathbb{R}^{F_n}$ and $\mathbb{R}^{F_e}$ into a common vector space $\mathbb{R}^d$, where subsequent message-passing operations are performed. This space is referred to as latent because its coordinates do not directly correspond to physical quantities, but rather to learned abstract features optimized during training. A multi layer perceptron is a feed-forward neural network composed of a sequence of linear transformations followed by nonlinear activation functions, which allows for learning nonlinear mappings between input and output spaces. In the present work, the encoder functions ϵ_{node} and ϵ_{edge} consist of stacked linear layers combined with nonlinear activation functions (ReLU) [49] and Layer Normalization [50]. The Rectified Linear Unit (ReLU) is defined element-wise as

$$\text{ReLU}(x) = \max(0, x),$$

which sets negative inputs to zero while leaving positive values unchanged. The introduction of this nonlinearity prevents the network from collapsing into a purely linear mapping and mitigates vanishing-gradient effects typically associated with saturating activation functions.

This design, showed in Figure 3.4, enables the projection of input features with different physical meanings and scales into a common latent space, while improving training stability and convergence.

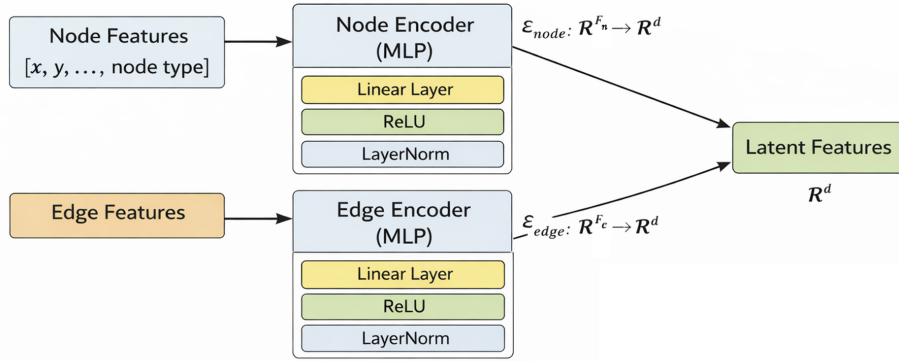


Figure 3.4: Schematic representation of the encoder architecture

3.2.2. Processor: Message Passing Layers

The processor represents the core of the MeshGraphNets architecture and is responsible for modeling the local interactions between nodes linked by edges in the underlying mesh graph. Starting from the latent node and edge embeddings produced by the encoder, the processor iteratively propagates information across the graph through a sequence of message-passing layers.

At a given message-passing layer, each node i is associated with a latent embedding h_i , which encodes its features, while each edge (i, j) is associated with a latent embedding e_{ij} . In this formulation, edge embeddings play the role of messages exchanged between neighboring nodes and mediate the flow of information across the mesh.

Each message-passing layer is composed of three successive steps: message computation, aggregation and node update.

Message computation. In the first step, messages are computed along each edge by combining the embeddings of the two incident nodes with the current edge embedding. This operation captures pairwise interactions between neighboring nodes and updates the

edge representation according to

$$e'_{ij} = e_{ij} + \phi_{\text{edge}}([h_i, h_j, e_{ij}]), \quad (3.1)$$

where ϕ_{edge} is a multilayer perceptron acting on the concatenation of the node and edge embeddings.

The term e_{ij} added to the nonlinear transformation defines a residual connection, i.e., an identity mapping that adds the input embedding to its learned update. Formally, a residual connection implements an update of the form

$$y = x + F(x),$$

instead of a purely transformed mapping $y = F(x)$. This mechanism facilitates gradient propagation across multiple stacked message-passing layers and stabilizes the optimization process. Moreover, it allows the edge embedding to be updated incrementally, preserving geometric information accumulated in previous layers while learning corrective interaction terms. After this step, the updated edge embeddings e'_{ij} represent the messages propagated from node j to node i .

Aggregation. In the second step, each node aggregates the messages received from its neighboring edges. Since graph neighborhoods have no intrinsic ordering and may differ in size from node to node, the aggregation operation must be permutation invariant. In this work, message aggregation is performed through summation:

$$\tilde{h}_i = \sum_{j \in \mathcal{N}(i)} e'_{ij}, \quad (3.2)$$

where $\mathcal{N}(i)$ denotes the set of nodes connected to node i . The aggregated message $\tilde{h}_i$ summarizes the information exchanged between node i and its local neighborhood.

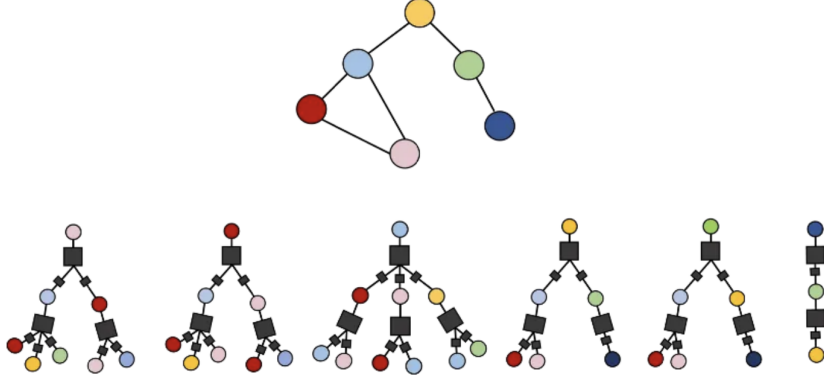


Figure 3.5: Example of message passing and aggregation in a graph neural network [29].

Figure 3.5 depicts a toy graph composed of six nodes connected by six edges and illustrates the effect of two consecutive message-passing layers. The top panel shows the original graph connectivity, while the lower diagrams visualize how the representation of each node is updated through successive local aggregations. After the first message-passing layer, each node embedding incorporates information from nodes that are directly connected to it in the graph. After the second layer, the updated embedding depends not only on its immediate neighbors, but also on nodes that are connected through an intermediate vertex. Therefore, with two stacked message-passing layers, information propagates across paths of length two in the graph, progressively enlarging the region of influence for each node representation. This example is characterized by topology-constrained, local information exchange. Node representations are updated exclusively through edges defined in the graph connectivity, and long-range dependencies emerge only through successive layers. This distinguishes message-passing networks from fully connected architectures or global attention mechanisms, where interactions between distant nodes are computed directly within a single layer. Here, information propagation is strictly structured by the graph topology and expands progressively with network depth.

Node update. In the final step, the aggregated neighborhood information is combined with the current node embedding to produce an updated node representation:

$$h'_i = h_i + \phi_{\text{node}}([h_i, \tilde{h}_i]), \quad (3.3)$$

where ϕ_{node} is a multilayer perceptron implementing the node update function. The residual connection preserves information from the previous embedding while allowing the network to incorporate the influence of neighboring nodes.

Together, message computation, aggregation and node update define a single message-

passing layer. By stacking l such layers, information is progressively propagated across larger regions of the mesh, enabling each node to capture both local interactions and longer-range spatial dependencies while sharing the same learned parameters across all nodes and edges. The number of stacked message-passing layers l determines the maximum propagation distance of information across the graph. Each layer enables a node to incorporate information from directly connected neighbors, so that deeper processors allow progressively wider regions of the mesh to influence each node representation. In practice, the choice of l depends on the spatial scale of the physical interactions to be modeled and on the topological extent of the mesh. A small number of layers restricts the model to highly local interactions, whereas increasing l enlarges the effective propagation range. However, excessively deep message-passing stacks may lead to representation degradation and higher computational cost. The selected depth therefore reflects a balance between propagation capability and numerical stability.

3.2.3. Decoder

After the message-passing stage, each node embedding contains a rich latent representation that encodes both local and non-local information about the flow dynamics. Here, local information refers to features originating from nodes directly connected in the mesh topology, whose influence is incorporated through a single message-passing step. Non-local information, instead, arises from the successive stacking of multiple message-passing layers, which allows information to propagate across longer paths in the graph. As a result, the final node embedding reflects not only the immediate neighborhood of the node, but also contributions from more distant regions of the mesh reached through iterative local updates.

The decoder maps these final node embeddings back to the physical output space through a multilayer perceptron

$$\text{MLP}_{\text{dec}} : \mathbb{R}^d \rightarrow \mathbb{R}^{F_{\text{out}}}.$$

In the present work, F_{out} corresponds to the number of predicted physical quantities at each node and includes the two velocity components and the pressure. Accordingly, the decoder predicts the variation of the velocity components and the pressure at each node. The decoder consists of two fully connected layers with ReLU activation [49] and is applied independently to each node embedding, producing the final node-wise predictions used for training and evaluation.

3.3. Physics-Informed Neural Networks

Purely data-driven neural network models, while effective in reproducing observed flow fields, often suffer from limited generalization and long-term instability when applied to complex physical systems, meaning that their predictive accuracy may degrade when confronted with unseen geometries, boundary conditions, or flow regimes, and that small prediction errors can accumulate over time in autoregressive settings, leading to physically inconsistent or unstable dynamics. In particular, the absence of explicit physical constraints may lead to predictions that violate fundamental conservation laws or deteriorate rapidly when applied to configurations outside the distribution of the training data [15, 30, 51].

To mitigate these issues, several approaches have been proposed to incorporate prior physical knowledge directly into the training process. These include physics-aware or physics-guided models, which integrate physical principles into the architecture or training objective through inductive biases, conservation-based regularization terms, or hybrid data-model formulations [52, 53]. Other strategies combine neural networks with classical numerical solvers in hybrid or operator-learning frameworks [18]. Among these approaches, Physics-Informed Neural Networks (PINNs) represent a prominent class of neural models in which the governing equations of the system are explicitly embedded into the loss function by minimizing the residual of the corresponding partial differential equations [21, 22].

In physics-informed learning approaches, neural networks are trained not only to match observed data, but also to satisfy the governing physical laws of the system. Rather than relying exclusively on supervised loss terms, physical constraints are incorporated into the training objective as additional regularization terms, promoting physically consistent predictions. In the context of incompressible fluid dynamics, this corresponds to enforcing the Navier–Stokes equations during training through additional physics-based loss terms. Specifically, the loss function is augmented with penalty terms associated with the residual of the momentum equations, the incompressibility constraint, and the prescribed initial and boundary conditions.

In the present work, the neural network outputs approximations of the velocity field $\mathbf{u}_\theta$ and the pressure field p_θ , where θ denotes the collection of trainable parameters of the network. Physical consistency is enforced by augmenting the training loss with additional terms that measure the residual of the governing equations and the mismatch with prescribed boundary conditions. Specifically, violations are penalized through weighted norm-based residual terms included in the objective function, so that deviations from

the Navier–Stokes equations and boundary constraints increase the overall loss during optimization.

The total loss function is defined as

$$\mathcal{L}(\boldsymbol{\theta}) = \lambda_{\text{PDE}} \mathcal{L}_{\text{PDE}} + \lambda_{\text{BC}} \mathcal{L}_{\text{BC}} + \lambda_{\text{IC}} \mathcal{L}_{\text{IC}} + \lambda_{\text{data}} \mathcal{L}_{\text{data}}, \quad (3.4)$$

where $\mathcal{L}_{\text{PDE}}$ penalizes the residual of the incompressible Navier–Stokes equations in the interior of the domain, while $\mathcal{L}_{\text{BC}}$ and $\mathcal{L}_{\text{IC}}$ penalize deviations from the prescribed boundary and initial conditions, respectively. It is important to note the presence of the supervised term $\mathcal{L}_{\text{data}}$ in the loss formulation. In principle, the final objective would be to drive its corresponding weight λ_{data} to zero, so that the model operates in a fully unsupervised, physics-driven regime. However, this condition was not always consistently attainable in practice. The persistence of a non-zero supervised weight indicates that, beyond the enforcement of boundary and initial conditions, additional supervision over selected regions of the domain is necessary to stabilize training and guide the solution toward physically meaningful states. In particular, supervision was required over spatial regions broader than the sole boundary constraints, highlighting the practical limitations of relying exclusively on physics-based residual terms.

Specifically, the PDE loss is evaluated at a set of interior nodes N_r as

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_r} \sum_{i=1}^{N_r} (\|\mathcal{R}_{\text{mom}}(x_i)\|^2 + \|\mathcal{R}_{\text{cont}}(x_i)\|^2), \quad (3.5)$$

where $\mathcal{R}_{\text{mom}}$ and $\mathcal{R}_{\text{cont}}$ denote the residuals of the momentum and continuity equations, respectively.

Starting from the incompressible Navier–Stokes equations (2.1), the physics-based residuals are obtained by substituting the neural network approximations $\mathbf{u}_{\boldsymbol{\theta}}$ and $p_{\boldsymbol{\theta}}$ into the governing equations. In particular, the momentum residual is defined as

$$\mathcal{R}_{\text{mom}} = \frac{\partial \mathbf{u}_{\boldsymbol{\theta}}}{\partial t} + (\mathbf{u}_{\boldsymbol{\theta}} \cdot \nabla) \mathbf{u}_{\boldsymbol{\theta}} + \nabla p_{\boldsymbol{\theta}} - \nu \nabla^2 \mathbf{u}_{\boldsymbol{\theta}}, \quad (3.6)$$

while the continuity residual is given by

$$\mathcal{R}_{\text{cont}} = \nabla \cdot \mathbf{u}_{\boldsymbol{\theta}}. \quad (3.7)$$

Boundary conditions are enforced through the loss term

$$\mathcal{L}_{\text{BC}} = \frac{1}{N_b} \sum_{i=1}^{N_b} (\|\mathbf{u}_\theta(x_i) - \mathbf{u}(x_i)\|^2 + \|p_\theta(x_i) - p(x_i)\|^2), \quad (3.8)$$

where N_b denotes the number of boundary nodes, located at the inlet, outlet, and wall regions of the domain.

In classical PINNs, the derivatives appearing in the PDE residual are computed via automatic differentiation (AD) [21]. More precisely, spatial and temporal derivatives of the neural network outputs are obtained through AD and then substituted into the governing equations to construct the residual terms. Automatic differentiation provides exact neural-network derivatives but may become computationally demanding for stiff operators or when high-order derivatives are required [25, 54]. These issues are well documented in recent work, which highlights the slow convergence and limited accuracy of classical PINNs, especially for time-dependent and high-dimensional PDEs [48].

3.3.1. Discrete Physics-Informed Losses

An alternative to automatic differentiation consists in evaluating the PDE residuals through a numerical discretization of the spatial differential operators. In this approach, the neural network still predicts the physical fields, but derivatives such as $\frac{\partial \mathbf{u}}{\partial x}$, $\frac{\partial \mathbf{u}}{\partial y}$, and $\nabla^2 \mathbf{u}$ are computed numerically rather than through backpropagation.

The computation of spatial derivatives is a critical aspect in physics-informed learning on graphs. Standard finite-difference (FD) schemes require a structured grid and regular connectivity, and are therefore not applicable to irregular domains or unstructured node layouts [55]. Automatic differentiation (AD), while flexible, may lead to high computational costs and numerical instabilities when computing higher-order derivatives, especially in graph-based architectures. The computational burden arises because each required derivative involves additional backpropagation steps through the entire computational graph of the network, effectively increasing both memory consumption and time complexity. In graph-based architectures, this issue is further amplified by the message-passing structure, where each node representation depends on multiple neighboring nodes across several layers. In contrast, meshfree methods based on radial basis functions (RBFs) or spline-based bases provide a solution to the problem of computing spatial derivatives on irregular domains [56]. Among these, the RBF-FD method has emerged as a particularly effective approach. By relying exclusively on local node coordinates and stencil-based approximations, RBF-FD enables the evaluation of differential operators directly on unstructured

point clouds, without requiring a structured mesh.

In the present work, spatial derivatives appearing in the momentum and continuity residuals are evaluated using RBF-FD differentiation weights, which are introduced in Section 3.4. For completeness, in the special case of structured grids, standard finite-difference approximations could be employed instead. However, this alternative is mentioned only for reference. RBF-FD operators are adopted throughout the present framework to provide a unified discretization strategy that remains applicable to both structured and unstructured meshes.

With this formulation, the physics-informed loss is constructed by replacing the continuous differential operators in the residuals $\mathcal{R}_{\text{mom}}$ and $\mathcal{R}_{\text{cont}}$ with their RBF-FD discrete counterparts. Accordingly, the PDE loss term is defined as

$$\mathcal{L}_{\text{PDE}} = \frac{1}{N_r} \sum_{i=1}^{N_r} \left(\|\mathcal{R}_{\text{mom}}^{\text{RBF-FD}}(x_i)\|^2 + \|\mathcal{R}_{\text{cont}}^{\text{RBF-FD}}(x_i)\|^2 \right), \quad (3.9)$$

where the superscript RBF – FD denotes that all spatial derivatives are computed through RBF-FD discretization.

Our approach retains the structure and philosophy of physics-informed neural networks, while replacing automatic differentiation with a discretization strategy that is consistent with the simulation mesh. When combined with a graph-based representation of the computational domain, this formulation naturally accommodates irregular and unstructured geometries, while respecting the spatial locality and connectivity inherent to the mesh. Spatial differential operators are evaluated through RBF-FD discretizations, enabling the enforcement of physical constraints without incurring the computational overhead typically associated with high-order automatic differentiation. At the same time, the resulting loss formulation remains fully aligned with the numerical discretization adopted for the governing partial differential equations.

In this sense, the proposed model can be interpreted as a discrete physics-informed neural network operating directly on mesh-based representations. Together, the graph neural network architecture and the RBF-FD discretization constitute the foundation of the numerical–neural framework developed in this thesis, enabling the approximation of PDE-driven dynamics on both structured and unstructured meshes while embedding physical constraints directly into the learning process.

3.4. Radial Basis Functions and RBF-FD Discretization

Radial Basis Functions (RBFs) provide a flexible meshless framework for interpolation and numerical differentiation on scattered nodes. Their ability to operate independently of a structured grid makes them particularly suitable for partial differential equations defined on irregular or unstructured domains.

In recent years, physics-informed neural networks have been combined with graph neural networks to enforce physical constraints directly on graph-structured data. Several approaches have been proposed to approximate spatial derivatives on graphs, including message passing based discretizations [57], graph Laplacian operators [58], and weak-form or variational formulations [29]. More recently, RBF-FD operators have been explicitly integrated into PINN–GNN frameworks to provide accurate and local approximations of differential operators on irregular meshes [48].

The computation of spatial derivatives on graphs is non-trivial due to the absence of a canonical coordinate system and structured connectivity. RBF-FD methods construct explicit local approximations of differential operators using only geometric information and function values on neighboring nodes. This allows for accurate, mesh-independent discretizations while preserving locality and sparsity, which are essential for scalability in graph-based learning models. In this context, locality refers to the fact that the approximation of a differential operator at a given node depends only on function values within a finite stencil of neighboring nodes, rather than on the entire domain. Each derivative is therefore computed through a local weighted combination of nearby nodal values. Sparsity follows directly from this construction: since each node interacts only with a limited number of stencil points, the resulting differentiation matrices contain a small number of non-zero entries per row. This yields sparse operators whose storage and computational cost scale linearly with the number of nodes, making them well suited for large-scale graph-based models.

For these reasons, RBF-FD is adopted in this work as a principled and flexible approach to compute spatial derivatives within the physics-informed loss.

Consider a set of scattered nodes $\{\mathbf{x}_i\}_{i=1}^n \subset \Omega \subset \mathbb{R}^2$, where Ω denotes the computational domain of the flow problem. Let $u_x : \Omega \rightarrow \mathbb{R}$ be a sufficiently smooth scalar field defined on Ω . In the present context, u_x denotes the x -component of the velocity field. The same construction applies analogously to the y -component u_y and to the pressure field p , which

are treated as scalar fields defined over the same domain.

The values of u_x are assumed to be known at the nodal locations $\mathbf{x}_i$. We focus on a target node $\mathbf{x}_i$ and construct a local stencil

$$\mathcal{N}(i) = \{\mathbf{x}_i^k\}_{k=1}^m,$$

consisting of the m nearest neighboring nodes of $\mathbf{x}_i$.

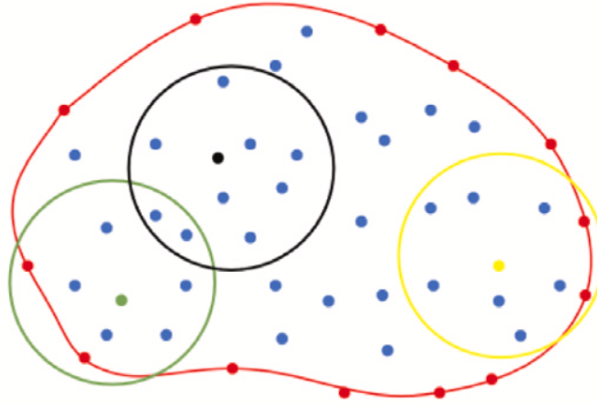


Figure 3.6: Example of stencil construction for RBF-FD discretization [48].

On this local stencil, the function $u_x(\mathbf{x})$ is approximated by an RBF interpolant augmented with low-degree polynomials:

$$u_x(\mathbf{x}_i) \approx \sum_{k=1}^m \lambda_k \phi(\mathbf{x}_i, \mathbf{x}_i^k) + \sum_{k=1}^q \mu_k p_k(\mathbf{x}_i), \quad (3.10)$$

where $\phi(r)$ is a radial basis function and $\{p_k\}_{k=1}^q$ denotes a polynomial basis of total degree up to a prescribed order. Here, q is the number of polynomial basis functions included in the augmentation. The polynomial terms are introduced to ensure stability and consistency of the approximation. To this end, the interpolation coefficients λ_k are required to satisfy the polynomial reproduction constraints

$$\sum_{k=1}^n \lambda_k p_j(\mathbf{x}_k) = 0, \quad j = 1, \dots, q, \quad (3.11)$$

Under these conditions, the interpolation problem reduces to the following linear system

$$\begin{bmatrix} A & P \\ P^\top & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\mu} \end{bmatrix} = \begin{bmatrix} \mathbf{u}_x \\ \mathbf{0} \end{bmatrix}, \quad (3.12)$$

where $A \in \mathbb{R}^{m \times m}$ is the local RBF interpolation matrix with entries $A_{jk} = \phi(\mathbf{x}_i^j, \mathbf{x}_i^k)$, $P \in \mathbb{R}^{m \times q}$ is the polynomial evaluation matrix with entries $P_{jk} = p_k(\mathbf{x}_i^j)$, $\boldsymbol{\lambda} \in \mathbb{R}^m$ and $\boldsymbol{\mu} \in \mathbb{R}^q$ are the interpolation coefficients, and $\mathbf{u}_x \in \mathbb{R}^m$ collects the nodal values of u_x on the stencil.

To approximate a differential operator $\mathcal{L}$ at the target node $\mathbf{x}_i$, the operator is applied to the interpolant:

$$\mathcal{L}u_x(\mathbf{x}_i) \approx \sum_{k=1}^m \lambda_k \mathcal{L}\phi(\mathbf{x}_i, \mathbf{x}_i^k) + \sum_{k=1}^q \mu_k \mathcal{L}p_k(\mathbf{x}_i). \quad (3.13)$$

The operator $\mathcal{L}$ on the solution $u_x(\mathbf{x})$ is approximated at the space point $\mathbf{x}_i$ as:

$$\mathcal{L}u_x(\mathbf{x}_i) \approx \sum_{k=1}^m w_{ik} u_x(\mathbf{x}_i^k), \quad (3.14)$$

where the differentiation weights $\{w_{ik}\}$ depend only on the stencil geometry and the chosen RBF. Substituting the interpolation coefficients into the expression above leads to the explicit formula for the weights:

$$\begin{bmatrix} \mathbf{w} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} A & P \\ P^\top & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{b} \\ \mathbf{c} \end{bmatrix}, \quad (3.15)$$

where $\mathbf{w} \in \mathbb{R}^m$ contains the RBF-FD differentiation weights associated with the stencil nodes, $\mathbf{v} \in \mathbb{R}^q$ corresponds to auxiliary weights associated with the polynomial constraints, $\mathbf{b} \in \mathbb{R}^m$ has entries $b_k = \mathcal{L}\phi(\mathbf{x}_i, \mathbf{x}_i^k)$, and $\mathbf{c} \in \mathbb{R}^q$ has entries $c_k = \mathcal{L}p_k(\mathbf{x}_i)$.

Solving this local system yields the weights $\mathbf{w}$, which depend solely on the stencil geometry and the chosen radial basis function. Once the RBF-FD weights are computed, spatial derivatives appearing in the governing PDEs can be evaluated directly from nodal values. In the present work, these discrete operators are employed to build the physics-informed loss function, enforcing the governing equations at the graph nodes without relying on automatic differentiation.

3.4.1. Geometric Quality: Fill Distance and Separation Distance

The accuracy and numerical stability of RBF-FD discretizations depend strongly on the geometric properties of the node distribution. Following the analysis of Le Borne and Leinen [59], two key quantities are the fill distance and the separation distance.

Given $X \subset \Omega$, where X denotes a node set and Ω denotes the computational domain of the flow problem, the fill distance $h_{X,\Omega}$ (radius of largest circle not intersecting with X) and the separation distance q_X (shortest distance of two disjoint nodes) are defined as:

$$h_{X,\Omega} := \sup_{\mathbf{x} \in \Omega} \min_{\mathbf{x}_j \in X} \|\mathbf{x} - \mathbf{x}_j\|_2, \quad q_X := \frac{1}{2} \min_{\substack{\mathbf{x}_i, \mathbf{x}_j \in X \\ i \neq j}} \|\mathbf{x}_i - \mathbf{x}_j\|_2. \quad (3.16)$$

The fill distance quantifies how well the point cloud covers the domain, while the separation distance measures the minimum spacing between nodes.

Figure 3.7 illustrates an example of a node distribution within the computational domain, together with the associated fill distance $h_{X,\Omega}$ and separation distance q_X .

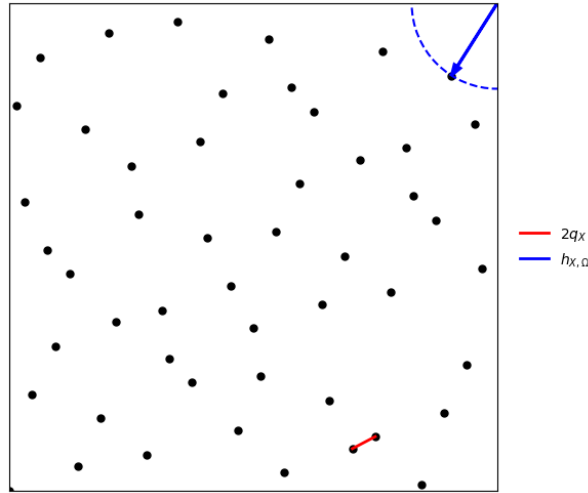


Figure 3.7: Example of a node distribution and the corresponding fill distance $h_{X,\Omega}$ and separation distance q_X .

The ratio $q_X/h_{X,\Omega}$ is often used as a practical indicator of stencil quality. Values of this ratio close to zero indicate highly non-uniform or clustered node distributions and may lead to ill-conditioned local interpolation matrices and unstable RBF-FD weights. In practice, quasi-uniform point clouds typically satisfy $q_X/h_{X,\Omega} = \mathcal{O}(1)$, while values below

approximately 0.1 are generally associated with poor geometric quality and potential numerical instability.

3.4.2. Differential Operators Used in This Work

In this work, we focus on the incompressible Navier-Stokes equations (2.1), which govern the dynamics of viscous, incompressible fluid flows.

To evaluate the spatial differential operators appearing in the momentum equations, the RBF-FD method is employed. In particular, RBF-FD is used to discretize:

- the Laplacian operator $\nabla^2 \mathbf{u}$, associated with viscous diffusion;
- the first-order spatial derivatives $\partial/\partial x$ and $\partial/\partial y$, required to compute the convective term $(\mathbf{u} \cdot \nabla) \mathbf{u}$, the divergence $\nabla \cdot \mathbf{u}$ and the pressure gradient ∇p .

The temporal derivative is discretized separately using a Crank–Nicolson (CN) scheme, providing second-order accuracy in time and improved stability for the evolution of the solution.

For each node $\mathbf{x}_i$, three sets of local RBF-FD differentiation weights are precomputed in order to discretize the spatial operators. Specifically, we compute:

$$\mathbf{A}_i = \{a_{ij}\}_{j \in \mathcal{N}(i)}, \quad \mathbf{B}_i = \{b_{ij}\}_{j \in \mathcal{N}(i)}, \quad \mathbf{C}_i = \{c_{ij}\}_{j \in \mathcal{N}(i)},$$

where $\mathcal{N}(i)$ denotes the local stencil associated with node $\mathbf{x}_i$. The weights $\mathbf{A}_i$ approximate the Laplacian operator, while $\mathbf{B}_i$ and $\mathbf{C}_i$ approximate the first-order derivatives in the x and y directions, respectively. Given the scalar field u_x defined at the nodes, representing the x -component of the velocity, the corresponding spatial derivatives at node $\mathbf{x}_i$ are approximated as

$$\nabla^2 u_x(\mathbf{x}_i) \approx \sum_{j \in \mathcal{N}(i)} a_{ij} u_x(\mathbf{x}_j), \quad (3.17)$$

$$\frac{\partial u_x}{\partial x}(\mathbf{x}_i) \approx \sum_{j \in \mathcal{N}(i)} b_{ij} u_x(\mathbf{x}_j), \quad (3.18)$$

$$\frac{\partial u_x}{\partial y}(\mathbf{x}_i) \approx \sum_{j \in \mathcal{N}(i)} c_{ij} u_x(\mathbf{x}_j). \quad (3.19)$$

These differentiation weights depend only on the geometry of the local stencil and are computed once prior to training. The same RBF-FD discretization applies analogously to the y -component u_y of the velocity and to the pressure field p .

The RBF-FD method requires choosing a local stencil around each node and specifying the type of radial basis function and the order of polynomial augmentation. In this work, the following configuration is adopted:

- **Stencil size:** 30 neighbouring nodes. This defines, for each central node, the local set of points used to construct the finite-difference weights.
- **Radial basis function:** polyharmonic spline PHS5, i.e. $\phi(r) = r^5$.
- **Polynomial augmentation:** the RBF basis is augmented with all monomials up to total degree 2, which enforces exactness of the scheme for quadratic polynomials and improves the approximation of differential operators.

These parameters were selected following a preliminary numerical validation on an analytical test function. In particular, the RBF-FD differentiation operators were applied to a known smooth function (e.g., $f(x, y) = \sin(x) + \cos(y)$), for which the exact spatial derivatives are available in closed form.

The discrete derivatives obtained through RBF-FD were then compared against the corresponding analytical derivatives, allowing us to assess the accuracy of different stencil sizes, radial basis functions, and polynomial augmentation orders. The selected configuration (stencil size 30, PHS5 basis, and polynomial order 2) provided the lowest approximation error while maintaining stable conditioning and moderate computational cost.

Finally, the resulting RBF-FD operators are embedded directly into the GNN model, where they are used to construct physics-based loss terms associated with the target differential operators.

The RBF-FD methodology offers several advantages in the context of physics-guided machine learning. By relying exclusively on nodal positions, it provides a fully meshless formulation that does not require a structured grid. The use of local stencils ensures both locality and sparsity, as each differential operator is constructed from a limited number of neighboring nodes, resulting in sparse and computationally efficient operators. Moreover, the method naturally adapts to irregular node distributions and curved geometries, making it well suited for complex computational domains. This localized stencil structure is also compatible with graph-based representations, as it aligns naturally with the neighborhood definitions used in message-passing graph neural networks.

At the same time, the RBF-FD approach is highly sensitive to stencil size and node distribution, requiring a meticulous tuning process to ensure numerical stability. From a computational standpoint, determining the differentiation weights involves solving a local linear system at each node, which introduces a non-negligible preprocessing overhead.

Crucially, suboptimal stencil configurations can lead to ill-conditioned or even singular local matrices, particularly when higher-order polynomial augmentation is employed. When such singularities occur, the weight computation fails, preventing the assembly of the final sparse differentiation operators. Therefore, while the global matrix might inherently possess better conditioning, its construction is strictly dependent on the successful and accurate resolution of these localized dense systems. Despite these limitations, the flexibility, locality and geometric adaptability of RBF-FD make it a suitable discretization strategy for integrating physical operators into graph-based PDE solvers.

3.5. Conclusions and next steps

This chapter has introduced the theoretical and methodological foundations underlying the proposed neural–numerical framework. The motivations for combining graph-based neural architectures with physics-informed learning have been discussed, together with the role of meshless discretization techniques based on RBF-FD for evaluating spatial differential operators on irregular domains.

The limitations of purely data-driven approaches and of classical physics-informed neural networks have been highlighted, motivating the adoption of a discrete, graph-based formulation that preserves physical consistency while remaining compatible with complex geometries and variable mesh resolutions. Together, these elements define the methodological basis of the approach developed in this thesis.

In the following chapters, this framework is applied to a series of increasingly complex fluid flow configurations. We first consider the canonical channel flow problem, which serves as a controlled test case for evaluating different learning paradigms and generalization across meshes (Chapter 4). Subsequently, more challenging scenarios involving channels with embedded obstacles and vortex-dominated dynamics are investigated, in order to assess the robustness and predictive capabilities of the proposed approach in geometrically complex settings (Chapter 5). Finally, the framework is extended to coupled fluid–structure interaction (FSI) problems, where the ability of the model to capture the interplay between fluid and structural dynamics is examined (Chapter 6).

4 | Channel Flow

This chapter presents the first application of the proposed MeshGraphNet framework to a canonical fluid dynamics benchmark: the incompressible channel flow. This configuration provides a controlled yet physically meaningful test case, in which essential features such as boundary layer development and shear-driven dynamics can be analyzed within a simple geometric setting.

Owing to the smoothness of the solution and the absence of separation or vortex-driven phenomena, the channel flow represents a favorable and well-controlled scenario. For this reason, the results presented in this chapter should be interpreted as a systematic validation of the methodological assumptions introduced in the previous chapters, and as an ablation study on the role of supervision and physics-informed constraints, rather than as a benchmark for complex or highly unsteady flow regimes.

The objectives of this chapter are twofold. First, the performance of the proposed surrogate model is evaluated under different learning paradigms, including fully supervised, partially supervised and fully unsupervised physics-informed training. This progressive reduction of supervision allows us to assess whether physical constraints can compensate for limited or absent labeled data in terms of both predictive accuracy and temporal stability.

Second, the generalization capabilities of the model are investigated by testing its performance across different mesh resolutions and topologies. In particular, we examine whether a model trained on a given discretization can accurately predict the flow dynamics on unseen meshes, thereby evaluating the robustness of the learned representation with respect to changes in spatial discretization within a controlled laminar regime.

The chapter is organized as follows. Sections 4.1 and 4.2 describe the numerical setup of the channel flow problem and the dataset generation procedure. The subsequent sections provide a systematic comparison between the fully supervised (see Section 4.3.1), partially unsupervised (see Section 4.3.3), and fully unsupervised training regimes (see Section 4.3.4). In particular, for the supervised case, an additional study on cross-mesh generalization is conducted (see Section 4.3.2).

4.1. Setup

To begin with, a simple reference case was considered: a two-dimensional laminar flow in a rectangular channel. This configuration was also used for the validation of the MeshGraphNet model in terms of its ability to reproduce smooth, steady-state laminar solutions.

The computational domain consists of a 2D channel with dimensions $2\text{ m} \times 1\text{ m}$. At the inlet, the velocity field is uniform along the y -direction and directed only along the x -axis. In other words, the inlet velocity profile is constant and can be expressed as

$$\mathbf{u}(x = 0, y) = (U_{\text{in}}, 0),$$

where U_{in} is the inlet velocity magnitude.

Under these conditions, the flow evolves toward a steady, incompressible laminar regime.

Figure 4.1 illustrates the computational domain adopted for the channel flow configuration.

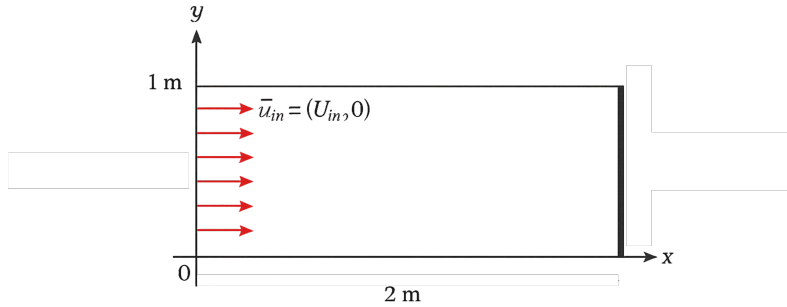


Figure 4.1: Two-dimensional computational domain of the channel flow problem. The rectangular domain has dimensions $2\text{ m} \times 1\text{ m}$ and is described in the Cartesian coordinate system (x, y) .

For fully developed laminar flow, the streamwise velocity follows the analytical parabolic solution associated with planar Poiseuille flow (see Section 2.1). For a channel of height $H = 1\text{ m}$, with the wall-normal coordinate $y \in [0, H]$, the fully developed streamwise velocity profile can be expressed in terms of the bulk velocity U_{bulk} as

$$u(y) = 6U_{\text{bulk}} \left(y - y^2 \right), \quad y \in [0, H]. \quad (4.1)$$

In the present configuration, the inlet velocity is uniform and equal to U_{in} . For incompressible steady flow, mass conservation implies that the volumetric flow rate is constant along the channel, and therefore $U_{\text{bulk}} = U_{in}$. The availability of this analytical solution makes the channel flow a suitable baseline case for model verification and comparison, allowing for a precise evaluation of the numerical errors introduced by the RBF-FD discretization.

4.2. OpenFOAM simulation

To construct a reliable reference dataset for the validation of the proposed learning framework, a two-dimensional laminar channel flow was simulated using OpenFOAM v2312 [38]. The numerical setup reproduces the same channel geometry ($2 \text{ m} \times 1 \text{ m}$) described in Section 4.1, ensuring full consistency between the physical problem definition and the reference solver.

The incompressible Navier-Stokes equations were solved under laminar flow assumptions using the transient solver icoFoam. Although the target solution corresponds to a steady-state regime, a transient formulation was deliberately adopted in order to generate time-resolved velocity and pressure fields, which are required for training and evaluating the autoregressive prediction capabilities of the MeshGraphNet model.

Time integration was performed using a first-order implicit Euler scheme. A fixed time step of $\Delta t = 0.01 \text{ s}$ was employed and the simulation was advanced up to a final time $T = 1 \text{ s}$, corresponding to 100 time steps. The time step was selected to satisfy the CFL condition throughout the domain. The solver log confirms that the Courant number remains well below unity during the entire run, with a maximum value $Co_{\text{max}} = 0.2985$ (mean $Co = 0.2062$) at the final time step. This ensures stable time integration and limits advection-related discretization errors in the generated reference dataset.

The boundary conditions were selected to reproduce a canonical fully developed laminar channel flow, and coincide with those described in Section 2.3. Specifically, a uniform inlet velocity, no-slip conditions at the walls, and a fixed reference pressure at the outlet were imposed. This configuration defines a well-posed benchmark problem with smooth solutions and well-understood physical behavior.

The Reynolds number is defined as

$$Re = \frac{U_{\text{in}} L}{\nu},$$

where U_{in} denotes the inlet velocity magnitude, L is the characteristic length scale and ν is the kinematic viscosity. For planar channel flow, the characteristic length is taken as the distance between the two parallel walls, $L = H = 1$ m.

For $U_{\text{in}} = 1$ m/s and $\nu = 0.1$ m²/s, the resulting Reynolds number is $Re = 10$, which lies well within the laminar regime. Under these conditions, the flow is characterized by smooth dynamics and admits a closed-form analytical solution in the fully developed region, as shown in Eq. (4.1).

After a short transient, the simulation converges to a steady laminar regime, as illustrated in Fig. 4.2. The velocity field develops the expected parabolic profile characteristic of planar Poiseuille flow, with maximum velocity at the channel centerline and zero velocity at the walls due to the no-slip condition.

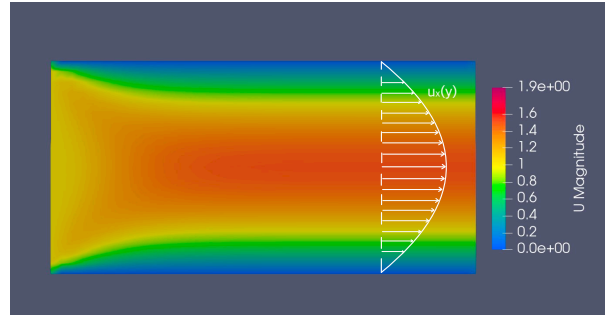


Figure 4.2: Velocity magnitude at the final simulation time, showing the fully developed parabolic Poiseuille profile.

The corresponding pressure field exhibits an approximately linear decrease along the streamwise direction, reflecting the viscous losses required to sustain the flow, as shown in Fig. 4.3. Minor deviations from perfect linearity are observed near the inlet region. These are mainly associated with the developing flow, where the initially uniform inlet velocity progressively evolves toward the fully developed parabolic Poiseuille profile. This transitional adjustment slightly perturbs the local pressure distribution but does not affect the global streamwise pressure gradient nor the overall solution accuracy.



Figure 4.3: Pressure field at the final simulation time, showing the expected linear stream-wise gradient.

Overall, the OpenFOAM simulation accurately reproduces the known analytical behavior of a two-dimensional laminar channel flow. The resulting time-resolved velocity and pressure fields $\mathbf{u}(x, y, t)$ and $p(x, y, t)$, collected over the interval $[0, 1]$ s, are therefore adopted as a high-fidelity reference dataset for the validation of the MeshGraphNet model in the subsequent sections.

4.3. Results

In this section, the proposed MeshGraphNet framework is systematically evaluated in a controlled laminar setting, with the objective of validating the training strategy and isolating the effects of supervision and physics-based regularization.

The study is organized into three progressive stages, designed to progressively reduce the amount of available supervision and to assess the extent to which physical constraints can replace data-driven information without compromising accuracy or stability.

This gradual approach allowed for systematic evaluation of performance of the model and its ability to generalize beyond the reference dataset obtained from the OpenFOAM simulation.

4.3.1. Supervised Case

Domain, graph and node features

The computational domain reproduces the same geometry used in OpenFOAM ($2 \text{ m} \times 1 \text{ m}$) and is discretized as a regular lattice with $n_x \times n_y = 40 \times 20$ cells. In the graph representation, nodes are located at the grid vertices, resulting in $(n_x+1) \times (n_y+1) = 41 \times 21 = 861$ nodes in total. The graph connectivity is explicitly defined on the structured

lattice: each node is connected to its immediate horizontal and vertical neighbors with bidirectional edges, as shown in Figure 4.4.

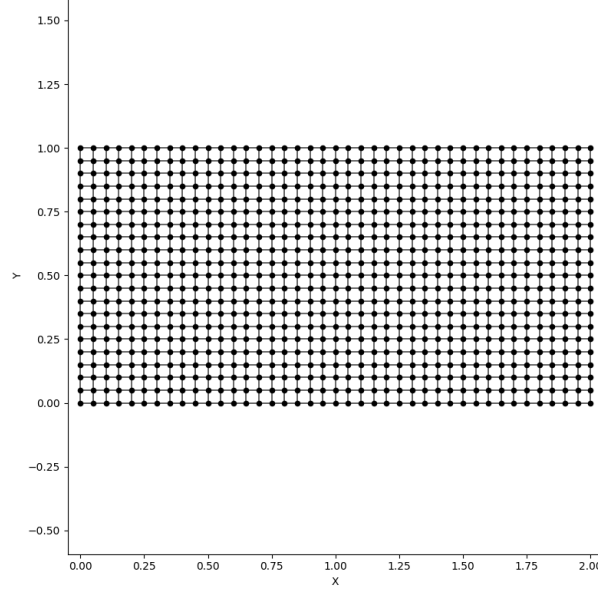


Figure 4.4: Graph representation of the structured lattice used for the channel domain. Each node is connected to its immediate horizontal and vertical neighbors through bidirectional edges, forming a regular grid topology.

Each node is assigned a type according to its spatial location through a one-hot encoded type vector:

$$\text{type}_i = \begin{cases} \text{INFLOW}, & x_i = 0, \\ \text{OUTFLOW}, & x_i = 2, \\ \text{WALL_BOUNDARY}, & y_i \in \{0, 1\}, \\ \text{NORMAL}, & \text{otherwise.} \end{cases}$$

The one-hot type vector is a binary categorical encoding in which each node is represented by a vector of zeros and ones, with a single unit entry indicating its physical role in the domain (inflow, outflow, wall_boundary, or normal). The normal label denotes the interior nodes of the computational domain, i.e., nodes that do not belong to any boundary. This representation allows the network to distinguish between inflow, outflow, wall boundary and normal nodes without introducing any artificial ordering. The node-type encoding is also used to construct spatial masks, assigning nodes to the appropriate subsets for the different loss terms.

The complete node features vector is defined as:

$$\mathbf{x}_i = [x_i, y_i, \text{one_hot_type_vector}_i],$$

where (x_i, y_i) are the spatial coordinates, and the one-hot vector specifies the physical and geometric role of each node in the domain. This feature formulation enables the graph neural network to recognize both spatial position and boundary condition context.

Dataset and normalization

The dataset consists of nine laminar channel simulations with inlet velocities ranging from $U_{\text{in}} = 1.0$ to 2.0 m/s. This range was selected to span multiple steady laminar regimes within the same geometric configuration, enabling assessment of the interpolation capability of the model across different flow intensities. Each simulation provides time-resolved velocity and pressure fields computed with OpenFOAM under steady laminar conditions.

To evaluate interpolation capability of the model, training was performed on all inlet velocity cases except for $U_{\text{in}} = 1.6$ m/s, which was used exclusively for testing. The test velocity was selected as an intermediate value within the training range, so that generalization performance could be attributed to interpolation across laminar flow conditions rather than to extrapolation effects or regime changes.

Each dataset, corresponding to a specific inlet velocity, was normalized independently. This choice was deliberately adopted to decouple the learning of flow structure from absolute magnitude effects and to prevent information leakage across inlet conditions. By computing normalization statistics separately for each simulation, the model is required to rely on relative spatial variations rather than on the absolute scale of the flow variables.

For each physical variable (u_x , u_y and p), the mean and standard deviation were computed over all nodes and time steps of the corresponding dataset. For instance, the streamwise velocity component was normalized as

$$\tilde{u}_x = \frac{u_x - \mu_{u_x}}{\sigma_{u_x}},$$

where μ_{u_x} and σ_{u_x} denote the empirical mean and standard deviation of u_x , respectively. An analogous normalization procedure was applied to u_y and p .

This normalization improves numerical stability during training and ensures a consistent scaling between variables with different physical units and magnitudes. During optimiza-

tion, the network operates exclusively on normalized quantities. In the testing phase, predictions are denormalized using the statistics (μ, σ) associated with the test dataset ($U_{\text{in}} = 1.6 \text{ m/s}$), allowing a physically meaningful evaluation of the predicted velocity and pressure fields.

Model, outputs and loss

The MeshGraphNet predicts the three primary physical quantities characterizing an incompressible flow at each node:

$$\hat{\mathbf{y}}_i = (u_{x,i}, u_{y,i}, p_i),$$

where $u_{x,i}$ and $u_{y,i}$ denote the velocity components in the streamwise (x) and wall-normal (y) directions at node i , and p_i is the corresponding pressure value. This choice reflects the minimal set of variables required to describe the incompressible Navier-Stokes equations. In the present channel-flow configuration, the reference solution corresponds to a steady-state regime, and therefore the predicted fields do not explicitly depend on time.

In the fully supervised setting, training is performed by minimizing a weighted mean squared error (MSE) between the predicted fields and the OpenFOAM reference solution:

$$\mathcal{L}_{\text{sup}} = w_{\text{vel}} \text{MSE}(\mathbf{u}_{\text{pred}}, \mathbf{u}_{\text{true}}) + w_p \text{MSE}(p_{\text{pred}}, p_{\text{true}}), \quad \text{with} \quad w_{\text{vel}} = 1.0, \quad w_p = 0.7.$$

Here, w_{vel} and w_p are dimensionless weighting coefficients that balance the relative contribution of the velocity and pressure errors in the total loss. A lower weight is assigned to the pressure term to account for its different physical scaling and for the fact that pressure primarily influences the dynamics through its gradient rather than through its absolute value. The loss is computed on all nodes of the graph and for every time step, enforcing global consistency between the predicted and reference fields across both space and time.

Training protocol

Figure 4.5 illustrates the overall architecture adopted in this work, including the encoder, message-passing processor, decoder, and loss computation.

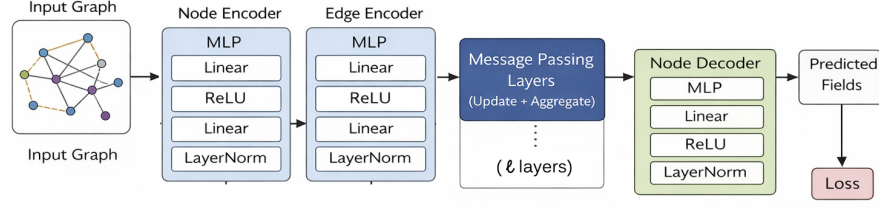


Figure 4.5: Schematic representation of the MeshGraphNet architecture adopted in this study. The input graph is first projected into a latent space through node and edge encoders. Information is then propagated through l message-passing layers before being mapped to the physical output fields by a node-wise decoder. The predicted fields are finally used to compute the training loss.

In particular, the network was trained using the Adam optimizer combined with a cosine learning-rate annealing schedule over 1000 epochs. The resulting architecture comprises approximately 6.3×10^5 trainable parameters, which provides sufficient expressive capacity to capture the smooth laminar dynamics of the flow while remaining computationally tractable.

The cosine annealing strategy progressively reduces the learning rate during training, promoting stable convergence and mitigating oscillations in the later optimization stages. In addition, a progressive temporal rollout strategy was adopted: at early epochs, the model predicts only a limited number of future time steps, while the rollout horizon is gradually extended as training proceeds. Although the target flow regime is steady and laminar, this strategy is introduced here to maintain methodological consistency with the training setup employed in more complex and unsteady configurations discussed in subsequent chapters.

To further stabilize autoregressive training, a teacher forcing mechanism was employed and linearly annealed to zero. During the initial training stages, ground-truth states are partially reintroduced to prevent error accumulation, whereas in later epochs the model relies entirely on its own predictions.

Boundary conditions were explicitly enforced through node-wise masks fixing the inlet velocity and imposing no-slip conditions at the walls. This ensures that the learned dynamics remain consistent with the physical constraints of the problem and prevents boundary-induced instabilities during rollout. Finally, gradient clipping with a maximum norm of 1.0 was applied at each optimization step to enhance numerical stability and prevent gradient explosion.

Table 4.1: Summary of the supervised training setup for the MeshGraphNet model.

Parameter	Symbol / Value
Optimizer	Adam
Initial learning rate	$\eta_0 = 1 \times 10^{-4}$
Weight decay	5×10^{-4}
Learning rate scheduler	Cosine annealing, $\eta_t = \eta_0 \cdot \frac{1}{2} \left(1 + \cos \left(\frac{\pi t}{T_{\max}} \right) \right)$
Teacher forcing	Linear annealing to zero
Gradient clipping	$\ \nabla\ _{\max} = 1.0$

The main hyperparameters and optimization choices adopted for the supervised training of the MeshGraphNet model are summarized in Table 4.1.

Results

The model successfully reproduced the laminar channel flow for all training inlet velocities and generalized effectively to the unseen case $U_{\text{in}} = 1.6$ m/s. After denormalization, the predicted velocity fields exhibited the expected parabolic (Poiseuille) profile, with a centerline maximum and zero velocity at the walls (see Fig. 4.6 and 4.7), while the pressure field maintained a nearly linear gradient along the streamwise direction (See Fig. 4.8).

The quantitative evaluation was carried out on the test configuration with inlet velocity $U_{\text{in}} = 1.6$ m/s. For the streamwise velocity component u_x , the discrete L_2 error norm was computed over all N graph nodes as

$$L_2(u_x) = \sqrt{\frac{1}{N} \sum_{i=1}^N \left(u_{x,i}^{\text{pred}} - u_{x,i}^{\text{true}} \right)^2}. \quad (4.2)$$

Here, $u_{x,i}^{\text{pred}}$ denotes the predicted streamwise velocity at node i , $u_{x,i}^{\text{true}}$ is the corresponding OpenFOAM reference value, and N is the total number of nodes in the computational domain. The same definition is applied analogously to compute the L_2 error for the wall-normal velocity component u_y and for the pressure field p , replacing u_x with the corresponding field variable in the expression above.

The results confirm that the model preserved both spatial accuracy and physical consistency across the entire rollout sequence. The mean L_2 error is reported in the Table 4.2.

$L_2(u_x)$	$L_2(u_y)$	$L_2(p)$
2.644×10^{-2}	1.045×10^{-2}	1.855×10^{-1}

Table 4.2: L_2 errors for velocity components and pressure.

The results reported in Table 4.2 show that the velocity components are predicted with high accuracy, with errors of order 10^{-2} . In particular, the smaller error on u_y is consistent with the physics of laminar channel flow, where the wall-normal velocity remains close to zero throughout the domain. The pressure error is approximately one order of magnitude larger, which is expected since pressure acts as a global constraint in incompressible flows and is more sensitive to small deviations. Overall, the error levels confirm that the model accurately reconstructs both the velocity profile and the pressure distribution while maintaining stable autoregressive behavior.

While Table 4.2 provides a quantitative assessment of the global error magnitude, it does not provide information about the spatial distribution of the discrepancies. For this reason, Figs 4.6, 4.7 and 4.8 report a qualitative comparison between the predicted fields, the corresponding L_2 error maps, and the OpenFOAM reference solution. A closer inspection reveals that the largest discrepancies are primarily located near the inlet region. This behavior is physically consistent with the problem setup: the imposed uniform inlet profile progressively adjusts toward the fully developed parabolic Poiseuille distribution, generating a developing region where velocity gradients are stronger. In this zone, even small deviations may result in comparatively larger local errors. Despite these localized discrepancies, the dominant flow dynamics are accurately captured. The streamwise velocity clearly exhibits the expected parabolic profile, the wall-normal component remains close to zero across the domain, and the pressure field preserves its global linear gradient. This confirms that the model correctly reproduces the fundamental structure of the laminar channel flow. A slightly increased error is also observed near the outlet. This can be attributed to the sensitivity of the pressure field to boundary conditions and to the accumulation of small autoregressive deviations toward the downstream region. However, these effects remain localized and do not alter the overall solution structure.

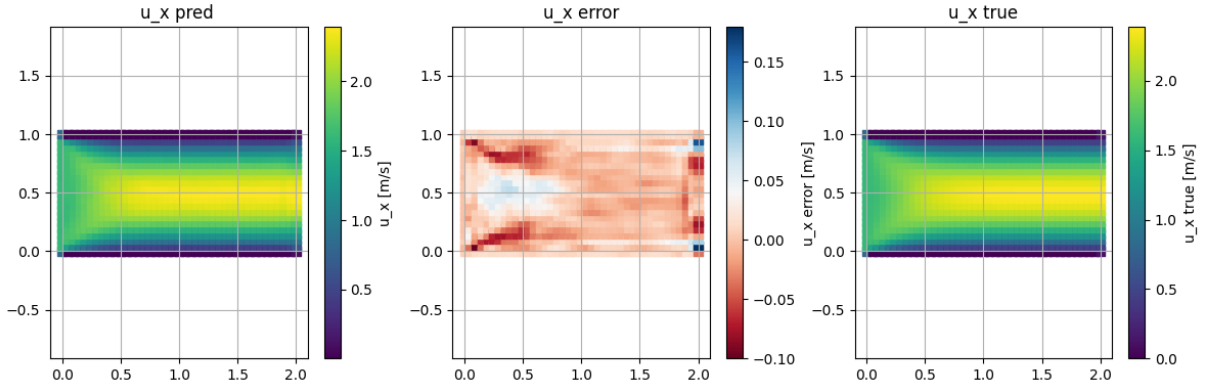


Figure 4.6: Predicted, L2 error and OpenFOAM results for the velocity component along the x -direction (u_x).

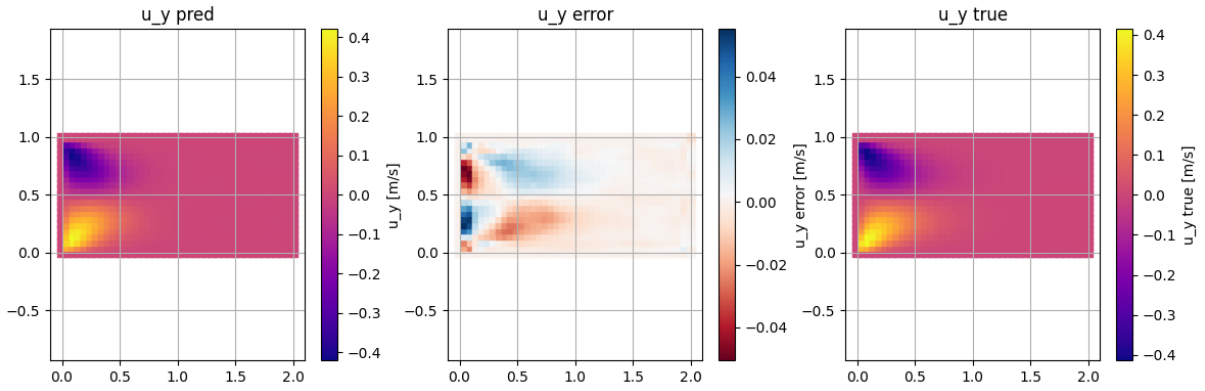


Figure 4.7: Predicted, L2 error and OpenFOAM results for the velocity component along the y -direction (u_y).

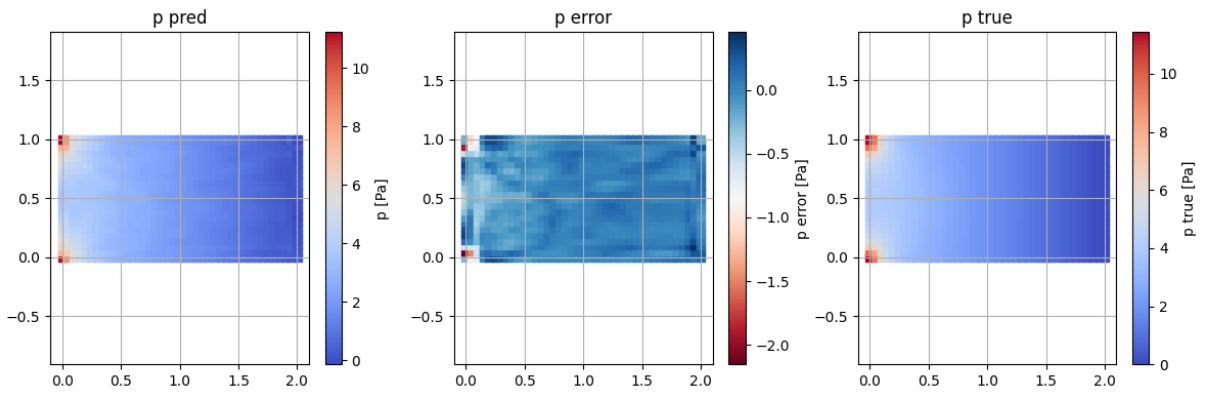


Figure 4.8: Predicted, L2 error and OpenFOAM results of the pressure (p) for the test case.

Together, the quantitative errors and qualitative comparisons indicate that the model accurately captures the steady laminar solution without introducing spurious oscillations or boundary artefacts. Overall, the supervised model generalizes well to the unseen inlet velocity, maintaining smooth, physically consistent flow predictions and low error levels across all quantities of interest.

It is important to note that this supervised success is obtained in a controlled laminar regime: the dynamics are weakly nonlinear, the solution remains laminar and regular, and the boundary conditions are simple. As a consequence, low rollout errors in this setting should not be over-interpreted as evidence of robustness in the presence of separation, unsteady wakes, or turbulence; these aspects are explicitly addressed in Chapter 5.

4.3.2. Generalization across meshes

To further assess the robustness and generalization capabilities of the proposed model, a set of additional supervised experiments was conducted by varying both the mesh resolution and the inlet velocity. The aim of this analysis was to verify whether the MeshGraphNet is able to produce consistent and physically meaningful predictions when applied to spatial discretizations finer than those seen during training, as well as to flow conditions corresponding to intermediate inlet velocities not included in the training set. In particular, this experiment investigates the resolution independence of the learned representation, testing whether the model captures the underlying continuous flow dynamics rather than overfitting to a specific mesh topology. At the same time, it evaluates the ability of the model to interpolate between known physical regimes, assessing its robustness to simultaneous changes in spatial resolution and flow parameters.

Training and test configurations

The model was trained on datasets generated using four different mesh resolutions:

$$20 \times 10, \quad 30 \times 10, \quad 30 \times 15, \quad 35 \times 15.$$

Each of these meshes represents the same physical domain ($2 \text{ m} \times 1 \text{ m}$), but with increasing node density and different inlet velocities. The inlet velocities used for training ranged from $U_{\text{in}} = 1.0$ to 2.5 m/s , and were selected to span a sufficiently wide range of steady laminar flow conditions, ensuring adequate variability and size of the training dataset. In total, 14 distinct mesh–velocity pairs were used for training. For each configuration, OpenFOAM simulations provided time-resolved reference data for velocity and pressure fields.

For testing, a significantly finer mesh (60×30) was used, corresponding to $(61 \times 31) = 1891$ nodes, combined with an inlet velocity of $U_{\text{in}} = 1.6$ m/s. This configuration was never seen by the model during training, allowing assessment of its simultaneous generalization to an unseen spatial resolution and an intermediate flow condition.

Results

The model exhibited strong generalization capabilities across both mesh density and inlet velocity. When evaluated on the unseen 60×30 mesh with $U_{\text{in}} = 1.6$ m/s, the predicted fields preserved smooth and physically coherent spatial distributions consistent with the OpenFOAM reference. The velocity field maintained the expected parabolic Poiseuille profile (see Figs. 4.9 and. 4.10), while the pressure field showed the correct linear gradient along the channel (see Fig. 4.11). A slightly larger error is observed in the vicinity of the inlet, particularly for the wall-normal velocity component u_y . This behavior is consistent with the presence of a developing region, where the initially uniform inlet profile has not yet fully evolved toward the steady parabolic distribution. In this area, stronger local gradients amplify small discrepancies between prediction and reference solution. No spurious oscillations or numerical artifacts were observed even in the finer discretization.

Quantitatively, the mean L_2 errors for the velocity components remained comparable to those of the training meshes, as reported in Table 4.3. The pressure error, however, increases markedly on the unseen configuration, highlighting its greater sensitivity to global deviations despite the correct reconstruction of its overall spatial distribution.

$L_2(u_x)$	$L_2(u_y)$	$L_2(p)$
4.023×10^{-2}	6.998×10^{-3}	9.065×10^{-1}

Table 4.3: L_2 errors for velocity components and pressure.

These results confirm that the model successfully generalizes across both physical and spatial variations, effectively learning a continuous representation of the laminar flow field that remains consistent across the considered mesh resolutions and inlet conditions.

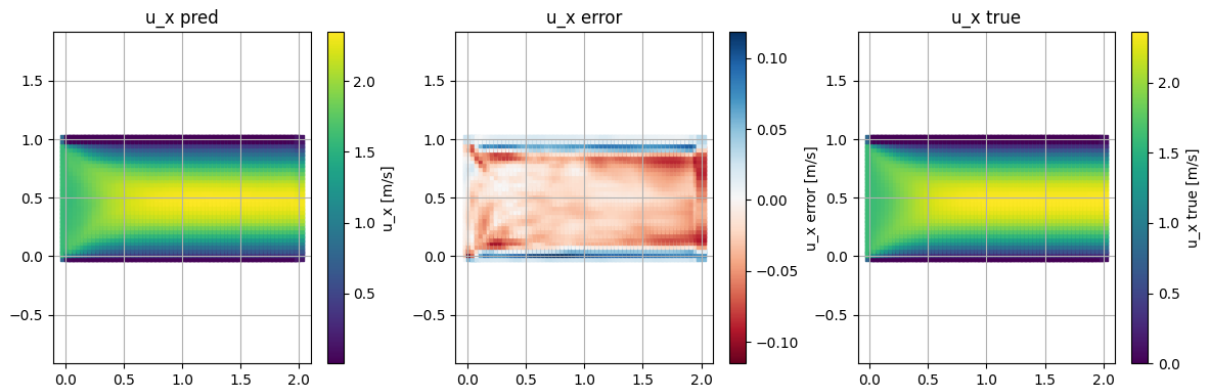


Figure 4.9: Predicted, L2 error and OpenFOAM results for the velocity component along the x -direction (u_x) for the finer mesh.

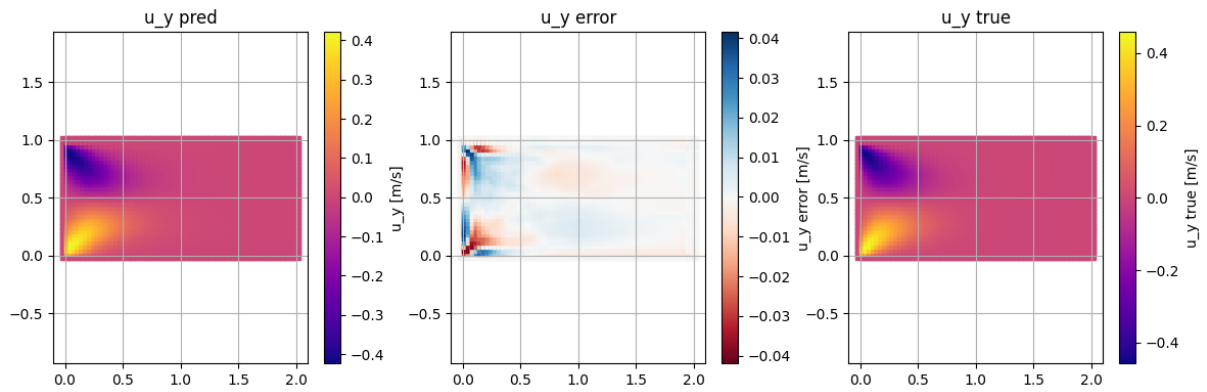


Figure 4.10: Predicted, L2 error and OpenFOAM results for the velocity component along the y -direction (u_y) for the finer mesh.

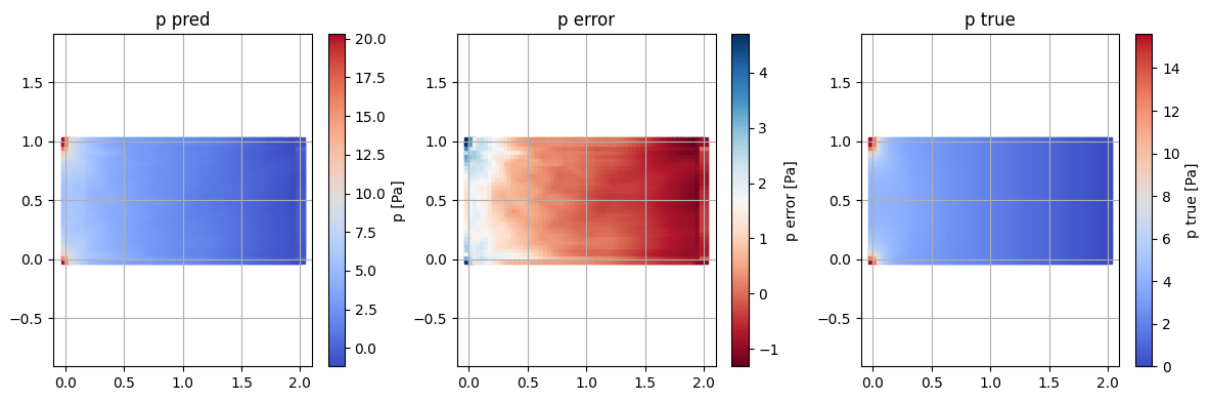


Figure 4.11: Predicted, L2 error and OpenFOAM results for the pressure for the finer mesh.

Overall, the results indicate that the model does not overfit the training discretizations but instead learns the underlying continuous fluid dynamics, enabling accurate transfer to new meshes and intermediate flow conditions.

4.3.3. Partially unsupervised Case

In this configuration, the same MeshGraphNet architecture and the same mesh used in the supervised case (41×21 nodes) were adopted (see Section 4.3.1). The key difference lies in the training objective, which combines both data-driven and physics-based components in a hybrid loss formulation. The goal of this experiment was to reduce the dependence on ground-truth data by introducing partial physical supervision through the Navier–Stokes equations.

Loss formulation

The total loss is defined as a weighted sum of several terms:

$$\mathcal{L}_{\text{total}} = w_{\text{vel}}\mathcal{L}_{\text{vel}} + w_p\mathcal{L}_p + w_{\text{div}}\mathcal{L}_{\text{div}} + w_{\text{res}}\mathcal{L}_{\text{res}} + w_{\text{tv}}\mathcal{L}_{\text{tv}}.$$

The first two components ($\mathcal{L}_{\text{vel}}, \mathcal{L}_p$) are supervised losses based on the mean squared error between predicted and reference velocity and pressure values, computed only on boundary and outer regions of the domain. The inner region of the domain is instead trained in an unsupervised manner, by minimizing the physical residuals of the Navier–Stokes equations and the incompressibility constraint. Specifically:

- **Momentum residuals:**

$$\begin{aligned}\mathcal{L}_{\text{res}} &= \langle (R_x)^2 + (R_y)^2 \rangle, \\ R_x &= \frac{\partial u_x}{\partial t} + u_x \frac{\partial u_x}{\partial x} + u_y \frac{\partial u_x}{\partial y} + \frac{\partial p}{\partial x} - \nu \nabla^2 u_x, \\ R_y &= \frac{\partial u_y}{\partial t} + u_x \frac{\partial u_y}{\partial x} + u_y \frac{\partial u_y}{\partial y} + \frac{\partial p}{\partial y} - \nu \nabla^2 u_y,\end{aligned}$$

where $\nu = 0.1 \text{ m}^2/\text{s}$ is the kinematic viscosity.

- **Incompressibility:**

$$\mathcal{L}_{\text{div}} = \langle (\nabla \cdot \mathbf{u})^2 \rangle = \left\langle \left(\frac{\partial u_x}{\partial x} + \frac{\partial u_y}{\partial y} \right)^2 \right\rangle.$$

- **Total variation regularization:** penalizes large variations of the predicted fields

between adjacent graph nodes, improving smoothness and physical consistency.

$$\mathcal{L}_{\text{tv}} = \left\langle \frac{1}{|\mathcal{E}_\Omega|} \sum_{(i,j) \in \mathcal{E}_\Omega} \left[(u_{x,i} - u_{x,j})^2 + (u_{y,i} - u_{y,j})^2 + \alpha(p_i - p_j)^2 \right] \right\rangle.$$

where $\mathcal{E}_\Omega$ denotes the set of edges connecting nodes inside the considered subgraph, and $\alpha = 0.1$ is a weighting coefficient for the pressure term.

The weights adopted for the different terms were:

$$w_{\text{vel}} = 1.0, \quad w_p = 1.0, \quad w_{\text{div}} = 1.0, \quad w_{\text{res}} = 1.0, \quad w_{\text{tv}} = 10^{-2}.$$

Training setup

The network was trained using the same model as before. The dataset consists of a single laminar channel flow trajectory with inlet velocity $U_{\text{in}} = 1.0$ m/s, used for both training and testing. During training, a rectangular region covering approximately 80% of the domain area was defined as the unsupervised zone, where only the physical losses were applied (See Fig. 5.17). The outer region (inlet, walls and outlet) remained supervised through data-based MSE terms.

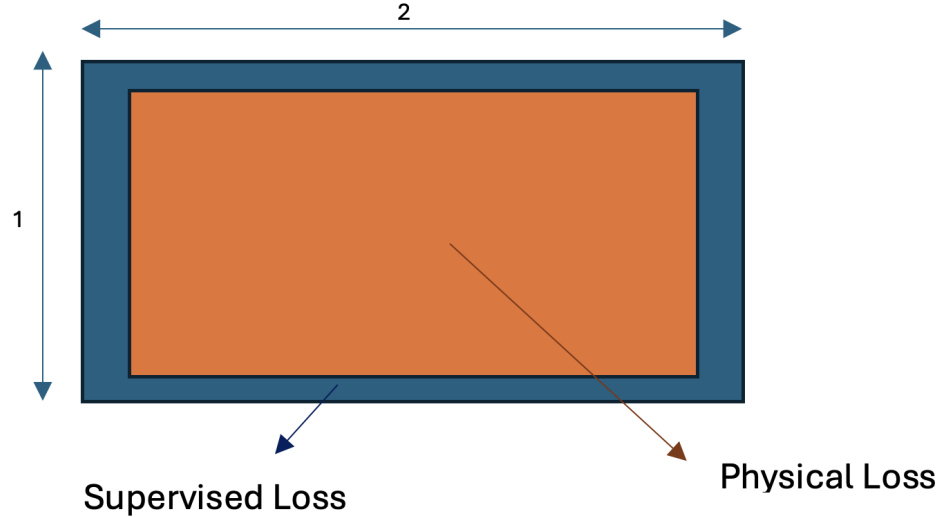


Figure 4.12: Division of the domain in supervised mask (20%) and unsupervised mask (80%)

Spatial derivatives required for the Navier–Stokes residuals were computed using differential matrices based on the RBF–FD method, which allows flexible and mesh-independent

differentiation on irregular graphs. A detailed description of these operators is provided in Section 3.4.

Results

The partially unsupervised model reproduced the laminar channel flow with high accuracy, even though part of the domain was trained exclusively through physics-based losses.

As shown in Fig. 4.13, the velocity loss during training exhibits a clear and monotonic decreasing trend over the epochs, indicating stable convergence of the model.

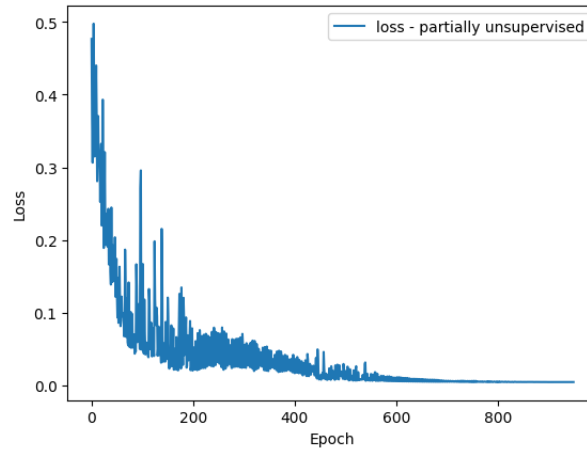


Figure 4.13: Velocity loss over epochs

The predicted velocity field preserved the expected parabolic shape (See Fig. 4.14 and 4.15), and the pressure distribution remained linearly decreasing along the channel (See Fig. 4.16). A detailed inspection shows that the largest discrepancies are consistently concentrated near the inlet region for all quantities (u_x , u_y , and p). This behavior is physically coherent with the problem setup. At the inlet, a uniform velocity profile is imposed, while the fully developed Poiseuille distribution emerges progressively along the streamwise direction. The transition from the imposed boundary condition to the equilibrium parabolic profile generates stronger local gradients, making this region more sensitive to small prediction errors. In contrast, in the fully developed portion of the channel, the solution is smooth and dominated by a well-defined balance between pressure gradient and viscous diffusion. In this region, the predicted fields closely match the reference solution, confirming that the dominant laminar dynamics are correctly captured by the hybrid physics-informed training.

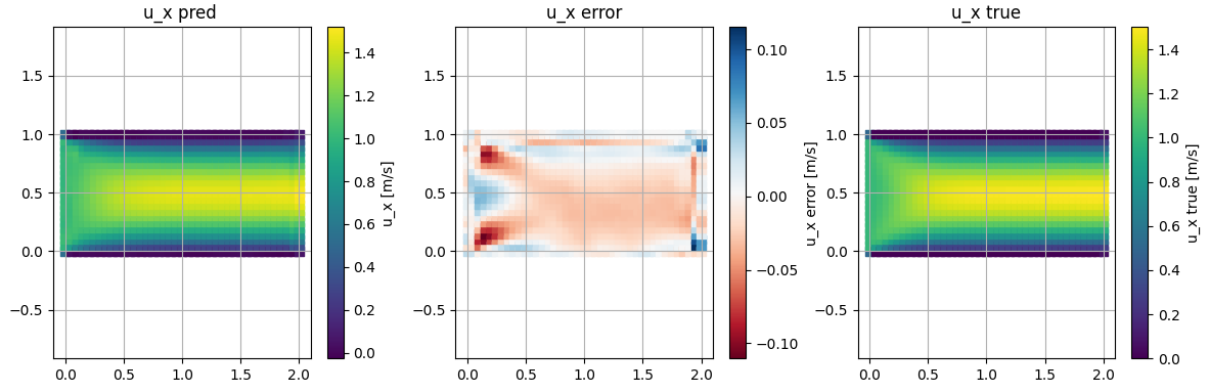


Figure 4.14: Predicted, L2 error and OpenFOAM results for the velocity component along the x -direction (u_x).

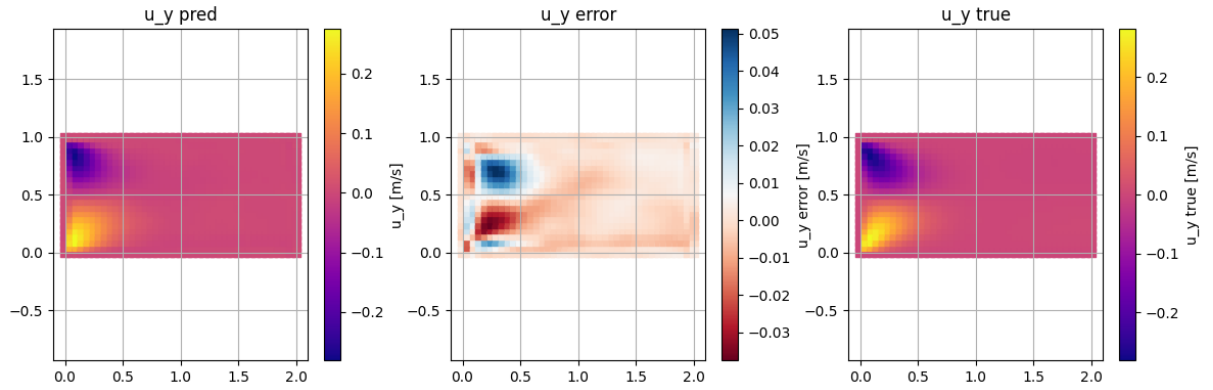


Figure 4.15: Predicted, L2 error and OpenFOAM results for the velocity component along the y -direction (u_y).

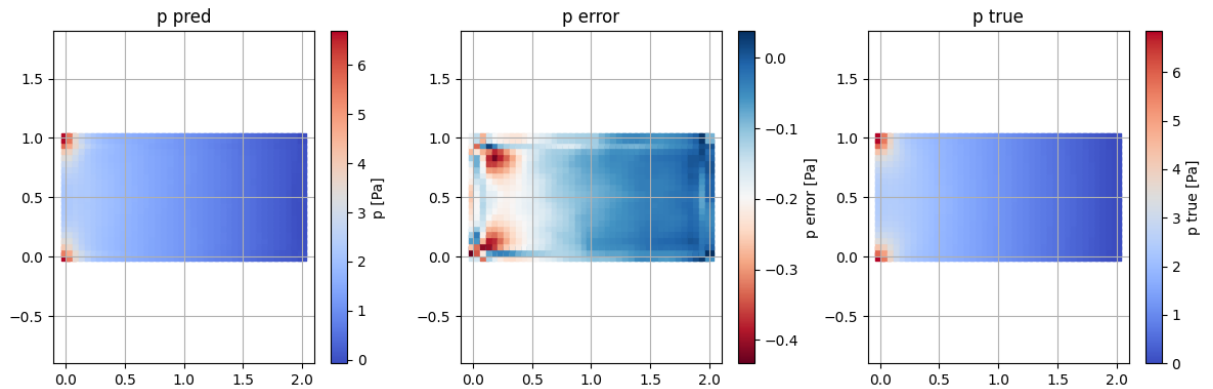


Figure 4.16: Predicted, L2 error and OpenFOAM results for the pressure (p).

Compared to the purely supervised case, the overall L_2 errors remain very close in magnitude. The velocity components retain errors of order 10^{-2} , nearly identical to the fully

supervised configuration. The pressure error, of order 10^{-1} , exhibits a slight decrease, suggesting that the residual-based regularization does not compromise, and may even modestly improve, the pressure prediction.

Quantity	Symbol	Mean L_2 Error
Velocity (x-direction)	u_x	2.71×10^{-2}
Velocity (y-direction)	u_y	1.09×10^{-2}
Pressure	p	1.34×10^{-1}

Table 4.4: Mean L_2 errors for velocity and pressure fields in the partially unsupervised case ($U_{\text{in}} = 1.0$ m/s).

This confirms that the physics-informed terms effectively constrain the solution and compensate for the absence of direct supervision in the central region.

Overall, this hybrid strategy demonstrates that even limited supervision combined with physical regularization can yield accurate and physically meaningful flow predictions.

4.3.4. Fully Unsupervised Case

We now consider the fully unsupervised configuration; the same MeshGraphNet architecture, node features and mesh resolution (40×20 nodes) used in the previous cases were retained (See Section 4.3.1 for details). The training setup and optimization strategy were also identical. The only difference lies in the loss formulation, which now excludes all direct supervision from reference data. Except for the inlet and wall boundaries, where physical boundary conditions are enforced through penalty terms, the entire domain is trained solely via physics-based constraints.

Loss formulation and boundary constraints

In the fully unsupervised configuration, the training objective is entirely physics-based and incorporates both interior residual terms and boundary constraints. The total loss is defined as

$$\mathcal{L}_{\text{unsup}} = w_{\text{res}}\mathcal{L}_{\text{res}} + w_{\text{div}}\mathcal{L}_{\text{div}} + w_{\text{BC}}\mathcal{L}_{\text{BC}} + w_{\text{tv}}\mathcal{L}_{\text{tv}} + w_{\text{out}}\mathcal{L}_{\text{out}}.$$

The terms $\mathcal{L}_{\text{res}}$, $\mathcal{L}_{\text{div}}$, and $\mathcal{L}_{\text{tv}}$ are defined as in the partially unsupervised configuration (see Section 4.3.3). They enforce, respectively, the Navier–Stokes momentum residuals, the incompressibility constraint, and spatial smoothness through total variation regular-

ization.

Although no supervision is used inside the domain, boundary conditions remain essential to constrain the solution. At the inlet, the prescribed velocity profile is imposed through a boundary mask included in $\mathcal{L}_{\text{BC}}$. Along solid walls, the no-slip condition ($\mathbf{u} = 0$) is similarly enforced via masked penalty terms.

At the outlet, no direct supervision is applied. Instead, a dedicated outflow term $\mathcal{L}_{\text{out}}$ softly enforces physically consistent downstream behavior:

$$\mathcal{L}_{\text{out}} = \mathcal{L}_{\text{out}}^{(u_x, u_y)} + \mathcal{L}_{\text{out}}^{(p)}.$$

The velocity component,

$$\mathcal{L}_{\text{out}}^{(u_x, u_y)} = \left\langle \left(\frac{\partial u_x}{\partial x} \right)^2 + \left(\frac{\partial u_y}{\partial x} \right)^2 \right\rangle_{\text{outlet}},$$

penalizes streamwise gradients at the downstream boundary, approximating a zero-gradient (free outflow) condition and reducing artificial reflections.

The pressure component,

$$\mathcal{L}_{\text{out}}^{(p)} = \langle p^2 \rangle_{\text{outlet}},$$

anchors the pressure to a fixed reference level, removing the additive constant indeterminacy of the incompressible Navier–Stokes equations.

Together, these boundary-related terms ensure that the learned solution remains consistent with the physical boundary behavior of the reference CFD simulations, while allowing the internal flow dynamics to be inferred without any direct supervision.

Results

As shown in Fig. 4.17, the loss associated with the velocity component along the x -direction exhibits a monotonically decreasing trend and progressively approaches zero, indicating stable convergence of the velocity prediction.

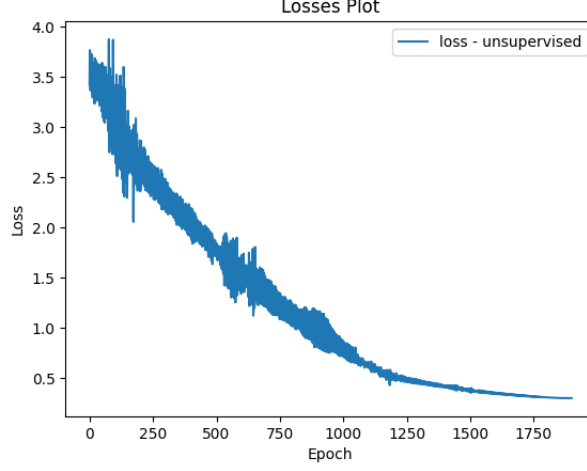


Figure 4.17: Velocity loss over training epochs.

Despite the presence of the outlet anchoring term, the pressure predictions obtained during the initial unsupervised experiments exhibited a noticeable drift, as shown in Fig. 4.20. While the velocity field largely converged to the expected laminar profile, its accuracy degraded specifically in the proximity of the outlet boundary. This suggests that the soft pressure anchoring, while addressing the gauge freedom, introduces a numerical imbalance that interferes with momentum conservation, as illustrated in Figs. 4.18 and 4.19.

The global drift is related to a fundamental property of the incompressible Navier-Stokes equations: the pressure field is defined only up to an arbitrary additive constant (gauge freedom). Indeed, if $p(\mathbf{x}, t)$ is a valid solution, then

$$p'(\mathbf{x}, t) = p(\mathbf{x}, t) + C(t), \quad (4.3)$$

with arbitrary $C(t)$, yields identical velocity dynamics since

$$\nabla p'(\mathbf{x}, t) = \nabla p(\mathbf{x}, t). \quad (4.4)$$

Although the outlet loss term

$$\mathcal{L}_{\text{out}}^{(p)} = \langle p^2 \rangle_{\text{outlet}}$$

acts to anchor the pressure at the downstream boundary, it constitutes a soft constraint applied only on a subset of nodes. Consequently, it does not fully eliminate the gauge freedom at the level of the optimization problem. In practice, the network may still generate global shifts or local inconsistencies. Without any supervision or boundary reference, the network cannot determine this offset, leading to pressure drift during training.

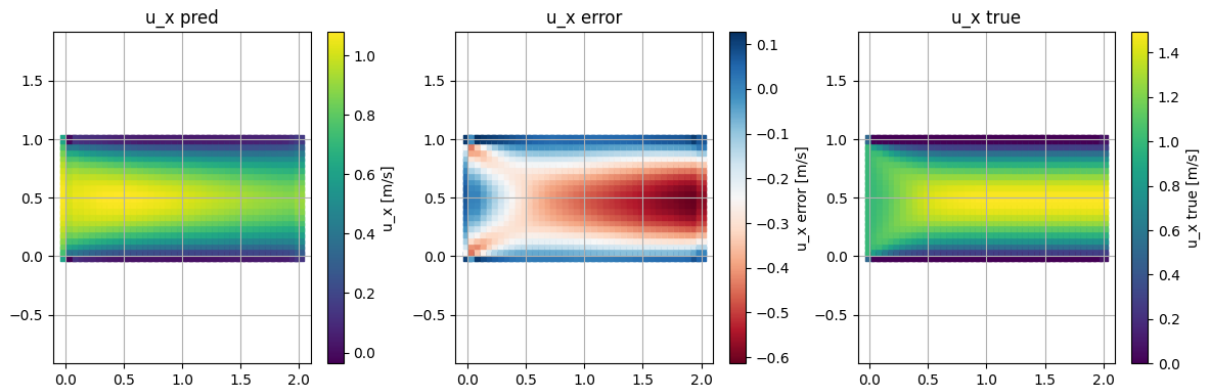


Figure 4.18: Predicted, L2 error and OpenFOAM results for the velocity component along the x-direction (u_x).

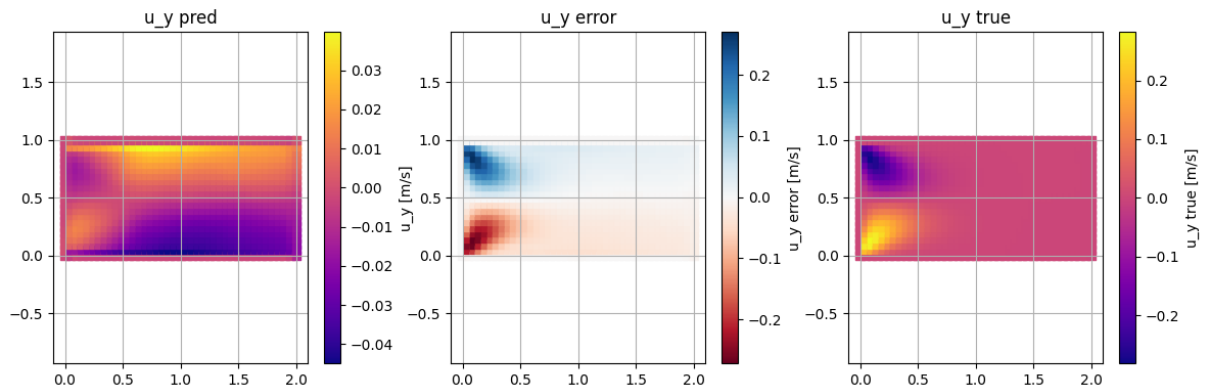


Figure 4.19: Predicted, L2 error and OpenFOAM results for the velocity component along the y-direction (u_y).

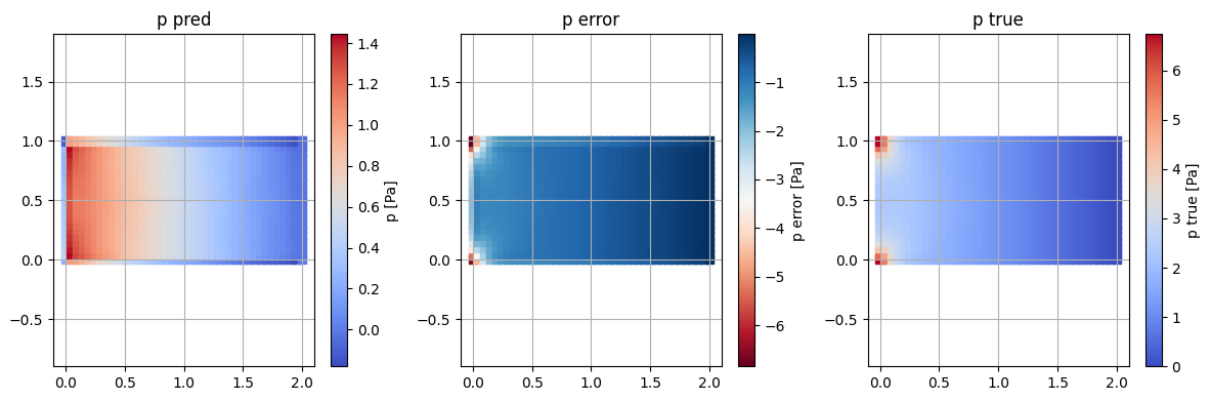


Figure 4.20: Predicted, L2 error and OpenFOAM results for the pressure (p). The pressure field exhibits oscillations and offset drift due to the indeterminacy of the gauge pressure.

To mitigate this, a weak supervision term was introduced on the pressure values at the domain boundaries. Without anchoring, the pressure field showed large oscillations and offset drift due to the gauge indeterminacy of the incompressible Navier–Stokes equations (See Fig. 4.20). These inconsistencies propagated to the velocity prediction through the coupling terms of the momentum equations, slightly deteriorating the overall accuracy (See Fig. 4.18 and 4.22).

Anchoring the pressure on the inflow and outlet regions effectively removed the constant-offset ambiguity, stabilized the pressure field (See Fig. 4.23) and also the velocity fields (See Fig. 4.21 and 4.22). The resulting predictions displayed a consistent linear pressure gradient and smooth contours across the domain.

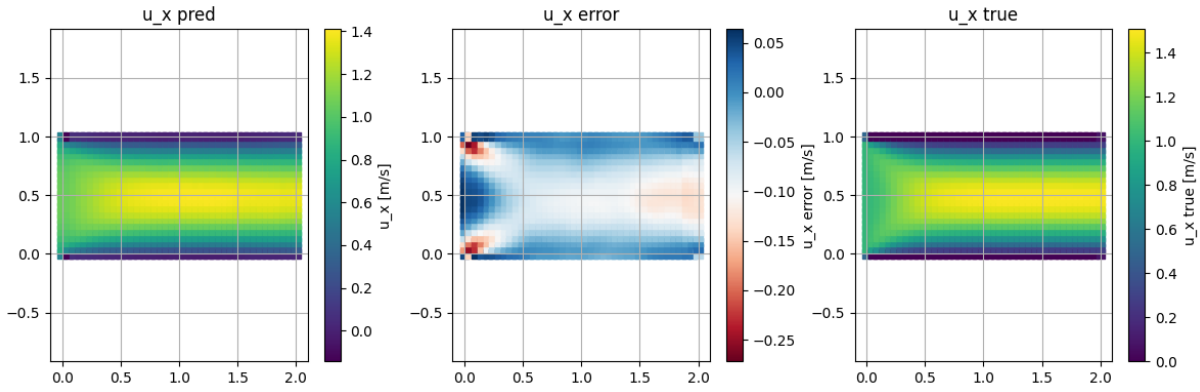


Figure 4.21: Fully unsupervised case with boundary pressure anchoring. Velocity along x-axes.

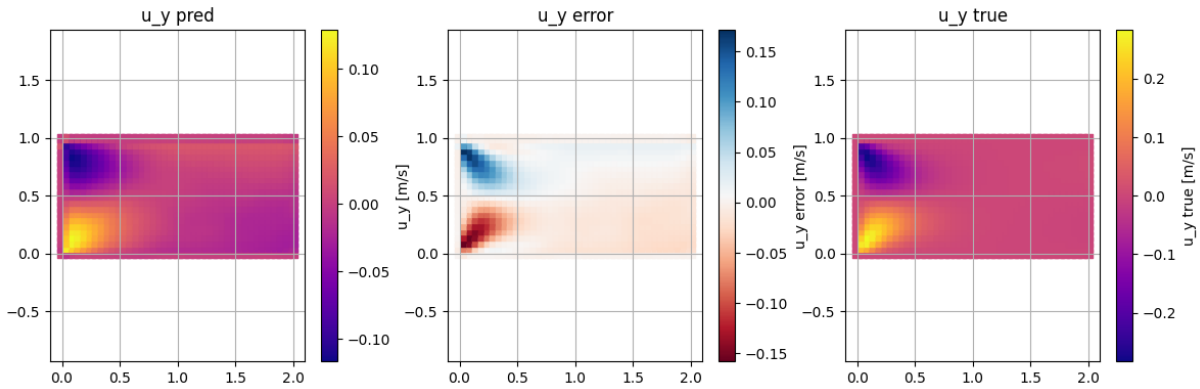


Figure 4.22: Fully unsupervised case with boundary pressure anchoring. Velocity along y-axes.

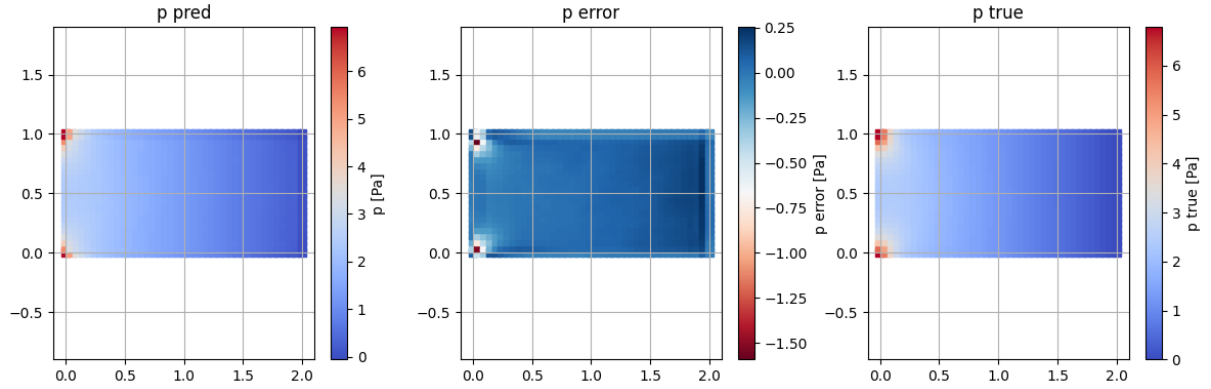


Figure 4.23: Fully unsupervised case with boundary pressure anchoring. Prediction of the pressure.

Quantity	No pressure supervision	With boundary anchoring
Velocity (u_x)	2.3×10^{-1}	7.0×10^{-2}
Velocity (u_y)	6.0×10^{-2}	4.0×10^{-2}
Pressure (p)	7.0×10^{-1}	2.3×10^{-1}

Table 4.5: Comparison of mean L_2 errors with and without boundary pressure supervision in the fully unsupervised case ($U_{\text{in}} = 1.0$ m/s).

The comparison in Table 4.5 shows that pressure anchoring improves not only the pressure prediction itself, but also the accuracy of both velocity components. Although the improvement is moderate, it is consistent across all quantities, confirming the global coupling role of pressure within the Navier–Stokes system: even small pressure drifts can perturb the momentum balance and progressively affect the velocity field. From a methodological standpoint, the usefulness of introducing boundary pressure supervision depends on the objective of the study. If the aim is to maintain a strictly physics-only unsupervised formulation, the additional constraint represents a departure from the pure residual-based setting. On the other hand, when numerical stability, pressure identifiability, and overall predictive accuracy are of primary interest, weak supervision at the boundaries provides an effective and physically consistent regularization. For this reason, the pressure anchoring term can be interpreted as a minimal supervisory condition that resolves the gauge indeterminacy of the incompressible Navier–Stokes equations without modifying their governing form.

Overall, the fully unsupervised model demonstrates that physics-informed training can reconstruct realistic laminar channel flows even without labeled data, provided that the

pressure field is properly anchored. Compared to the purely unsupervised configuration, the introduction of pressure anchoring leads to a substantial reduction of the pressure error and to a noticeable improvement in velocity predictions. Although the resulting accuracy remains lower than in the supervised and partially unsupervised cases, the errors are reduced to the same order of magnitude (Table 4.6).

Quantity	Supervised	Part. Unsup.	Unsup.	Unsup. + PA
Velocity (u_x)	2.64×10^{-2}	2.71×10^{-2}	2.3×10^{-1}	7.0×10^{-2}
Velocity (u_y)	1.05×10^{-2}	1.09×10^{-2}	6.0×10^{-2}	4.0×10^{-2}
Pressure (p)	1.86×10^{-1}	1.34×10^{-1}	7.0×10^{-1}	2.3×10^{-1}

Table 4.6: Comparison of mean L_2 errors for velocity and pressure fields under different supervision regimes.

5 | Channel with an obstacle

After validating the proposed methodology on a simple laminar flow in a channel, a more challenging configuration is investigated to assess both the generalization capability of the MeshGraphNet model and the robustness of the physics-informed loss formulation. The new test case consists of a two-dimensional channel containing an obstacle, which introduces additional physical phenomena such as flow separation, wake formation, and, depending on the Reynolds number, unsteady vortex shedding or fully developed turbulent behavior.

Compared to the laminar benchmark, this configuration represents a significant step forward in complexity. The considered flow regimes span from steady laminar conditions to transitional and turbulent dynamics, thereby testing the ability of the network to learn and reproduce increasingly rich spatio-temporal structures. At the same time, the presence of an obstacle and unsteady wake dynamics makes long-horizon autoregressive rollout substantially more demanding, providing a stringent test for the stability of the learned surrogate.

In this setting, model performance is not evaluated solely in terms of pointwise accuracy, but also in terms of qualitative and dynamical consistency of the predicted flow fields. In particular, we assess whether the surrogate preserves the dominant large-scale structures of the wake, such as recirculation regions and coherent vortex-shedding patterns, rather than merely matching local values. We also examine whether the model avoids non-physical behaviors during autoregressive temporal rollout, including spurious oscillations, artificial damping or amplification of vortical structures, or progressive pressure drift. Finally, we assess whether the model maintains stable behavior over long temporal horizons, preventing the progressive accumulation of errors and the drift toward physically inconsistent solutions, even under reduced supervision. To address these challenges, additional training strategies are explored, including multi-phase optimization with optimizer resets and explicit temporal rollout during training (see Section 5.2).

In particular, this chapter provides a systematic investigation of flow configurations involving solid obstacles, all generated through high-fidelity CFD simulations using the

OpenFOAM framework (see Section 5.1). These simulations serve a dual purpose: on the one hand, they provide supervised data used to compute data-driven loss terms during training; on the other hand, they represent the reference physical ground truth against which the predictive accuracy and stability of the learned surrogate are evaluated.

The analysis begins with a steady laminar square obstacle flow at low Reynolds number (Section 5.3), which serves as a controlled benchmark to assess baseline accuracy under relatively simple flow conditions. The study then extends to a transitional circular obstacle flow at moderate Reynolds number (Section 5.4), characterized by unsteady wake dynamics and vortex shedding.

The generalization capability of the proposed model is further examined in Section 5.5, where training and evaluation are performed on a significantly larger and more heterogeneous dataset, spanning multiple geometries and inlet conditions. Subsequently, a transitional airfoil obstacle configuration at higher Reynolds number is considered (Section 5.6), introducing increased geometric complexity and richer flow features.

Finally, the most challenging regime is addressed in Section 5.7, which focuses on a turbulent square obstacle flow at very high Reynolds number. This case provides a stringent test for the robustness of the physics-informed formulation and for the long-term stability of autoregressive rollouts under highly chaotic dynamics, ensuring that the predicted solution remains consistent with the reference flow behavior even over long time horizons.

5.1. OpenFOAM setup and boundary conditions

The flow configurations analyzed in this chapter are deliberately organized to follow a progressive increase in physical and geometrical complexity. All numerical datasets are generated using OpenFOAM, whose numerical setup and boundary conditions have been described in detail in Chapter 2 (see Section 2.2 and 2.3).

When the obstacle geometry is simple and the flow remains steady, as in the low-Reynolds-number case ($Re = 20$), a structured mesh is sufficient. The square obstacle and the absence of unsteady dynamics allow the use of a regular grid aligned with the main flow direction (Fig. 2.5), ensuring uniform node spacing and minimal numerical diffusion. This configuration provides a controlled reference case for assessing model accuracy under well-resolved and stationary conditions.

As geometric curvature and unsteady wake dynamics become relevant, structured discretizations are no longer adequate. For the transitional regime at $Re = 200$, the circular obstacle introduces curved boundaries and periodic vortex shedding. An unstructured

triangular mesh generated with Gmsh is therefore adopted (Fig. 2.6), allowing accurate geometric representation while preserving locally isotropic node distributions. This property is particularly important for the construction of radial basis function finite-difference (RBF–FD) operators, whose accuracy depends on the spatial arrangement of neighboring nodes (see Section 3.4 for details).

At higher Reynolds number ($Re = 2000$), the airfoil configuration further increases geometric complexity and introduces strong near-wall gradients. In this case, mesh refinement is driven primarily by physical considerations: a locally refined unstructured triangular grid is employed to resolve boundary-layer dynamics and sharp velocity gradients near the airfoil surface (Fig. 2.7). This refinement is essential both for numerical stability of the CFD simulations and for the accurate evaluation of the differential operators used during training.

In the fully turbulent regime ($Re = 5 \times 10^5$), the focus shifts from geometric fidelity to the ability of the model to capture the dominant large-scale wake dynamics generated by strong flow separation. The square obstacle is reintroduced to promote strong flow separation and vorticity production, while a structured mesh is adopted to provide a regular discretization of the canonical geometry. This choice facilitates the analysis of long-horizon prediction stability under highly chaotic dynamics.

Across all configurations, the characteristic obstacle dimensions are kept fixed to ensure comparability: the square obstacle has side length $a = 1$ m, while the circular obstacle has diameter $D = 1$ m.

Flow regimes and solvers

Different flow regimes are investigated by varying the Reynolds number,

$$Re = \frac{U_{\text{in}} D}{\nu},$$

while keeping the inlet velocity fixed at $U_{\text{in}} = 1$ m/s. For the square and circular obstacles, the characteristic length is set to $a = D = 1$ m, whereas the airfoil configuration has a chord length of $c = 1$ m and an inclination angle of 7.13° , chosen to promote flow separation and vortex formation in the wake. As a result, the Reynolds number is controlled exclusively through variations of the kinematic viscosity ν .

Depending on the flow regime, different OpenFOAM solvers are employed in order to maintain physical consistency with the underlying modeling assumptions. For laminar and transitional regimes, the incompressible transient solver icoFoam is adopted. This

solver directly integrates the incompressible Navier-Stokes equations without introducing any turbulence closure, making it suitable for low- and moderate Reynolds number flows in which all dynamically relevant scales are resolved.

At $Re = 20$, the flow remains steady and laminar, with a stable recirculation region forming downstream of the obstacle. In this regime, icoFoam provides an accurate reference solution without requiring additional modeling assumptions. As the Reynolds number increases to $Re = 200$ and $Re = 2000$, the wake becomes unsteady and exhibits periodic or irregular vortex shedding. These transitional regimes are still simulated using icoFoam, allowing the unsteady dynamics to emerge naturally from the resolved Navier-Stokes equations.

For the highest Reynolds number considered ($Re = 5 \times 10^5$), the flow enters a fully turbulent regime characterized by broadband temporal fluctuations and a wide separation of scales. In this case, resolving all turbulent structures directly would be computationally prohibitive. The simulations therefore rely on the pimpleFoam solver coupled with the $k-\omega$ SST turbulence model, which introduces a Reynolds-averaged closure to account for the unresolved turbulent stresses.

For all configurations, the initial time step is set to $\Delta t = 0.001$ s, and flow fields are stored every 0.1 s to construct a temporally regular dataset suitable for training and evaluation.

Simulations

Figures 5.1, 5.2, 5.3, and 5.4 show representative snapshots of the flow fields obtained from the OpenFOAM simulations at different Reynolds numbers. For each configuration, two time instants are reported in order to better highlight the temporal evolution of the flow structures. The transition from steady laminar recirculation to unsteady vortex shedding and finally to fully developed turbulence can be clearly observed.

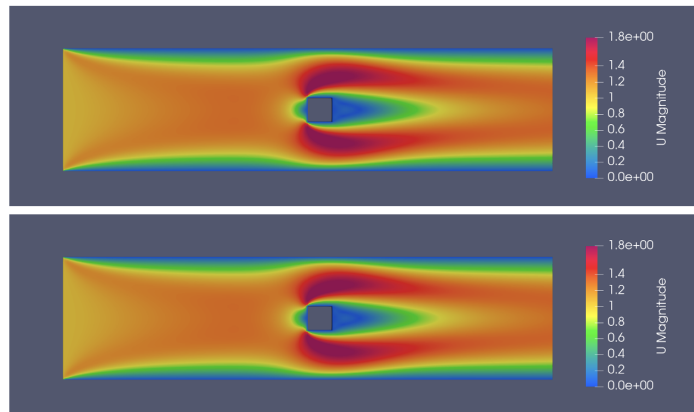


Figure 5.1: $Re = 20$ — Steady laminar wake with a stable recirculation zone.

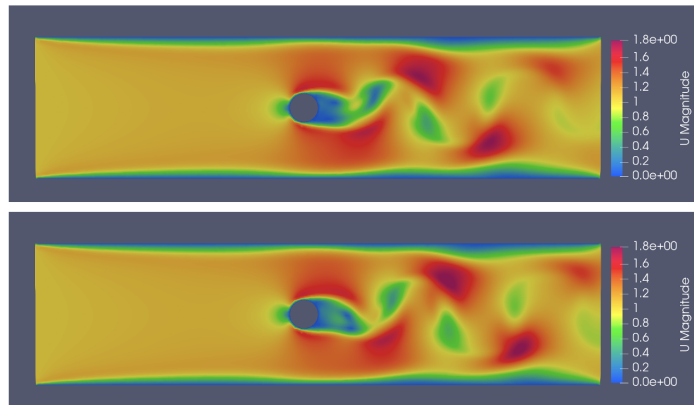


Figure 5.2: $Re = 200$ — Transitional regime with periodic vortex shedding.

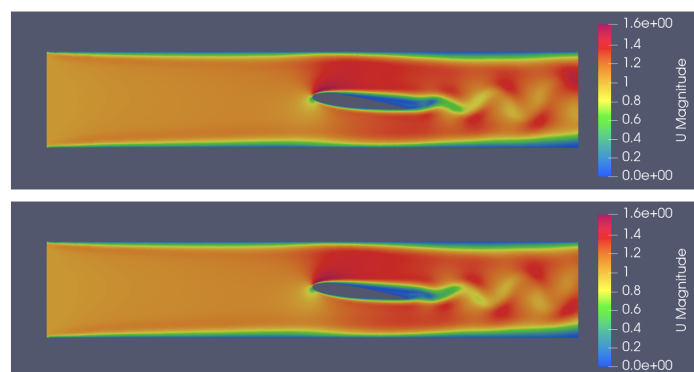


Figure 5.3: $Re = 2000$ -Transitional regime with irregular vortex shedding.

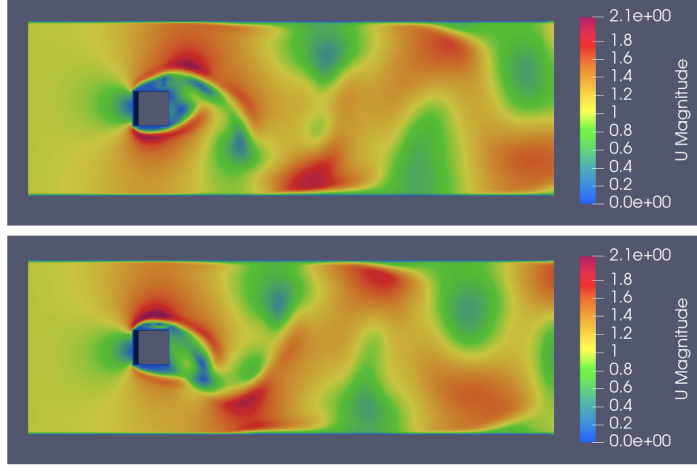


Figure 5.4: $Re = 5 \times 10^5$ — Fully turbulent wake.

5.2. Training methodology and learning strategies

The same MeshGraphNet architecture described in the previous chapters was adopted for all the obstacle-channel experiments. The model used the Adam optimizer and cosine learning-rate annealing, with gradient clipping and progressive temporal rollout (see Table 4.1). A new strategy was introduced to improve convergence stability and long-term prediction accuracy: a multi-phase training schedule.

5.2.1. Multi-phase training

The multi-phase procedure consists of dividing the entire training process into consecutive phases, each characterized by different relative weights ($w_{\text{vel}}, w_p, w_{\text{div}}, w_{\text{res}}$) of the supervised and physics-informed loss components. In the early phases, higher weights are assigned to the supervised terms, promoting accurate learning of boundary conditions and reference flow behavior. As training progresses, the supervised weights are gradually reduced, while the weights associated with the physics-informed losses are increased, encouraging the network to enforce the underlying physical constraints.

At the end of each phase, both the optimizer and learning-rate scheduler are restart, while the model parameters are preserved. This allows the network to progressively rebalance the contribution of the physical constraints without getting trapped in local minima [60].

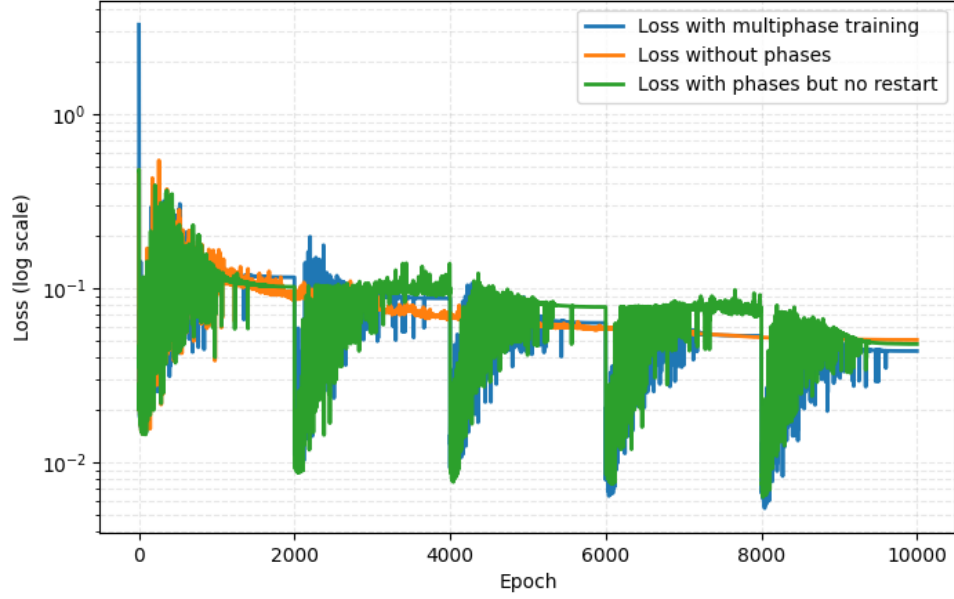


Figure 5.5: Comparison of the evolution of the velocity training loss across different phases of the multi-phase schedule, across one single phase and across the multi-phase without the restart.

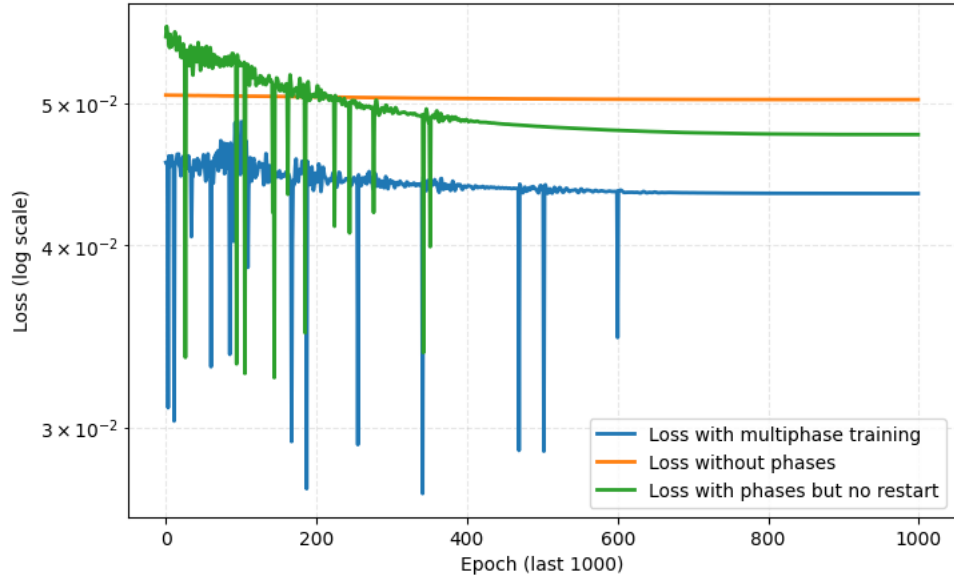


Figure 5.6: Comparison of the evolution of the velocity training loss across different phases of the multi-phase schedule, across one single phase and across the multi-phase without the restart, for the last 1000 epochs.

Figures 5.5 and 5.6 refer to the case analyzed in Section 5.4, for which a four-phase loss-weighting schedule was employed:

Phase 1: $w_{\text{vel}} = w_p = 1.0$, $w_{\text{div}} = w_{\text{res}} = 0.2$,

Phase 2: $w_{\text{vel}} = w_p = 0.8$, $w_{\text{div}} = w_{\text{res}} = 0.5$,

Phase 3: $w_{\text{vel}} = w_p = 0.7$, $w_{\text{div}} = w_{\text{res}} = 0.7$,

Phase 4: $w_{\text{vel}} = w_p = 0.5$, $w_{\text{div}} = w_{\text{res}} = 1.0$.

The purpose of presenting this example is not to provide a case-specific performance analysis, but rather to illustrate the effectiveness of the proposed multi-phase training strategy. This effectiveness has been verified across the other flow configurations considered in this work, for which the same training strategy is consistently adopted in the following sections.

The comparison between the multi-phase training, the single-phase setup and the multi-phase variant without scheduler and optimizer resets clearly shows that the full multi-phase strategy is the most effective in reducing the loss. While the alternative configurations tend to stagnate after an initial decay, the multi-phase approach continues to improve throughout training and achieves significantly lower error levels.

Each phase comprises approximately 2000 epochs, with progressively increasing weights assigned to the divergence and residual terms. This curriculum markedly enhances the physical consistency of the predicted fields and yields the lowest error, particularly in the wake region downstream of the obstacle.

The downward spikes observed in the curve are related to the temporal rollout strategy employed during training. At the beginning of each training phase, the rollout length is increased, which changes the effective prediction horizon and introduces transient variations in the error dynamics. Moreover, the reported quantity is not the total training loss, but the mean squared error on the velocity along the x-direction u , which acts as a trajectory-level accuracy metric rather than an optimized objective. As a result, this metric may exhibit non-monotonic behavior, including sudden decreases, without indicating instability in the overall training process.

To assess the robustness of the observed performance improvements, an additional sensitivity analysis with respect to the random initialization was conducted, to exclude the possibility that the observed performance differences arise from a favorable random initialization.

In particular, multiple independent training runs were performed by varying the random seed affecting the network weight initialization, while keeping all other hyperparameters, datasets, and training configurations fixed. For each run, the final mean squared error

on the velocity along the x-direction was recorded using the same evaluation protocol adopted in the main analysis.

The resulting error values for the three training strategies considered are reported below:

- **Multi-phase training (with optimizer and scheduler restart):**

$$\ell_1 = [0.0412, 0.0399, 0.0371, 0.0443, 0.0530, 0.0352, 0.0392, 0.0400, 0.0422, 0.0414]$$

- **Single-phase training:**

$$\ell_1 = [0.0473, 0.0488, 0.0472, 0.0521, 0.0532, 0.0536, 0.0484, 0.0532, 0.0463, 0.0430]$$

- **Multi-phase training without optimizer and scheduler restart:**

$$\ell_1 = [0.0501, 0.0480, 0.0522, 0.0472, 0.0456, 0.0488, 0.0447, 0.0379, 0.0435, 0.0580]$$

For each configuration, the mean and variance of the mean squared error across different random seeds were computed. The results are summarized in Table 5.1.

Table 5.1: Mean and variance of the L1 velocity error across different random seeds.

Training strategy	Mean	Variance
Multi-phase (with restart)	4.14×10^{-2}	2.10×10^{-5}
Single-phase	4.93×10^{-2}	1.15×10^{-5}
Multi-phase (no restart)	4.76×10^{-2}	2.62×10^{-5}

The analysis shows that the multi-phase training strategy with optimizer and scheduler restart consistently outperforms both the single-phase approach and the multi-phase variant without restart across all tested random seeds. Although some variability is observed, as expected in stochastic optimization, the mean performance of the full multi-phase method remains systematically lower, and its variance does not indicate unstable or erratic behavior. In contrast, the single-phase strategy exhibits a higher mean error, while the multi-phase configuration without restart displays both a higher mean and a larger variance, suggesting reduced robustness and a higher sensitivity to initialization. These results confirm that the improved accuracy observed for the multi-phase strategy in the main text is a robust property of the training methodology and not an artifact of a particular random seed. Consequently, the conclusions regarding the superiority of the multi-phase training approach are supported by statistically consistent evidence.

5.2.2. Temporal rollout

In addition to the multi-phase optimization, a temporal rollout scheme with a growing prediction horizon was employed. The model is trained autoregressively: at each timestep, the network prediction is fed back as input for the next one, producing a full rollout sequence. During training, the total rollout time is not fixed from the beginning. Instead, the rollout length is increased progressively every few epochs, starting from a single timestep and gradually extending to the full temporal horizon of the simulation, as illustrated in Fig. 5.7.

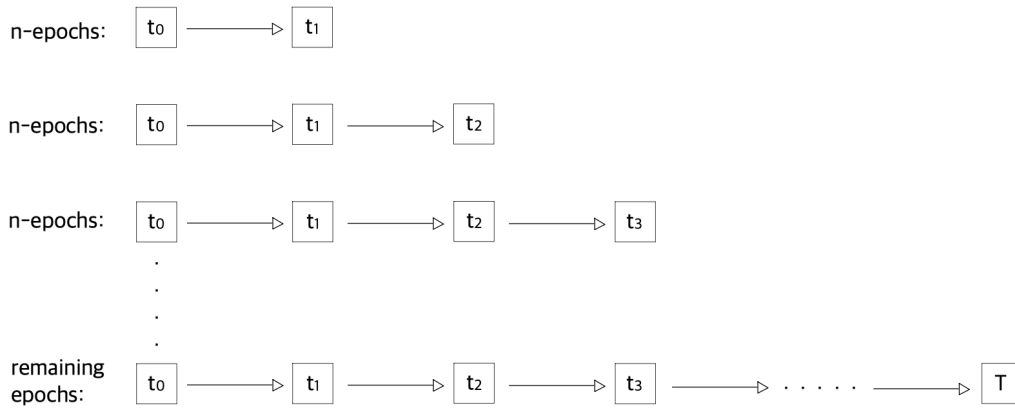


Figure 5.7: Schematic illustration of the temporal rollout strategy. The model is first trained using short autoregressive rollouts (e.g., one timestep), and the prediction horizon is progressively increased after a fixed number of epochs, until the full rollout length is reached.

This curriculum over time allows the model to first learn short, stable transitions and then progressively handle longer and more complex temporal dependencies. At the beginning of each new temporal stage (i.e., after increasing the rollout horizon), training continues from the weights obtained in the previous stage, ensuring continuity and stability. This strategy effectively mitigates the accumulation of autoregressive errors and improves the temporal generalization ability of the network, particularly in periodic and unsteady regimes. The combination of multi-phase optimization and curriculum-based temporal rollout provides a robust and flexible framework for training physics-informed graph neural networks on complex fluid-dynamics problems.

Training loop structure. The combined temporal-curriculum and multi-phase loss schedule used in this work can be summarized as:

Multi-phase temporal training scheme

```

for each loss phase in loss_phases:
    Reset optimizer and scheduler
    Set epochs for the current phase
    Set weights ( $w_{\text{vel}}, w_p, w_{\text{div}}, w_{\text{res}}$ ) for this loss phase
    for epoch = 1 to  $N_{\text{epochs}}$ :
        Set max_rollout for the current epoch
        for each batch in training set:
            Initialize features and previous state (normalized)
            for  $t = 1$  to max_rollout:
                Predict ( $\Delta u, \Delta v, p$ ) at step  $t$  (autoregressive)
                Compute supervised terms (masked) and physics terms (NS residuals & div)
                Accumulate total loss  $\mathcal{L}$ 
            Backpropagate  $\mathcal{L}$ , clip gradients, optimizer step
        Save best model for this loss phase
    Load best checkpoint to initialize the next temporal phase

```

5.3. Laminar and steady state square obstacle flow ($Re = 20$)

The first test case focuses on the steady laminar regime ($Re = 20$), characterized by a stable recirculation region downstream of the obstacle and a stationary pressure field. This configuration provides a controlled environment to evaluate the effectiveness of the physics-informed and hybrid training strategies in reconstructing a steady flow.

Domain and mesh

The computational domain consists of a rectangular channel ($20 \text{ m} \times 5 \text{ m}$) containing a square obstacle of side 1 m , placed at $(x, y) \in [10, 11] \times [2, 3]$. The same mesh used in the OpenFOAM simulation was employed for the graph representation, but subsampled by retaining one node every eight to reduce computational cost while preserving the geometric fidelity. The mesh is locally refined around the obstacle to accurately capture the recirculation zone (Fig. 5.8).

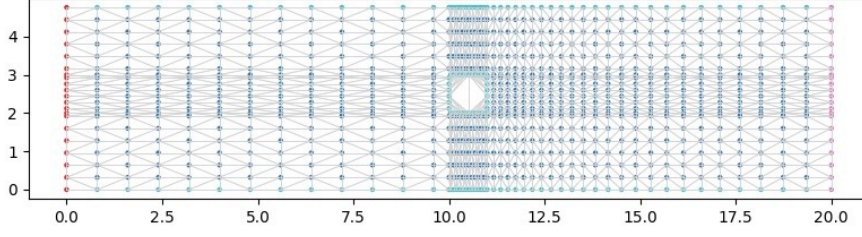


Figure 5.8: Structured mesh used for the $Re = 20$ laminar obstacle flow. The nodes are colored according to the categorical labels, which assigns a distinct node type to each region of the computational domain.

Model inputs and outputs

Since the flow is steady, temporal information was excluded from the input features. Each node is represented by:

$$\mathbf{x}_i = [x_i, y_i, \text{one_hot_type_vector}_i],$$

where (x_i, y_i) denote coordinates of the node, and the one-hot type encodes the node category (inlet, outlet, wall, obstacle, or interior). All physical quantities were normalized per dataset using their mean and standard deviation, with denormalization applied only during post-processing and physics-based loss evaluation.

The MeshGraphNet model predicts the variation of the velocity components and the absolute pressure:

$$\hat{\mathbf{y}}_i = [\Delta u_{x,i}, \Delta u_{y,i}, p_i].$$

Training configurations

Two training configurations were explored:

- **Partially unsupervised:** only 10% of the domain is supervised, mostly near the obstacle and walls. A mask of 0.2 m around the obstacle defines a narrow supervised band, while the rest of the flow field (about 90%) is trained via physics-informed losses (see Fig. 5.9).

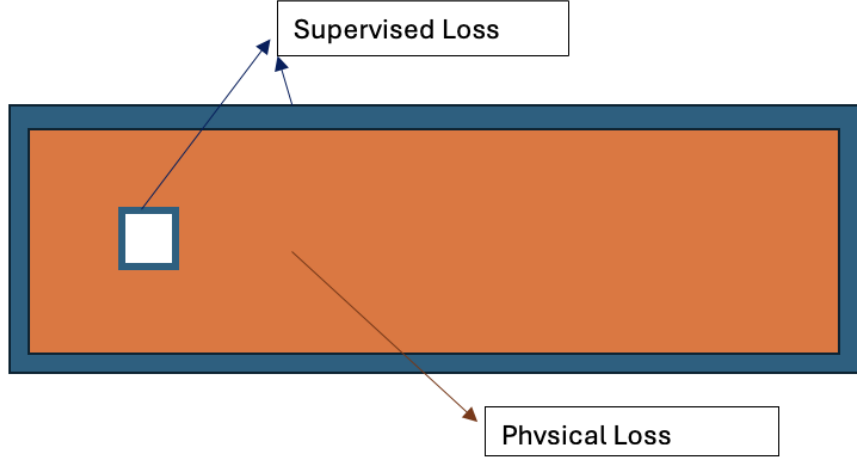


Figure 5.9: Domain subdivision for the partially unsupervised case. Supervised regions (blue) surround the obstacle and boundaries, while the interior flow is driven by physical constraints.

- **Fully unsupervised:** supervision is applied only to the inlet and wall nodes, enforcing prescribed velocity boundary conditions. The interior field is learned exclusively from the physical losses (orange).

The total loss is given by:

$$\mathcal{L}_{\text{tot}} = w_{\text{vel}} \mathcal{L}_{\text{vel}} + w_p \mathcal{L}_p + w_{\text{div}} \mathcal{L}_{\text{div}} + w_{\text{res}} \mathcal{L}_{\text{res}}.$$

However, the two configurations differ in how the loss weights were handled:

Partially unsupervised case. A multi-phase schedule was adopted to progressively shift the training from data fitting to physics-informed refinement. At each phase, the optimizer and learning-rate scheduler were reset, resulting in a staircase-shaped loss evolution and improved convergence. The training weights were:

Phase 1: $w_{\text{vel}} = w_p = 1.0$, $w_{\text{div}} = w_{\text{res}} = 0.08$,

Phase 2: $w_{\text{vel}} = w_p = 1.0$, $w_{\text{div}} = w_{\text{res}} = 0.10$,

Phase 3: $w_{\text{vel}} = w_p = 0.8$, $w_{\text{div}} = w_{\text{res}} = 0.30$.

Fully unsupervised case. Only one phase was required, as the steady-state regime naturally stabilized the training without further curriculum or phase transitions. Weights

were fixed to:

$$w_{\text{vel}} = 1.0, \quad w_{p,\text{outlet}} = 0.1, \quad w_{\text{div}} = w_{\text{res}} = 0.01.$$

Results and discussion

Both configurations successfully reconstructed the laminar flow pattern around the obstacle. The partially unsupervised model closely matched the OpenFOAM reference, correctly reproducing the recirculation bubble and the pressure recovery downstream of the obstacle (Fig. 5.10, Fig. 5.11, Fig. 5.12).

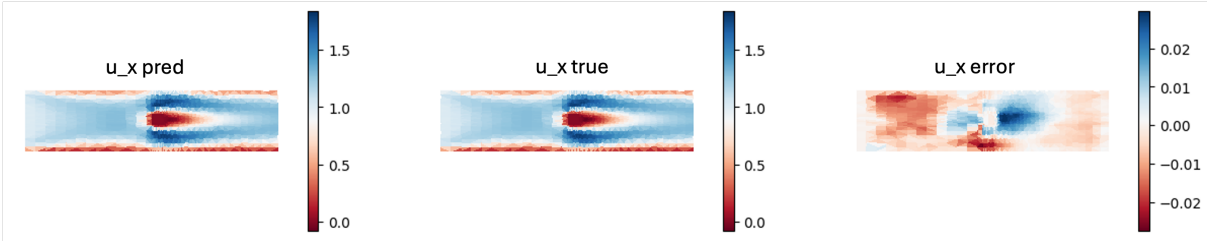


Figure 5.10: Predicted velocity along x-direction field (u_x) compared with the OpenFOAM reference.

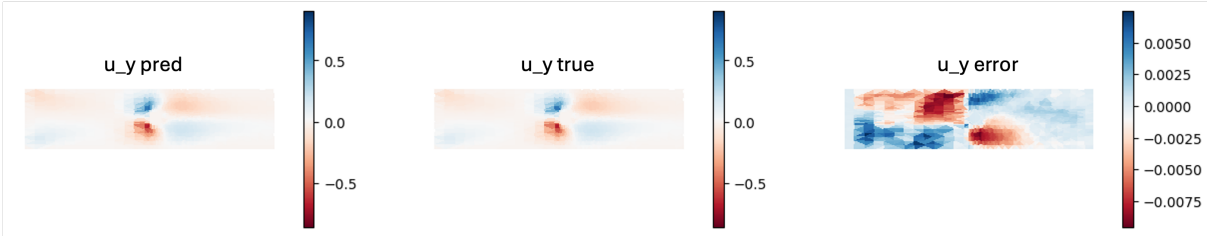


Figure 5.11: Predicted velocity along y-direction field (u_y) compared with the OpenFOAM reference.

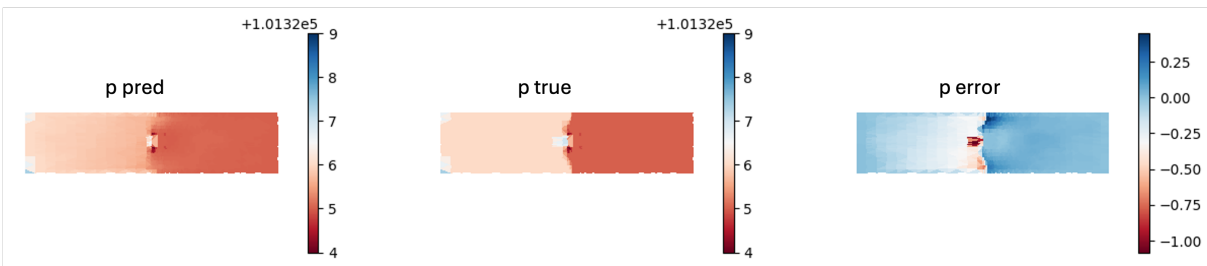


Figure 5.12: Predicted pressure field (p) compared with the OpenFOAM reference.

The fully unsupervised configuration, despite relying solely on boundary supervision, yielded physically coherent results (Fig. 5.13, Fig. 5.14, Fig. 5.15). A critical difficulty, however, concerns the pressure field. As already highlighted in Section 4.3.4, the Navier–Stokes equations determine the pressure field only up to an arbitrary additive constant. Consequently, in a fully unsupervised setting, the model has no intrinsic mechanism to fix a consistent pressure gauge, which may lead to arbitrary global shifts of the predicted pressure field.

To mitigate this, the pressure is anchored at the outlet. The outlet nodes are identified by `batch.node_type = 2`, and at each iteration the mean predicted pressure over these nodes is used as a reference,

$$p_{\text{gauge}} = \text{mean}(p_{\text{outlet}}).$$

The pressure field is then expressed relative to this reference,

$$p_{\text{rel}} = p - p_{\text{gauge}},$$

and a penalty is applied to enforce consistency at the outlet,

$$\mathcal{L}_{p,\text{outlet}} = \text{mean}(p_{\text{rel,outlet}}^2).$$

The same gauge correction is applied when computing the Navier–Stokes residuals, i.e.

$$p_t \leftarrow p_t - p_{\text{gauge}}, \quad p_{t+1} \leftarrow p_{t+1} - p_{\text{gauge}}.$$

where p_t is the pressure at the timestep t and p_{t+1} is the pressure at the timestep $t + 1$. This removes the gauge ambiguity and stabilizes the pressure prediction, though its accuracy remains slightly inferior to the partially supervised case.

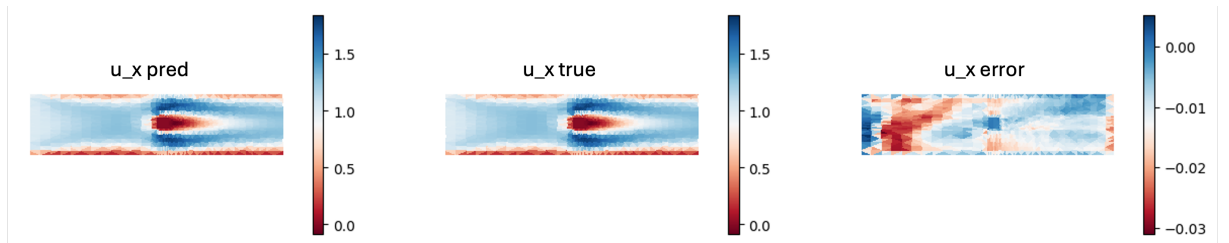


Figure 5.13: Predicted velocity along x-direction field (u_x) compared with the OpenFOAM reference.

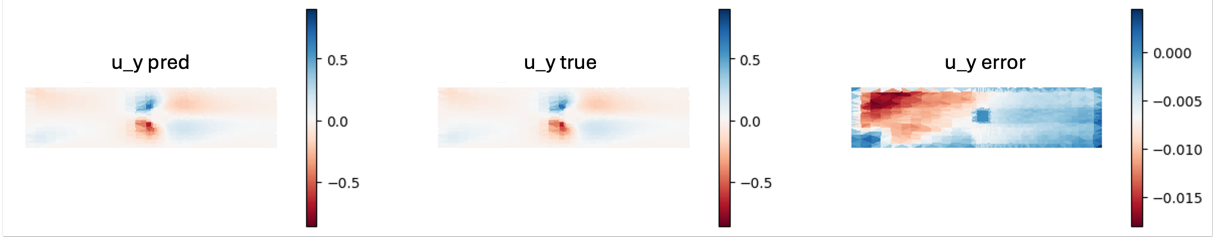


Figure 5.14: Predicted velocity along x-direction field (u_y) compared with the OpenFOAM reference.

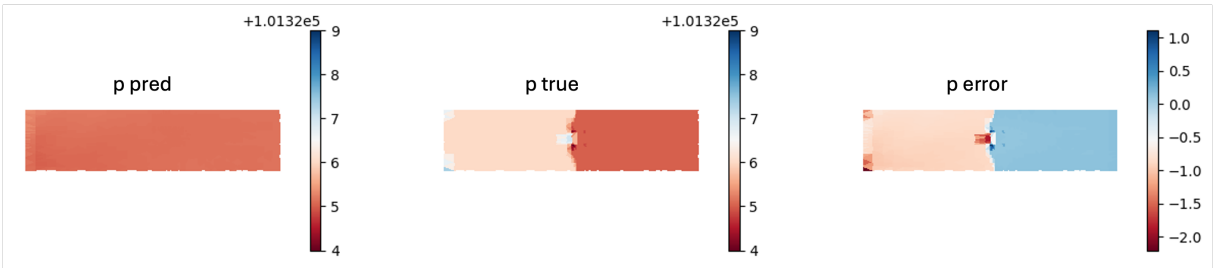


Figure 5.15: Predicted pressure field (p) compared with the OpenFOAM reference.

Quantitatively, the two models achieve comparable accuracy on the velocity components, whereas the pressure error is markedly larger in the fully unsupervised case, as shown in Table 5.2. The increase in velocity error is relatively modest, indicating that the overall structure of the velocity field is preserved even without interior supervision. By contrast, the pressure error grows by approximately a factor of three, reflecting the sensitivity of the pressure field to gauge indeterminacy in the absence of explicit anchoring.

Quantity	Symbol	Partially unsup.	Fully unsup.
Velocity (x-direction)	u_x	1.114×10^{-2}	1.321×10^{-2}
Velocity (y-direction)	u_y	3.465×10^{-3}	6.260×10^{-3}
Pressure	p	1.833×10^{-1}	5.622×10^{-1}

Table 5.2: Mean L_2 errors for the laminar obstacle flow ($Re = 20$).

Overall, the $Re = 20$ experiments demonstrate that the MeshGraphNet framework can accurately reconstruct steady laminar flows even under predominantly unsupervised training. While the absence of pressure supervision introduces a gauge offset, the velocity predictions remain precise and physically consistent, confirming that the physics-informed loss alone can enforce the steady Navier–Stokes balance across the domain.

5.4. Transitional circular obstacle flow ($Re = 200$)

Having validated the model on the steady laminar configuration, we now move to a more challenging scenario. The second case examines a transitional regime in which the flow develops periodic vortex shedding downstream of the obstacle. At $Re = 200$, the dynamics become unsteady and quasi-periodic, marking a substantial increase in complexity compared to the steady-state case. This configuration allows us to assess the capability of the model to reproduce time-dependent phenomena and to evaluate the effectiveness of the temporal curriculum and multi-phase training strategies introduced in this work.

Domain, mesh and input features

The computational domain and obstacle geometry are identical to the previous case, but the obstacle now is a circle with diameter $D = 1m$. The flow regime produces an unsteady wake with alternating vortices (von Kármán vortex street). A full vortex-shedding period corresponds approximately to $T = 19$ timesteps, each of $\Delta t = 0.2$ s.

To improve the accuracy of the differential operators in regions with complex gradients, an unstructured triangular mesh was employed (Fig. 5.16). This layout provides better local isotropy, meaning that the stencil points are more uniformly distributed in all directions around each node, and allows the RBF-FD operators to achieve higher accuracy near the obstacle and in the wake region, where regular grids tend to introduce directional bias (see Section 3.4).

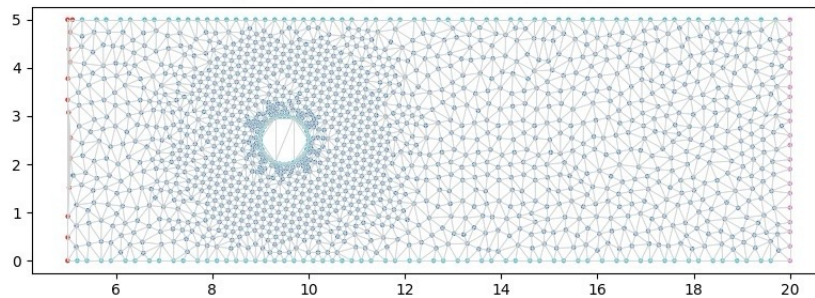


Figure 5.16: Unstructured mesh used for the $Re = 200$ obstacle flow. The nodes are colored according to the categorical labels, which assigns a distinct node type to each region of the computational domain.

Node features now also incorporate the velocity field from the previous time step. Specif-

ically, each node i is associated with a feature vector

$$\mathbf{x}_i = [u_{x,i}^{t-1}, u_{y,i}^{t-1}, x_i, y_i, \text{one_hot_type_vector}_i],$$

where $u_{x,i}^{t-1}$ and $u_{y,i}^{t-1}$ denote the velocity components at node i at the previous time step, (x_i, y_i) are the spatial coordinates of the node, and $\text{one_hot_type_vector}_i$ is a one-hot encoded vector identifying the node category.

Model outputs are defined as

$$\hat{\mathbf{y}}_i = [\Delta u_{x,i}^t, \Delta u_{y,i}^t, p_i^t],$$

where $\Delta u_{x,i}^t$ and $\Delta u_{y,i}^t$ denote the predicted velocity increments at node i between consecutive time steps, while p_i^t is the predicted pressure at the current time step.

All quantities were normalized using dataset-specific mean and standard deviation.

Training strategy and temporal curriculum

The training followed a partially unsupervised scheme, with approximately 90% of the domain governed by physics-informed losses and 10% supervised near the obstacle and boundaries (See Fig. 5.17).

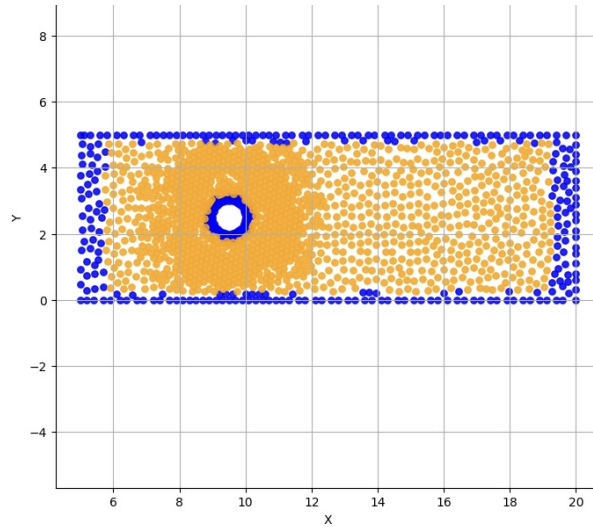


Figure 5.17: The subdivision of the nodes in supervised and unsupervised mask. The blue one is the supervised mask and the orange one is the unsupervised mask.

A major challenge in unsteady flows is the accumulation of temporal error during multi-step rollouts: each prediction becomes the input for the next, and small discrepancies

can rapidly amplify over time. To mitigate this issue, a rollout curriculum learning approach was used, where the prediction horizon (rollout length) was progressively increased through the epochs.

Temporal curriculum. Two temporal configurations were tested:

- **Single temporal rollout:** 30 time steps (6 s), used both for training and evaluation, with a progressive increase in the prediction horizon during training.
- **Sequential temporal rollouts in the periodic regime:** two full vortex-shedding periods ($2T = 38$ timesteps). The model was first trained on the initial 19 timesteps, and then further trained on the next 19 timesteps starting from the previously saved model, following the same loss schedule. The final test was performed over three periods ($3T = 57$ timesteps), including one unseen period.

Loss weighting. In all cases, a four-phases loss schedule was used to progressively emphasize physical consistency:

Phase 1: $w_{\text{vel}} = w_p = 1.0$, $w_{\text{div}} = w_{\text{res}} = 0.2$,

Phase 2: $w_{\text{vel}} = w_p = 0.8$, $w_{\text{div}} = w_{\text{res}} = 0.5$,

Phase 3: $w_{\text{vel}} = w_p = 0.7$, $w_{\text{div}} = w_{\text{res}} = 0.7$,

Phase 4: $w_{\text{vel}} = w_p = 0.5$, $w_{\text{div}} = w_{\text{res}} = 1.0$.

This gradual increase of the physical term weights allowed the model to transition from data-driven fitting to fully physics-dominated training.

Results

The MeshGraphNet successfully reproduced the unsteady wake dynamics, capturing the periodic vortex shedding behind the obstacle (Figures 5.18, 5.19 and 5.20 show a snapshot of the results at time $t = 0$ for the single-rollout case). Even over extended rollouts, the model maintained temporal stability and correctly reconstructed the alternating vortices characteristic of the $Re = 200$ regime.

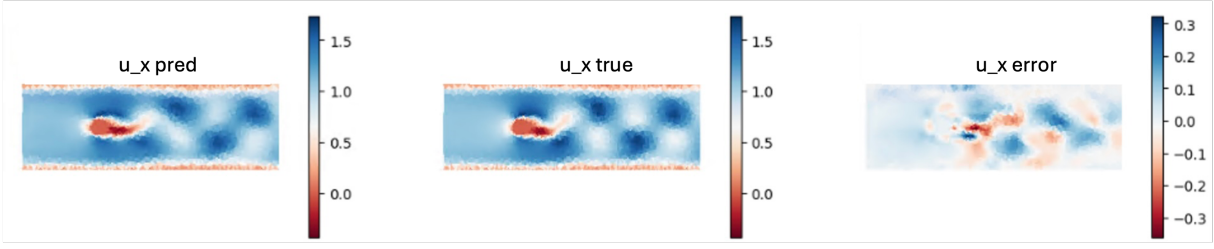


Figure 5.18: Predicted velocity along axes-x for the $Re = 200$ case. The network correctly reproduces the periodic vortex shedding in the wake region.

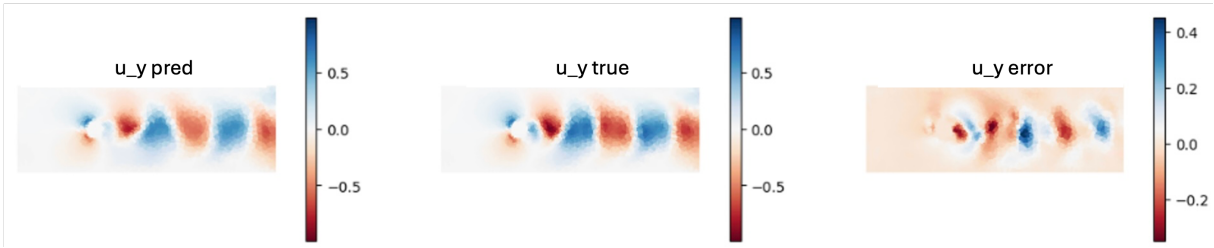


Figure 5.19: Predicted velocity along axes-y for the $Re = 200$ case. The network correctly reproduces the periodic vortex shedding in the wake region.

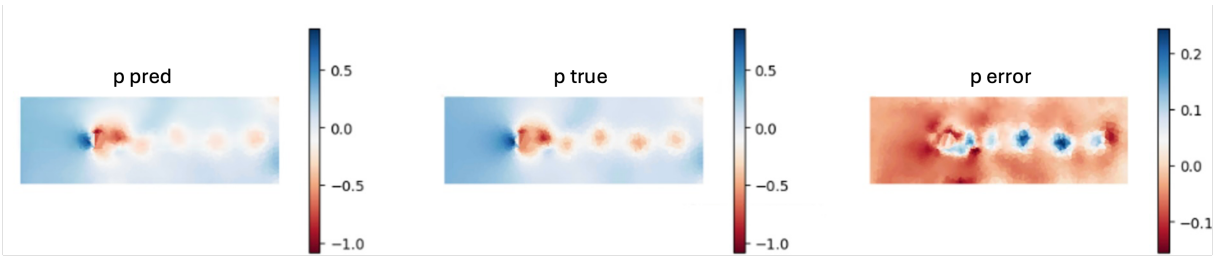


Figure 5.20: Predicted pressure for the $Re = 200$ case. The network correctly reproduces the periodic vortex shedding in the wake region.

Before analysing the temporal evolution of the error, we first examine its spatial distribution. We observe that the largest discrepancies are concentrated in the wake region behind the obstacle, which is expected since this is where vortices form and the flow becomes more complex.

We now examine the temporal evolution of the L_2 error, computed as the spatial average over all nodes in the domain. We first consider the short-horizon prediction task (30 time steps). As shown in Fig. 5.21, the error of the velocity u_x increases up to approximately the 19th time step, corresponding to one vortex-shedding period, reflecting the progressive

accumulation of model and discretization errors during autoregressive rollout. Beyond this point, the error does not continue to grow monotonically, but instead shows a slight reduction. This behavior suggests that the periodic structure of the flow may help limit further error growth once a full cycle has been reconstructed. In contrast, the error evolution for u_y and p exhibits a more clearly periodic pattern, with oscillations that closely follow the shedding cycle. In these cases, the modulation appears to be primarily driven by the intrinsic dynamics of the system rather than by a purely cumulative rollout effect.

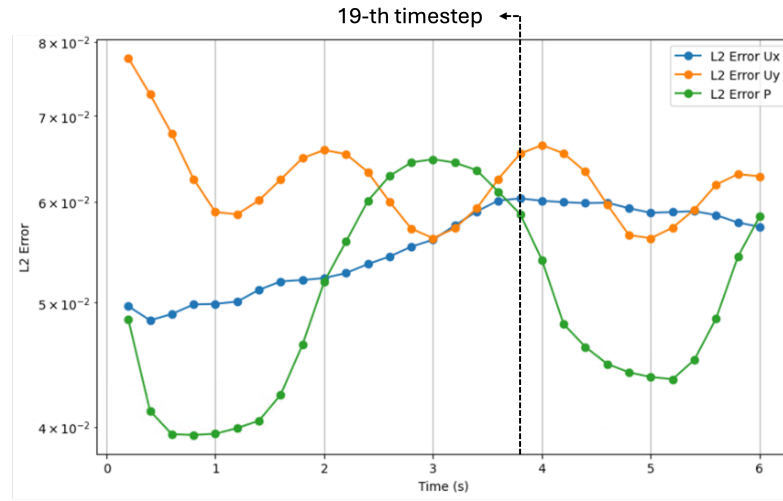


Figure 5.21: L2 error over the time, for the case test and trained with 30 timesteps .

We then consider the case in which the model is trained on two shedding periods $2T$, corresponding to 38 time steps. The evaluation is performed both on the training periods themselves (see Fig. 5.22) and on an additional unseen shedding period (see Fig. 5.23).

For the u_x component, the L_2 error increases up to approximately $t = 19$, corresponding to the end of the first shedding period. Beyond this point, the error does not evolve monotonically but instead exhibits alternating phases of growth and partial reduction, with local maxima occurring near successive shedding cycles. This indicates that the error evolution is modulated by the underlying periodic dynamics rather than being purely cumulative. The pressure error displays a similarly non-monotonic trend, with a pronounced maximum around $t = 19$ and a nearly symmetric pattern across the two periods. Although a mild cumulative tendency is still observable, the overall behavior reflects the periodic structure of the flow.

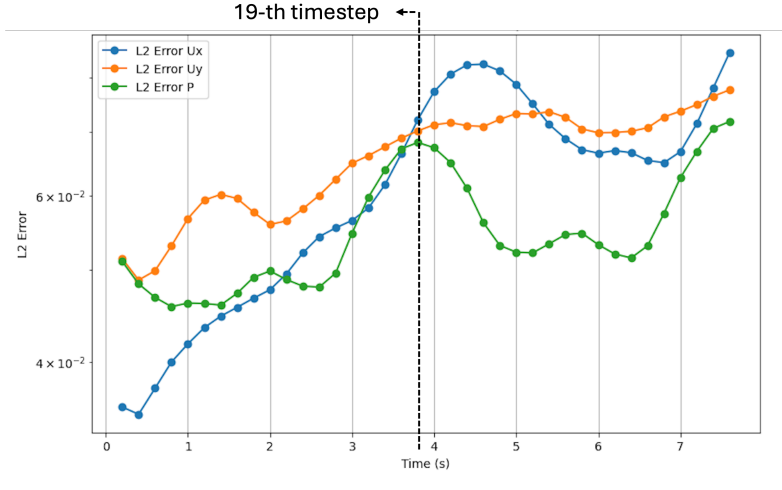


Figure 5.22: L2 error over the time, for the first two periods trained and tested (periodic regime).

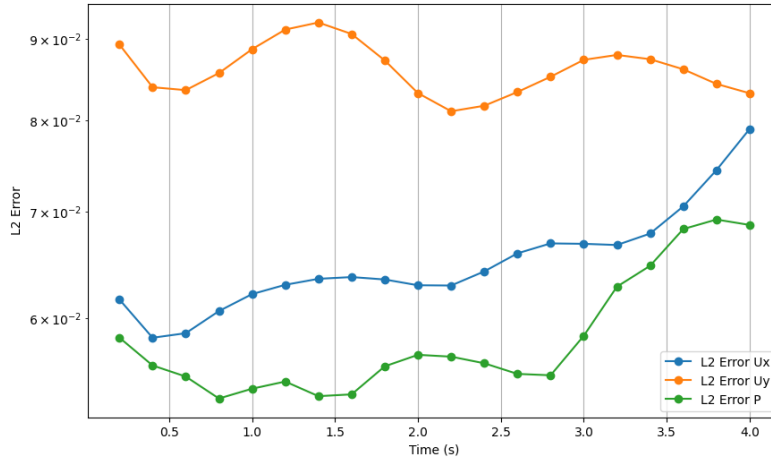


Figure 5.23: L2 error over the time, tested only on the last unseen period (periodic regime).

Although the model was trained only on the first two periods, the error over the unseen third period (See Fig. 5.23) remains well controlled. For both u_x and p , the L_2 error continues to increase over time, yet the growth remains moderate and does not exhibit runaway behavior or instability. In contrast, the u_y component shows a more clearly periodic modulation, closely aligned with the vortex-shedding cycle. Given that the third period was not included during training, these results provide strong evidence that the model is not simply memorizing specific temporal sequences, but has effectively learned the underlying oscillatory dynamics of the system. The MeshGraphNet therefore demonstrates a robust capacity to reproduce the dominant periodic flow mechanisms while maintaining bounded

and physically consistent predictions beyond the training horizon.

In summary, the $Re = 200$ experiments demonstrate that the adoption of a multi-phase loss-weighting strategy plays a key role in stabilizing the training process and in progressively enforcing physical consistency, as discussed in Section 5.2.1. At the same time, the model is able to generalize across multiple vortex-shedding cycles, maintaining coherent unsteady dynamics even in the absence of explicit supervision at each time step.

5.4.1. Comparison supervised vs unsupervised

To further assess the robustness of the proposed architecture, we conducted a comparative analysis between the hybrid quasi-unsupervised model and the fully supervised variant.

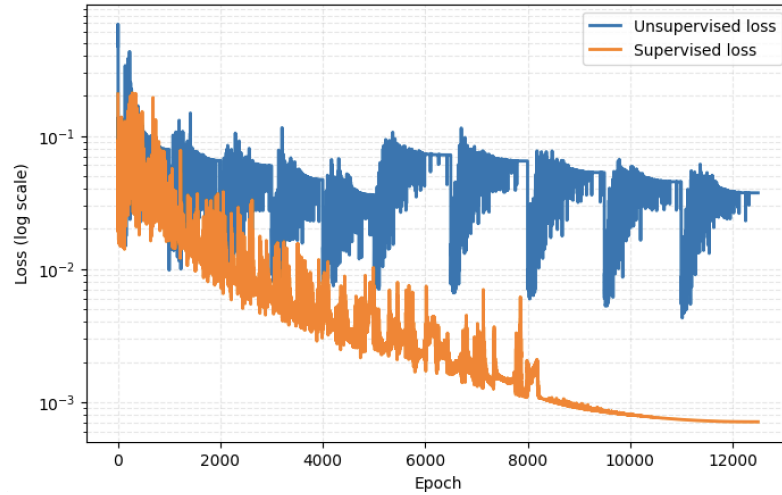


Figure 5.24: Behaviour of the velocity train loss for the supervised and for the unsupervised case.

As expected, in Figure 5.24 we clearly observe that the supervised loss decreases substantially more than the loss of the hybrid model, reflecting the additional guidance provided by the labelled targets.

However, when examining the temporal evolution of the L_2 error (see Fig. 5.25), the difference between the two models is not as pronounced as one might expect based solely on the behaviour of the supervised loss (see Fig. 5.24). In particular, although the fully supervised model achieves a lower training loss, this advantage does not translate into a substantially lower prediction error over time. On the contrary, the L_2 error curves tend to approach each other as the rollout progresses. This behaviour suggests that the fully supervised model may be fitting the labelled data more aggressively, without achieving a proportional gain in long-term generalization. In contrast, the hybrid model, despite

exhibiting a higher supervised loss, appears to retain a more faithful representation of the physical evolution, leading to comparable (and in some regions even lower) prediction errors over time.

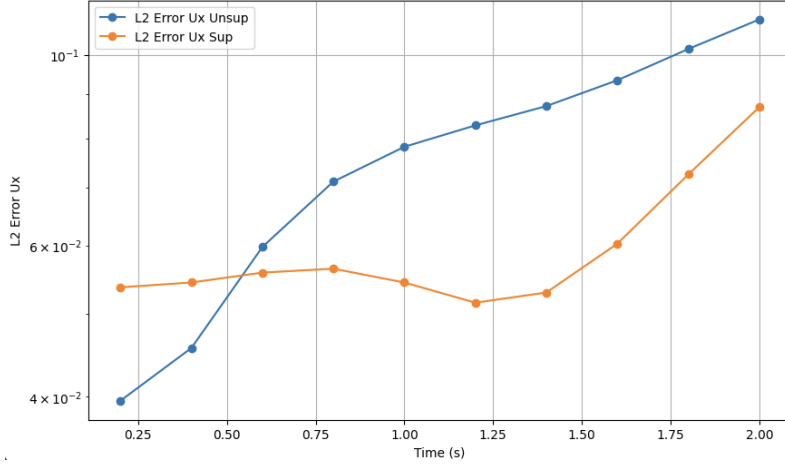


Figure 5.25: L2 error over the time for the supervised and the unsupervised cases.

5.5. Generalization for a larger dataset

We now move to the next step: assessing the ability of the model to generalise to a significantly larger and more diverse dataset, characterised by a greater number of flow trajectories. The dataset employed is the one provided in the Colab associated with the MeshGraphNets article by Pfaff [29]. This dataset, originally generated using COMSOL, consists of 100 trajectories, each comprising 600 timesteps. Each trajectory corresponds to a different configuration, varying both the position and the size of the obstacle, thus introducing substantial geometric and dynamical variability.

The same model architecture previously presented was used without modifications. Out of the 100 available trajectories, only 23 corresponded to unsteady flow regimes. Since the primary objective of this analysis was to assess the ability of the model to capture non-stationary dynamics, we restricted the study to this subset of unsteady cases. Among these 23 trajectories, 15 were used for training, while the remaining 8 were held out for testing. This split ensures that the evaluation is conducted on entirely unseen flow evolutions within the unsteady regime.

Figures 5.26, 5.27 and 5.28 illustrate three representative trajectories from the dataset used for training. It can be observed that the obstacle varies both in size and in spatial position within the domain. These geometric differences naturally lead to distinct wake structures and flow separation patterns, resulting in diverse unsteady dynamics across the

dataset.

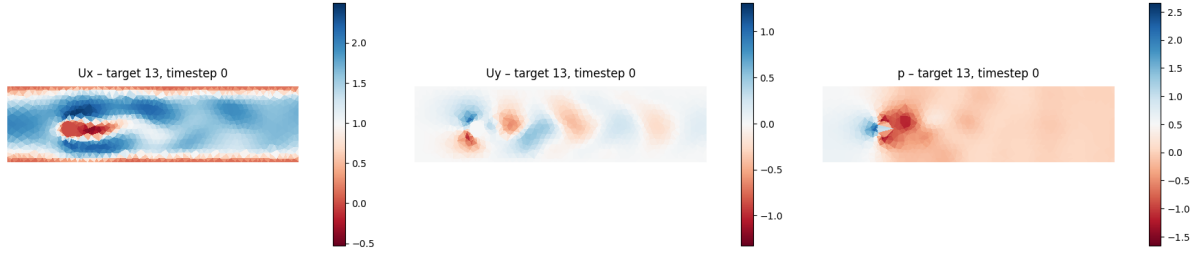


Figure 5.26: Example of the trajectories of the dataset.

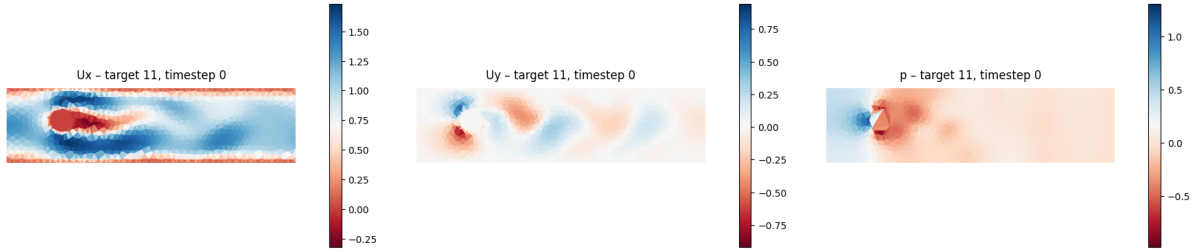


Figure 5.27: Example of the trajectories of the dataset.

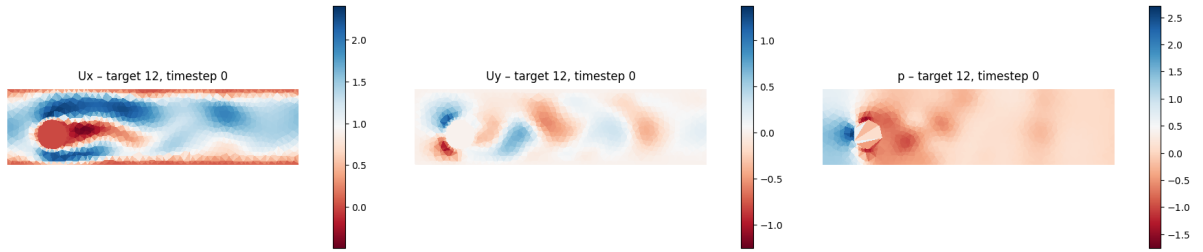


Figure 5.28: Example of the trajectories of the dataset.

The results obtained on this larger dataset are very encouraging. We directly compare the proposed hybrid model with the fully supervised baseline. Once again, the training velocity loss decreases much more steeply in the supervised case; however, when inspecting the validation loss evaluated on three trajectories that are not included in the training set a different picture emerges (See Fig. 5.29 and 5.30).

In the supervised setting, while the training loss continues to decrease significantly, the corresponding validation loss exhibits a much milder reduction, eventually flattening out. In contrast, for the hybrid model, both the training and validation losses decay with a similar slope. This behaviour is indicative of a potential overfitting regime for the purely supervised model: the training velocity error appears to keep improving, but this

improvement does not translate into better generalisation on unseen trajectories, whereas the hybrid approach maintains a more consistent gap between training and validation performance.

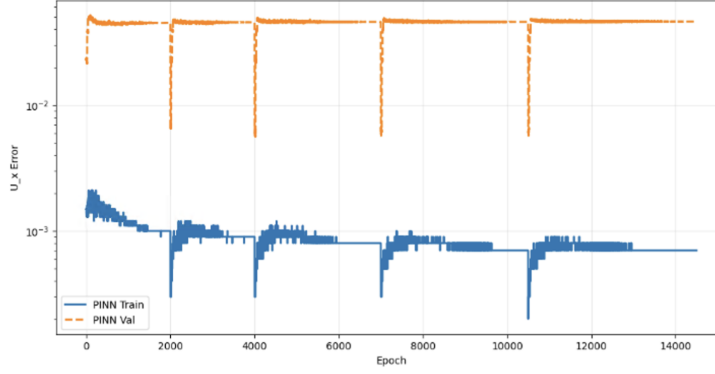


Figure 5.29: Comparison between the validation and the train loss of the velocity for our model.

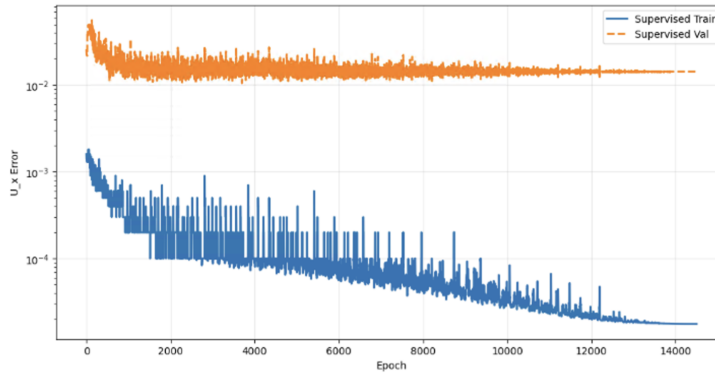


Figure 5.30: Comparison between the validation and the train loss of the velocity for the supervised case.

We now examine the temporal evolution of the mean squared error (MSE), averaged over eight test trajectories that were never seen during training (See Fig. 5.31). The gap between the supervised model and our hybrid approach is noticeably smaller than what one would infer from the training loss alone, confirming the behaviour already observed in the validation curves. Moreover, this gap tends to decrease as time progresses, indicating that the supervised model does not maintain its apparent advantage when evaluated on fully unseen rollout sequences.

These results support the conclusion that our model which relies predominantly on a physics-based loss over the majority of the domain is able to generalise remarkably well.

Despite using substantially fewer supervised signals, its predictive accuracy remains competitive with, and in several regimes approaches, that of the fully supervised baseline, while avoiding the overfitting tendencies displayed by the latter.

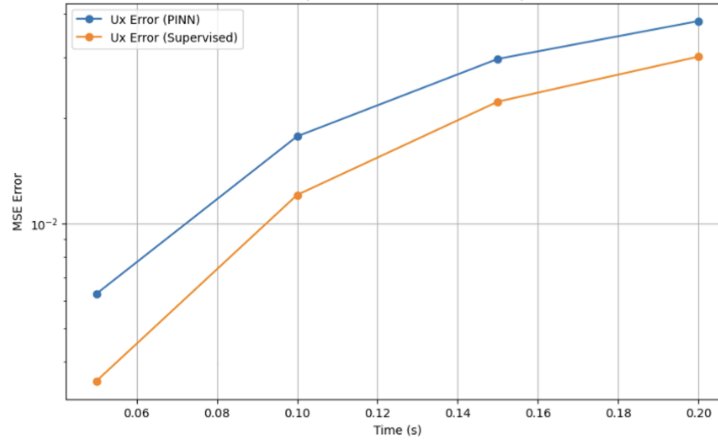


Figure 5.31: A mean of the Mean Squared Error over the 8 test trajectories. Comparison between supervised case and our model.

5.5.1. Limits

While the previous experiments demonstrate that the model generalises within the class of single obstacle configurations, where obstacle size and position vary but the domain topology remains unchanged, it is important to assess how far this capability extends. In particular, we investigate whether a model trained exclusively on one obstacle geometries can still produce physically consistent predictions when evaluated on a qualitatively different setup, namely a channel containing two obstacles. This experiment is not intended to test the mere reproduction of a new flow realization, which is expected to differ from those seen during training. Rather, it aims to assess whether the model has internalised the governing dynamics of the system, to the extent that it can operate as a stable time integrator of the underlying Navier–Stokes equations in a modified geometric configuration. To this end, we employ exactly the same model introduced in the previous section, without any retraining or architectural modification, and evaluate it on a newly generated dataset featuring two obstacles (see Fig. 5.32).

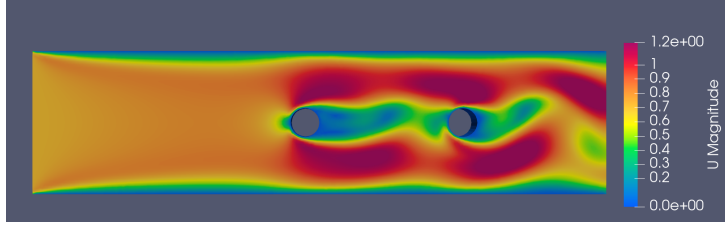


Figure 5.32: Configuration of the two-obstacle case in OpenFOAM. The figure shows the magnitude of the velocity field.

This scenario constitutes a genuine out of distribution generalisation test: although the model has learned to handle variability in geometry (size, position, wake features), it has never encountered multi-body interactions or the more complex vortex dynamics induced by two interacting wakes.

As expected, the performance of the model degrades noticeably in this setting (see Figs. 5.33, 5.34 and 5.35). While the initial steps of the rollout remain qualitatively plausible, the error grows much faster compared to the single obstacle case. In particular, for both u_x and the pressure field, the predicted dynamics fail to capture the correct wake evolution already at early stages of the rollout (around $t = 5$). The vortex structures in the wake are not reproduced accurately, and the temporal development does not reflect the expected interaction between the two obstacles. This indicates that the model does not generalise its learned dynamics to the altered topology of the domain.

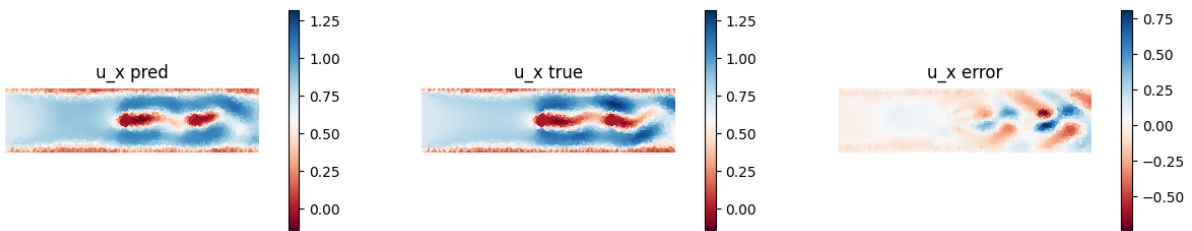


Figure 5.33: Predicted velocity along x-direction field compared with the OpenFOAM reference at the last timestep ($t = 5$).

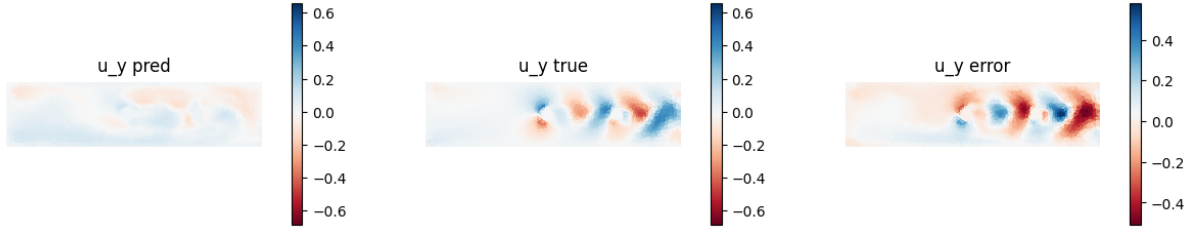


Figure 5.34: Predicted velocity along y-direction field compared with the OpenFOAM reference at the last timestep ($t = 5$).

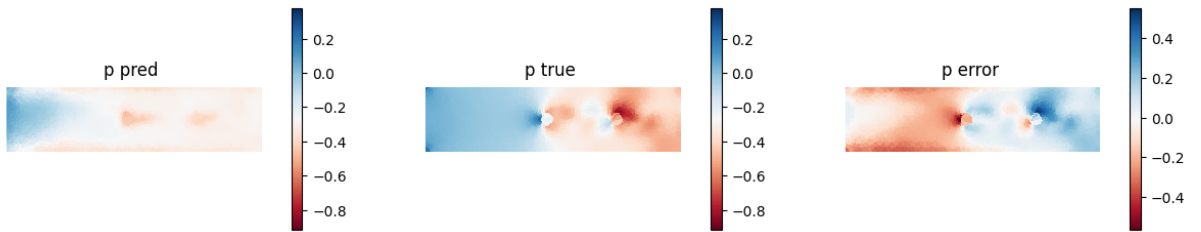


Figure 5.35: Predicted pressure field compared with the OpenFOAM reference at the last timestep ($t = 5$).

The predictions deteriorate particularly in the region where the two wakes interact, indicating that the features and message-passing patterns learned from single-obstacle flows do not transfer to more complex multi-obstacle regimes.

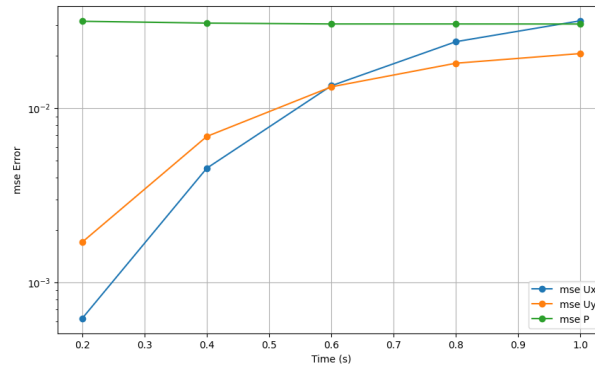


Figure 5.36: Mean Squared Error over the time.

This experiment highlights a limitation in topological generalisation: although the model exhibits strong robustness within the training distribution, it struggles to reliably extrapolate to configurations whose flow dynamics differ substantially from those encountered during training. This behavior suggests that the network may not learn a fully topology-independent representation of the governing physics, but instead relies, at least partially,

on interpolation within the manifold of solutions associated with the training configurations. Overcoming this limitation would likely require extending the training distribution to more diverse geometrical configurations or incorporating architectural mechanisms aimed at improving physics-consistent generalisation across topology changes.

5.6. Transitional airfoil obstacle flow ($Re = 2000$)

Now, to investigate the behavior of our model when applied to obstacles with different geometries, we extend the analysis to a domain consisting of a channel with an airfoil. To induce vortex shedding in the wake, the airfoil is inclined by 7.13° . Since the model has already been thoroughly validated in the previous configurations, we directly adopt the partially unsupervised training setup for this case.

Domain, mesh and input features

Again, the computational domain are identical to the previous cases, but the obstacle now is the airfoil, with a chord of 1 m. The timestep was set to $\Delta t = 0.05$. This value provides a suitable balance between numerical stability and temporal resolution. In particular, it ensures that the characteristic flow features generated around the airfoil, especially the onset and evolution of vortex shedding, are adequately resolved.

The features used in this configuration remain the same as in the previous cases. In particular, node features include the velocity at the previous timestep, the coordinates of the node and the vector indicating the type of the node:

$$\mathbf{x}_i = [u_{x,i}, u_{y,i}, x_i, y_i, \text{one_hot_type_vector}_i],$$

so that the model is explicitly conditioned on the current flow state in the autoregressive setting. The network outputs the velocity increments and the pressure,

$$\hat{\mathbf{y}}_i = [\Delta u_{x,i}, \Delta u_{y,i}, p_i].$$

All quantities are normalized using dataset-specific mean and standard deviation.

Training and Results

As said before, the training used a partially unsupervised scheme, applying a supervised mask at the boundaries and around the obstacle, and the unsupervised mask in the remaining points.

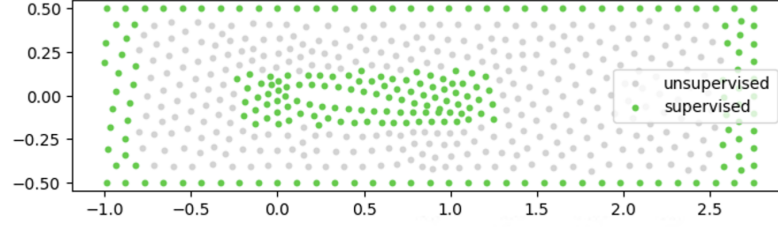


Figure 5.37: The subdivision of the nodes in supervised and unsupervised masks. The green one is the supervised mask and the grey one is the unsupervised.

The configuration is tested with a curriculum learning rollout of 20 timesteps. Again, the multi-phases strategy is applied. A three-phases loss schedule was used:

$$\begin{aligned} \text{Phase 1: } w_{\text{vel}} = w_p = 1.0, \quad w_{\text{div}} = w_{\text{res}} = 0.1, \\ \text{Phase 2: } w_{\text{vel}} = w_p = 1.0, \quad w_{\text{div}} = w_{\text{res}} = 0.2, \\ \text{Phase 3: } w_{\text{vel}} = w_p = 0.8, \quad w_{\text{div}} = w_{\text{res}} = 0.5, \end{aligned}$$

This soft gradual increase of the physical term weights is the one that better allowed the model to capture the dynamic of the problem.

Looking at the results, MeshGraphNet successfully reproduces the unsteady transitional wake. Moreover, even over a long temporal rollout of 20 timesteps, the model predictions remain stable and accurately track the evolving flow dynamics (See Fig. 5.38, 5.39 and 5.40).

Below, it is possible to see a timeframe of the simulation at the timestep $t = 1$, showing how the model can predict the dynamic, and the behaviour of the L2 error over the time.

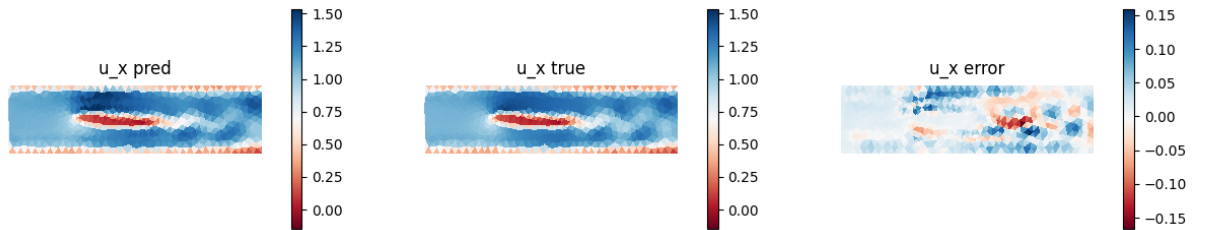


Figure 5.38: Predicted velocity along axes-x for the $Re = 2000$ case. The network correctly reproduces the vortex in the wake region.

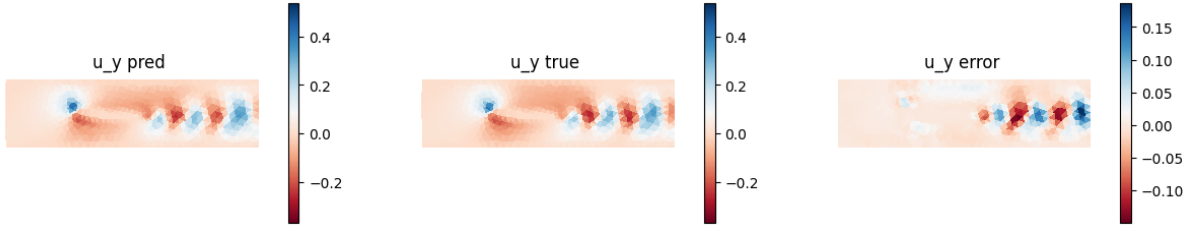


Figure 5.39: Predicted velocity along axes-x for the $Re = 2000$ case. The network correctly reproduces the vortex in the wake region.

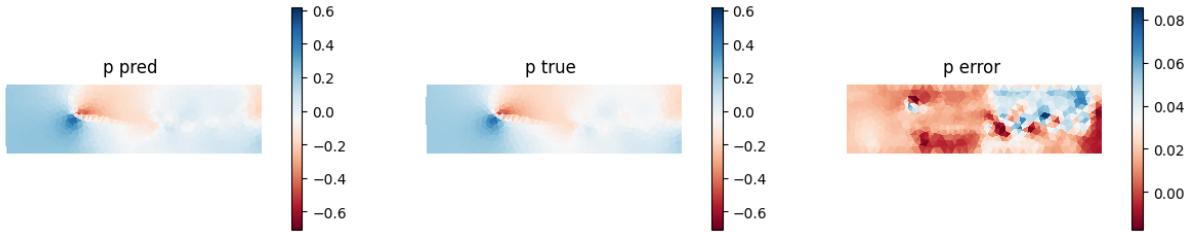


Figure 5.40: Predicted velocity along axes-x for the $Re = 2000$ case. The network correctly reproduces the vortex in the wake region.

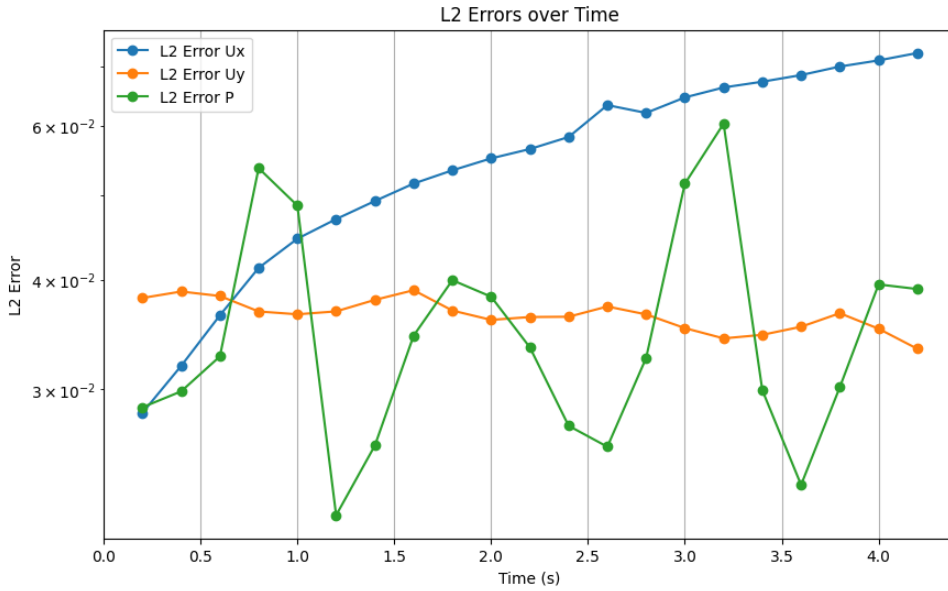


Figure 5.41: The mean of the L2 error on the computational domain over the time.

As expected from the analysis of the previous cases, the L_2 error remains bounded over time, although its evolution differs across variables. For u_x , the error exhibits a gradual increase along the rollout; however, the growth remains sub-linear and does not show

signs of divergence. Once again, this behaviour can be attributed both to numerical instabilities arising from the temporal and spatial discretization, and to the intrinsic model error introduced by the autoregressive rollout, which naturally accumulates over time. In contrast, the error associated with u_y remains approximately constant, indicating a comparatively stable prediction of the transverse velocity component. For the pressure field, the error displays oscillatory behaviour throughout the rollout. These oscillations appear to be correlated with the unsteady motion of the flow structures, suggesting that the pressure prediction is more sensitive to phase variations in the wake dynamics.

Moreover, the error is predominantly concentrated downstream of the obstacle, in the wake region where vortices are generated. This is fully consistent with our expectations, since the wake dynamics represent the most challenging part of the flow to learn and to accurately reproduce.

5.7. Turbulent square obstacle flow ($Re = 2 \times 10^6$)

The final test investigates a fully turbulent regime at $Re = 2 \times 10^6$, representing a challenging configuration characterized by strong unsteadiness, vortex breakdown, and large-scale turbulent structures in the wake of the obstacle. This setup was designed to test the upper limits of the MeshGraphNet model and assess its ability to reproduce high-Reynolds-number dynamics using physics-based inductive biases.

Setup and training configuration

The dataset for this case was generated using the solver OpenFOAM v2012, and consists of a single flow trajectory, which is used both for the training and the testing phases. The computational geometry is the same channel with obstacle configuration used in the previous tests, but here the flow is fully turbulent. The mesh is structured, and the obstacle is a square with $L = 1m$

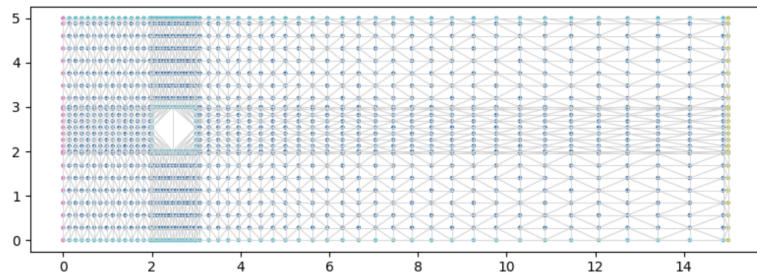


Figure 5.42: Structured mesh used for the turbulent channel-with-obstacle case. The grid is denser in the wake region to better resolve vortex shedding and turbulent fluctuations.

Given the inherent complexity of the turbulent flow, only the supervised training configuration was adopted. Each timestep from the OpenFOAM simulation was treated as a labeled target, allowing the network to learn both spatial coherence and temporal evolution. Snapshots were extracted every $\Delta t = 0.4$ s, and the model was trained to predict five consecutive timesteps ($t \in [0, 2.0]$ s).

The same MeshGraphNet architecture used in previous cases was employed, trained with the Adam optimizer and cosine learning-rate annealing over 1000 epochs. Gradient clipping ($\|\nabla\|_{\max} = 5.0$) was applied to maintain stability (see Table 4.1). Since the training was fully supervised, no multi-phase or physics-informed weighting was required. However, the model still used an autoregressive rollout approach, where predictions from one timestep were recursively used as inputs for the next.

The model inputs consisted of both geometric and physical quantities:

$$\mathbf{x}_i = [u_{x,i}, u_{y,i}, x_i, y_i, \text{one_hot_type_vector}_i].$$

while the outputs represented the predicted increments in velocity and the absolute pressure:

$$\hat{\mathbf{y}}_i = [\Delta u_{x,i}, \Delta u_{y,i}, p_i].$$

All quantities were normalized per dataset using z-score normalization (with mean and standard deviation computed independently for each simulation) and denormalized only for evaluation and post-processing.

Results and discussion

The trained model reproduced the overall flow topology and major vortex structures of the turbulent wake, maintaining coherence over multiple timesteps (See Figures 5.43, 5.44 and 5.45). Qualitatively, the MeshGraphNet captured the alternating vortex shedding behind the obstacle and reproduced the large-scale unsteady patterns observed in the OpenFOAM reference.

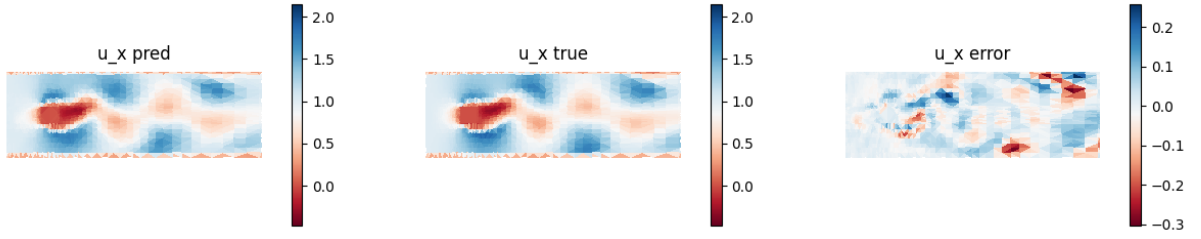


Figure 5.43: Comparison between MeshGraphNet predictions and OpenFOAM reference fields of the velocity along x-direction at the timestep $t = 1$.

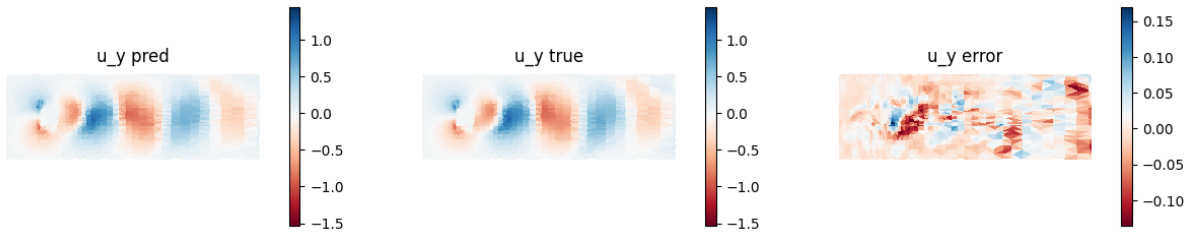


Figure 5.44: Comparison between MeshGraphNet predictions and OpenFOAM reference fields of the velocity along y-direction at the timestep $t = 1$.

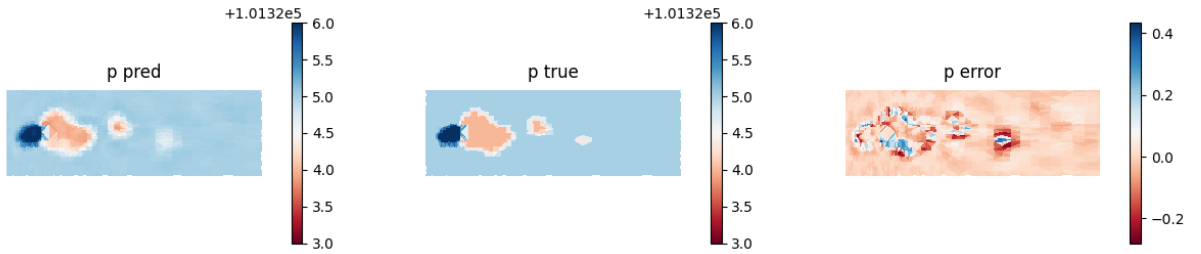


Figure 5.45: Comparison between MeshGraphNet predictions and OpenFOAM reference fields of the pressure at the timestep $t = 1$.

Quantitatively, the short-horizon predictions (first few timesteps) achieved L_2 errors on the order of 10^{-2} – 10^{-1} for the velocity components, comparable to the lower-Re cases (See Fig. 5.46). However, as the rollout length increased, the cumulative temporal error grew approximately linearly, highlighting the well-known challenge of error accumulation in autoregressive GNNs for unsteady or chaotic flows.

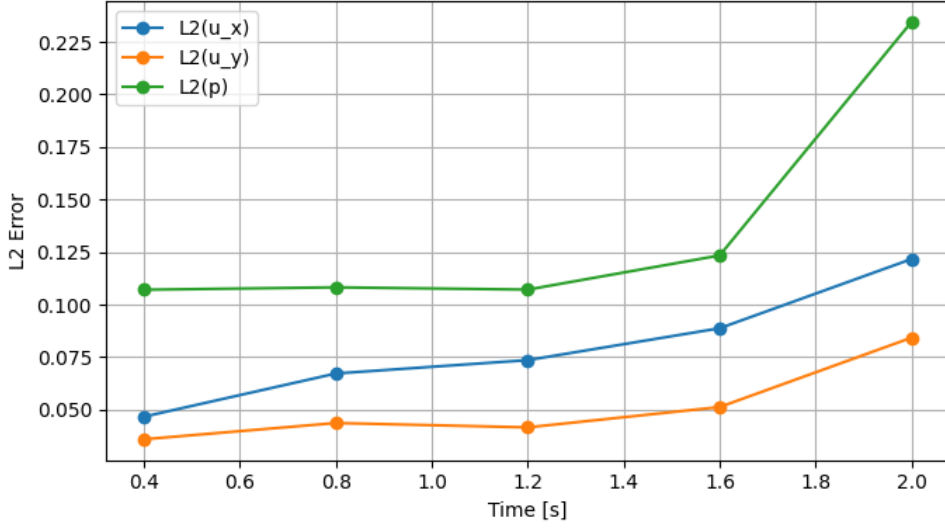


Figure 5.46: Evolution of the mean L_2 error over time during autoregressive rollout for the test of the turbulent case ($Re = 2 \times 10^6$). The error grows linearly with time due to temporal compounding.

Overall, the results demonstrate that the MeshGraphNet architecture can generalize to complex turbulent configurations when sufficient supervised data are available. Nevertheless, long-horizon temporal stability remains the main limitation of purely supervised autoregressive training, suggesting that future developments should incorporate hybrid physics–data regularization or recurrent-in-time strategies to mitigate error accumulation in chaotic flow regimes.

6 | Extension to Fluid-Solid Interaction

This chapter extends the proposed quasi-unsupervised MeshGraphNet framework to a significantly more demanding class of problems, namely fluid–solid interaction (FSI). The objective is not limited to accurate short-term predictions, but to evaluate the adaptability, stability, and physical consistency of the proposed methodology when applied to configurations with moving boundaries and strong fluid–structure coupling.

In contrast to the previously analyzed benchmarks involving fixed geometries, FSI problems introduce an additional layer of complexity: the computational domain itself evolves in time as a consequence of structural motion. The geometry therefore becomes a dynamic variable of the system, and its evolution directly affects the governing equations, boundary conditions, and stability of autoregressive rollouts.

To investigate the transferability of the framework to this setting, multiple training strategies are explored. A fully supervised configuration is first considered (see Section 6.2), where both fluid variables and structural displacements are learned directly from data. Subsequently, a partially unsupervised setting is analyzed by considering two configurations. In the first, the mesh motion is imposed, whereas in the second the structural displacement is predicted by the model, while the remaining quantities are inferred indirectly through the governing equations and coupling constraints (see Section 6.3). By systematically comparing these configurations, this Chapter aims to clarify how sensitive the model is to geometric drift, how error propagates in the presence of moving interfaces, and to what extent the proposed framework can be extended to problems that are inherently more complex than fixed-domain flow benchmarks.

6.1. Fluid-Structure Interaction: mathematical framework

Fluid–structure interaction problems describe systems in which a deformable solid interacts dynamically with a surrounding fluid. The coupling is inherently bidirectional: fluid stresses induce structural deformation, while structural motion modifies the fluid domain and consequently alters the governing flow equations. This mutual interaction introduces strong nonlinearities and geometric dependencies that make FSI significantly more challenging than fixed-domain flow problems.

FSI plays a central role in several engineering and biomedical applications. In aeroelasticity, the interaction between airflow and flexible wings or panels may lead to flutter and dynamic instabilities [61]. Offshore and marine structures experience vortex-induced vibrations due to current loading. In biomechanics, fluid–structure interaction is fundamental for modeling blood flow in compliant vessels, heart valves, and aneurysms [62, 63]. In such contexts, vessel wall compliance directly affects pressure wave propagation, wall shear stress distribution, and flow patterns, making FSI modeling essential for physiological accuracy.

From a mathematical standpoint, FSI involves the coupling of the incompressible Navier–Stokes equations in the fluid domain with structural mechanics equations in the solid domain.

Let $\Omega_f(t) \subset \mathbb{R}^2$ denote the two-dimensional fluid domain and $\Omega_s(t)$ the solid domain at time t .

The fluid motion is governed by the incompressible Navier-Stokes equations:

$$\rho_f \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) = -\nabla p + \mu_f \Delta \mathbf{u} \quad \text{in } \Omega_f(t), \quad (6.1)$$

$$\nabla \cdot \mathbf{u} = 0 \quad \text{in } \Omega_f(t), \quad (6.2)$$

The structural displacement field $\mathbf{d}(\mathbf{x}, t)$ in the solid domain satisfies the elastodynamics equation:

$$\rho_s \frac{\partial^2 \mathbf{d}}{\partial t^2} - \nabla \cdot \boldsymbol{\sigma}_s = 0 \quad \text{in } \Omega_s(t), \quad (6.3)$$

where $\boldsymbol{\sigma}_s$ denotes the Cauchy stress tensor of the solid and ρ_s its density. Volumetric body forces are neglected.

At the fluid-structure interface $\Gamma_{FSI}(t)$, two coupling conditions are imposed:

- Kinematic condition (velocity continuity):

$$\mathbf{u} = \frac{\partial \mathbf{d}}{\partial t} \quad \text{on } \Gamma_{FSI}(t),$$

- Dynamic condition (stress equilibrium):

$$\boldsymbol{\sigma}_f \mathbf{n} = \boldsymbol{\sigma}_s \mathbf{n} \quad \text{on } \Gamma_{FSI}(t),$$

where σ_f is the fluid stress tensor and n the interface normal.

6.1.1. Arbitrary Lagrangian–Eulerian formulation

A central difficulty in FSI problems arises from the fact that the fluid domain $\Omega_f(t)$ is time-dependent. To handle the moving boundary while preserving a mesh-based discretization, the Arbitrary Lagrangian–Eulerian (ALE) formulation is commonly adopted [64].

In the ALE framework, the fluid equations are rewritten in a reference configuration through a time-dependent mapping

$$\mathcal{A}_t : \Omega_f(0) \rightarrow \Omega_f(t),$$

which defines the mesh displacement. The mesh velocity is defined as

$$\mathbf{w} = \frac{\partial \mathcal{A}_t}{\partial t}.$$

The Navier-Stokes equations in ALE form become

$$\rho_f \left(\frac{\partial \mathbf{u}}{\partial t} + ((\mathbf{u} - \mathbf{w}) \cdot \nabla) \mathbf{u} \right) = -\nabla p + \mu_f \Delta \mathbf{u}, \quad (6.4)$$

$$\nabla \cdot \mathbf{u} = 0. \quad (6.5)$$

The additional term involving the mesh velocity $\mathbf{w}$ accounts for the moving computational frame.

6.1.2. Hron-Turek FSI3 benchmark

To evaluate the proposed framework on a standardized and widely adopted fluid–structure interaction configuration, the Hron-Turek FSI3 benchmark [65] is considered. The numerical reference solution is generated using the official implementation provided within the solids4foam extension of OpenFOAM [66].

The benchmark consists of a two-dimensional channel containing a rigid circular cylinder with an attached elastic beam. The incoming laminar flow induces vortex shedding behind the cylinder, which excites oscillations of the flexible structure. The resulting coupled dynamics make this configuration a stringent test for both classical numerical solvers and learning-based models.

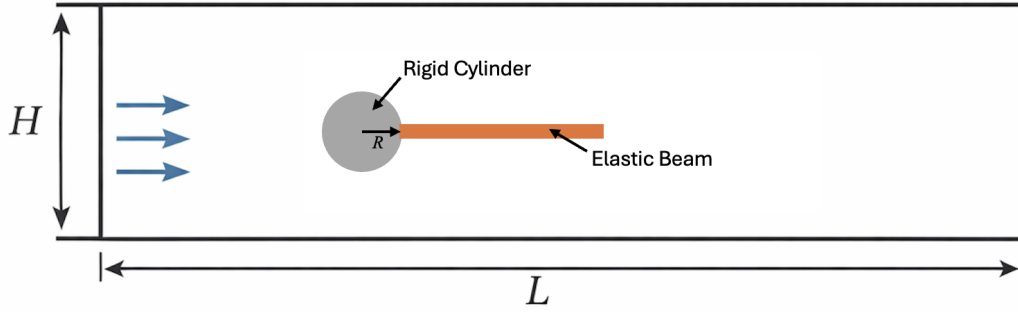


Figure 6.1: Schematic representation of the Hron-Turek FSI3 configuration.

A schematic representation of the computational domain and its main geometric quantities is shown in Figure 6.1. The channel has length L and height H . A rigid cylinder of radius R is positioned near the inlet, and a flexible beam of length l_s is clamped to its downstream side. The main physical and geometrical parameters of the FSI3 benchmark are summarized in Table 6.1.

Quantity	Symbol	Value
Channel length	L	2.5 m
Channel height	H	0.41 m
Cylinder radius	R	0.05 m
Beam length	l_s	0.35 m
Fluid density	ρ_f	1000 kg/m ³
Fluid viscosity	μ_f	1 Pa·s
Solid density	ρ_s	1000 kg/m ³
Young's modulus	E	1.4×10^6 Pa
Poisson ratio	ν_s	0.4
Time step	dt	0.02 s

Table 6.1: Parameters of the Hron–Turek FSI3 benchmark.

The fluid is modeled as incompressible and Newtonian, and the structure is described by linear elasticity. Mesh motion in the fluid domain is handled through an Arbitrary Lagrangian-Eulerian (ALE) formulation, consistently with the reference implementation in OpenFOAM.

In Figure 6.2, a representative snapshot of the velocity magnitude field obtained from the OpenFOAM FSI3 simulation is shown. The wake formation behind the cylinder and the deformation of the elastic beam are clearly visible.

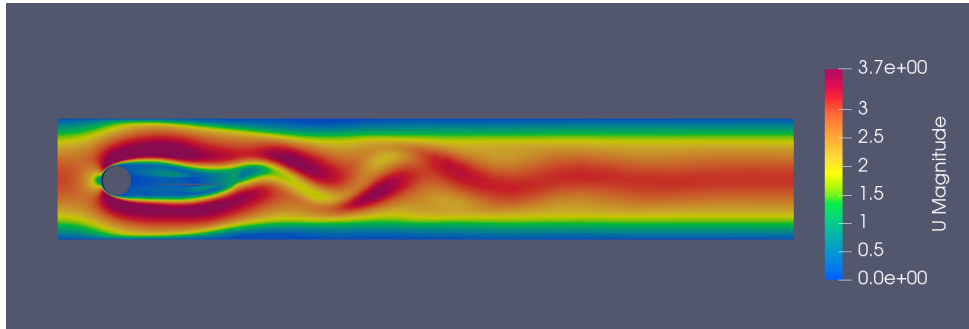


Figure 6.2: Velocity magnitude field of the Hron–Turek FSI3 configuration obtained with OpenFOAM. The vortex shedding behind the cylinder excites oscillations of the attached elastic beam.

To further highlight the structural dynamics, two additional zoomed-in snapshots corresponding to opposite displacement peaks are reported in Figure 6.3. These views focus on the near-field region around the elastic beam and emphasize the alternating deflection

induced by the vortex shedding mechanism. The apparent misalignment between the beam surface and the surrounding field is a visualization effect related to the rendering of the moving mesh and does not reflect any inconsistency in the simulation.

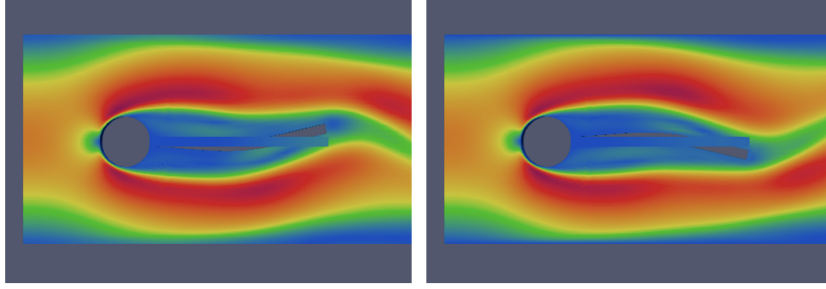


Figure 6.3: Zoomed-in views of the elastic beam at two opposite displacement peaks within one oscillation cycle. The consistent color scale allows a direct qualitative comparison of the flow structure and beam deformation.

6.2. Supervised case

Before introducing configurations in which physics-based constraints replace direct supervision in parts of the domain, a fully supervised configuration is first analyzed. This choice follows the same methodological progression adopted in the previous benchmarks: the objective is to verify that the proposed MeshGraphNet framework can correctly represent and learn the coupled dynamics of a moving-boundary FSI problem when complete state information is available.

The supervised setting therefore serves as a reference baseline, allowing us to assess whether the graph-based formulation, the ALE representation, and the dynamic reconstruction of the mesh connectivity are implemented consistently. Establishing this baseline is essential before investigating more challenging configurations in which supervision is partially removed or geometric evolution is treated differently.

6.2.1. Modeling strategy and learning setup

In the fully supervised configuration, the same MeshGraphNet architecture adopted in the previous chapters is employed, without modifications to the network structure or hyperparameters. The model configuration and training parameters therefore coincide with those summarized in Table 4.1, ensuring full methodological consistency across all benchmarks.

Unlike the fixed-domain cases, the FSI configuration involves a time-dependent computa-

tional mesh. The graph is constructed from the OpenFOAM finite-volume discretization exported at each time step. The centers of the control volumes are treated as spatial samples and used as graph nodes, without performing any interpolation or remeshing procedure. A representative mesh snapshot is shown in Figure 6.4.

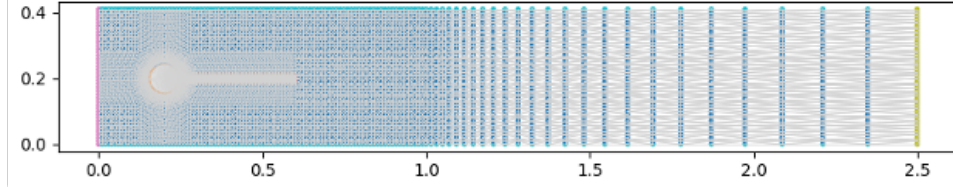


Figure 6.4: Computational mesh at the initial timestep of the Hron-Turek FSI3 configuration extracted from OpenFOAM. All mesh vertices are used as graph nodes in the supervised setting.

Each mesh vertex is treated as a graph node and classified according to its physical role (fluid, solid, obstacle, boundary, interface or interior). This information is encoded through a one-hot type vector included in the node feature representation.

For each node i at time step t , the feature vector is defined as

$$\mathbf{x}_i = [u_{x,i}^t, u_{y,i}^t, x_i^t, y_i^t, \text{one_hot_type_vector}_i],$$

where $(u_{x,i}^t, u_{y,i}^t)$ denote the fluid velocity components and (x_i^t, y_i^t) are the current ALE coordinates of the mesh node. Since the geometry evolves in time, spatial coordinates are dynamic variables and directly affect the message-passing process.

The model predicts incremental updates of the fluid state,

$$\hat{\mathbf{y}}_i^t = [\Delta u_{x,i}^t, \Delta u_{y,i}^t, p_i^{t+1}],$$

where $\Delta u_{x,i}^t$ and $\Delta u_{y,i}^t$ are the variation of the velocities along the x-direction and the y-direction for the i -node at the timestep t , and p is the pressure.

A key difference with respect to the previous fixed-geometry benchmarks concerns the geometric evolution of the graph. Although the mesh topology remains unchanged throughout the simulation, node coordinates vary over time due to ALE deformation. As a consequence, edge attributes must be recomputed at each time step to remain consistent with the instantaneous mesh configuration. In particular, relative position vectors and their norms are updated according to the current node coordinates. The edge connectivity

itself is preserved, while the geometric features used during message passing reflect the evolving domain.

6.2.2. Results

The supervised configuration is trained using autoregressive rollouts of 5 time steps, whereas evaluation is carried out on extended trajectories of 20 time steps. This setup makes it possible to evaluate the predictive accuracy of the learned dynamics beyond the training horizon, in terms of the capability of the solution to preserve physical correctness over longer temporal evolutions.

Figure 6.5 reports the evolution of the L2 error over time. For both the velocity in the x-direction, u_x , and the pressure, p , a gradual increase in the error is observed as the autoregressive rollout progresses, which is consistent with typical autoregressive behavior. Nevertheless, the growth remains moderate and does not show any divergent trend within the considered prediction window, indicating that the accumulation is mainly due to standard rollout error propagation rather than to an intrinsic instability of the learned dynamics. In contrast, the error associated with the transverse velocity component, u_y , remains approximately constant over time, with only mild oscillations that may be related to periodic flow features such as the inversion of the wake.

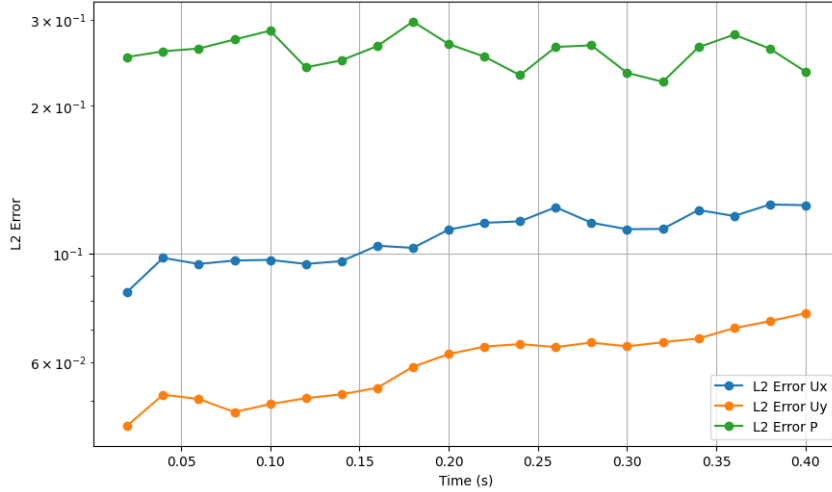


Figure 6.5: L2 error over the time of the velocity along x-direction, the velocity along y-direction, and the pressure

Qualitative comparisons in Figures 6.6, 6.7 and 6.8, reported at time step 20, further confirm this observation. The predicted velocity field preserves the main flow structures, including vortex shedding behind the cylinder, while the deformation of the elastic beam

remains consistent with the reference solution. In particular, the tail motion follows the structural displacement without visible phase shift or artificial damping within the considered rollout horizon. A particularly relevant aspect is that autoregressive error growth remains bounded, and the solution preserves physical consistency over a temporal window four times longer than the one used during training. This behavior is likely facilitated by the periodic nature of the vortex shedding dynamics, which enables the network to extrapolate the learned evolution beyond the supervised rollout length.

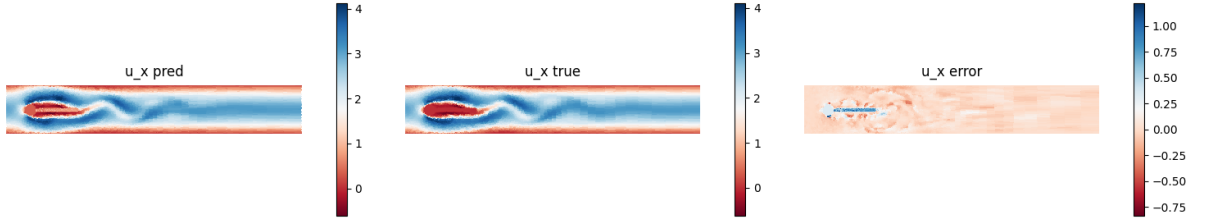


Figure 6.6: Predicted, L_2 error and OpenFOAM results for the velocity component along the x -direction (u_x) in the fully supervised configuration.

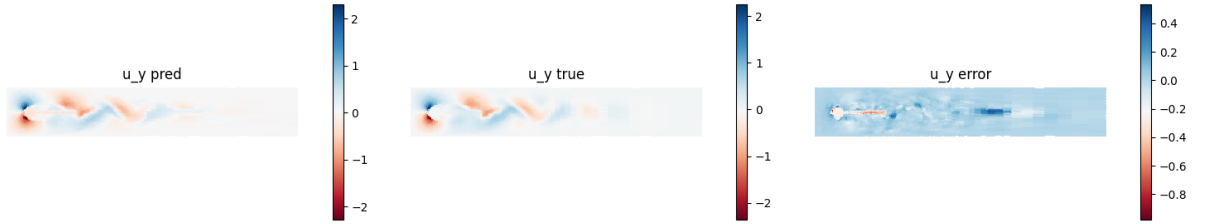


Figure 6.7: Predicted, L_2 error and OpenFOAM results for the velocity component along the y -direction (u_y) in the fully supervised configuration.

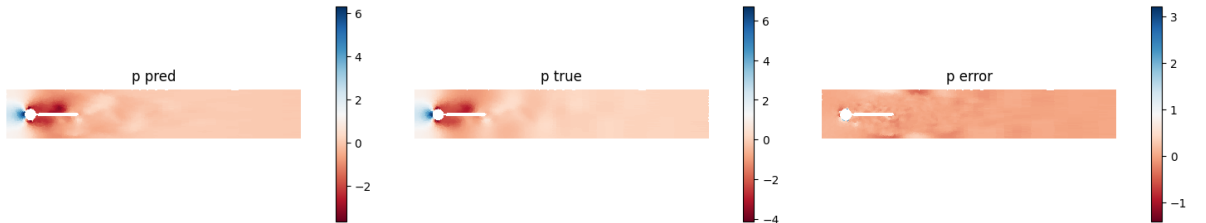


Figure 6.8: Predicted, L_2 error and OpenFOAM results for the pressure field (p) in the fully supervised configuration.

These results indicate that, when full supervision is available, the proposed graph-based formulation is capable of learning the coupled fluid–structure dynamics with satisfactory

accuracy. The moving mesh and interface coupling do not introduce significant degradation compared to the fixed-domain benchmarks, provided that displacement and flow variables are directly supervised.

6.3. Partially unsupervised case

After validating the framework under full supervision, a partially supervised configuration is considered in order to assess the robustness of the model when supervision is spatially restricted.

In this setting, supervision is not removed globally but applied only in selected regions of the domain. As shown in Figure 6.9, a narrow band near the domain boundaries and around the rigid cylinder is supervised, while the interior fluid region remains unsupervised. The supervision near the domain boundaries ensures that inlet, outlet, and wall conditions remain anchored to the reference solution, preventing global drift during autoregressive rollouts. Similarly, supervision around the cylinder and fluid–structure interface stabilizes the highly nonlinear interaction region where strong velocity gradients and vortex shedding originate. In contrast, the interior fluid region is left unsupervised so that the flow evolution is primarily governed by the physics-based residual constraints. This configuration allows us to assess whether the imposed Navier–Stokes equations are sufficient to reconstruct coherent flow structures.

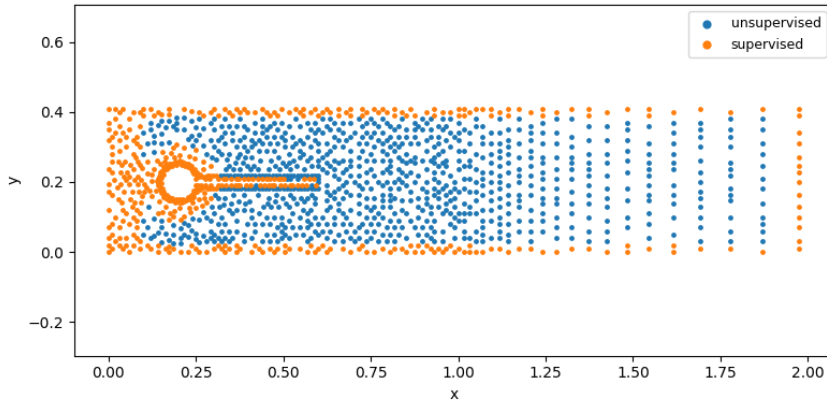


Figure 6.9: The subdivision of the nodes in supervised and unsupervised masks. The orange one is the supervised mask and the blue one is the unsupervised.

6.3.1. Modeling strategy and geometric treatment

Two variants are analyzed in order to disentangle the role of geometric supervision within the coupled fluid–structure dynamics.

In the first variant (ALE teacher-forced), the mesh coordinates (x_i, y_i) at each time step are directly replaced by the reference ALE positions obtained from the OpenFOAM simulation, and the corresponding mesh velocity is computed consistently from this prescribed geometry. Under this setting, the geometric evolution is entirely imposed and the network is required to predict only the fluid state increments, thus isolating the learning of the fluid dynamics from potential error accumulation in the structural motion. In the second variant, the structural displacement is instead predicted by the network and used to update the solid geometry at each time step. However, this displacement prediction is still directly supervised, ensuring that the structural evolution remains anchored to the reference solution while allowing the model to actively participate in the geometric update. The fluid mesh motion is then computed using the reference mesh-velocity field, and the model outputs therefore include both fluid increments and solid displacement increments. This comparison enables the assessment of how progressively relaxing geometric constraints affects stability and accuracy in the coupled fluid–structure prediction. It is worth noting that the increased difficulty in the second variant does not arise only from reduced geometric supervision, but also from the fact that the model must predict additional state variables, namely the structural displacement increments. As a result, the learned dynamical system becomes more strongly coupled and inherently more challenging to predict over multiple time steps.

In both cases, the graph topology remains fixed, while edge attributes are recomputed at each time step to reflect the current ALE configuration. Due to the large number of control volumes in the original OpenFOAM mesh, leading to substantial memory requirements and quadratic growth in message-passing cost, a coarsened version of the mesh is employed. The resulting node distribution is shown in Figure 6.10.

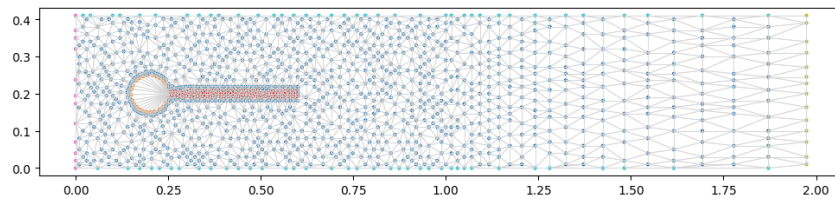


Figure 6.10: Node distribution of the coarsened mesh used for the FSI simulations, obtained from the original OpenFOAM geometry to reduce the computational complexity while preserving the main spatial features.

Due to the presence of mesh motion in the ALE framework, the RBF-FD discretization adopted in the fixed-domain experiments is no longer applicable. Besides requiring the reconstruction of the differentiation weight matrices at every time step, the ALE setting

would force the RBF-FD operators to be recomputed throughout each autoregressive rollout and for every training epoch. While in the fixed-domain case the differentiation weights are assembled only once, here they would be reconstructed at every rollout step and across all epochs, effectively multiplying the preprocessing cost by the number of time steps and training iterations. This repeated reconstruction leads to a substantial computational burden, making the approach impractical for long rollouts. Furthermore, the evolving mesh configuration may fail to satisfy the geometric quality criteria discussed in Section 3.4. This can lead to poorly conditioned local systems and, consequently, to numerical instability in the evaluation of spatial derivatives. For this reason, spatial differential operators are instead computed using a finite element (FEM) formulation, which naturally accommodates mesh deformation and allows stable operator assembly at each time step [67]. The FEM-based discretization therefore ensures robustness of the physics-informed residual evaluation throughout the coupled fluid–structure evolution.

The loss function combines spatially restricted supervision with physics-informed constraints. The incompressible Navier-Stokes residual in ALE form is enforced in the unsupervised region, including the convective term $(\mathbf{u} - \mathbf{w}) \cdot \nabla \mathbf{u}$ and the divergence-free condition.

A multi-phase training schedule is adopted, progressively increasing the weight of the physics-informed terms while reducing the relative contribution of supervised terms. The weighting strategy is summarized below:

Phase 1: $w_{\text{vel}} = 1.0$, $w_p = 1.0$, $w_{\text{div}} = 0.1$, $w_{\text{res}} = 0.1$, $w_{\text{dsup}} = 1.0$,

Phase 2: $w_{\text{vel}} = 1.0$, $w_p = 1.0$, $w_{\text{div}} = 0.2$, $w_{\text{res}} = 0.2$, $w_{\text{dsup}} = 1.0$,

Phase 3: $w_{\text{vel}} = 0.8$, $w_p = 0.8$, $w_{\text{div}} = 0.5$, $w_{\text{res}} = 0.5$, $w_{\text{dsup}} = 0.8$,

Phase 4: $w_{\text{vel}} = 0.7$, $w_p = 0.7$, $w_{\text{div}} = 0.7$, $w_{\text{res}} = 0.7$, $w_{\text{dsup}} = 0.7$,

Phase 5: $w_{\text{vel}} = 0.5$, $w_p = 0.5$, $w_{\text{div}} = 1.0$, $w_{\text{res}} = 1.0$, $w_{\text{dsup}} = 0.5$.

It is worth noting that, in the ALE teacher-forced configuration where the mesh motion is imposed, the displacement supervision term is not included, and therefore the weight w_{dsup} is omitted from the loss definition.

6.3.2. Results

Training is performed using autoregressive rollouts of 5 time steps. Evaluation is also conducted over the same 5 step horizon, as prediction quality degrades significantly beyond the fourth time step, corresponding to the onset of tail reversal.

Figures 6.11 and 6.12 report the temporal evolution of the L2 error for the two ALE variants. In both cases, the error remains bounded and does not exhibit divergence over time. The magnitude of the error is of the order of 10^{-1} for all the considered quantities, indicating a non negligible but controlled deviation from the reference solution. For the transverse velocity component u_y , the error remains approximately constant throughout the rollout, whereas for u_x and the pressure field a gradual increase is observed. Importantly, however, this growth does not lead to numerical instability or unbounded behaviour. A slight quantitative improvement can be observed in the case where the ALE formulation is imposed directly, which consistently yields lower error levels compared to the supervised- d variant. Nevertheless, the difference between the two approaches remains moderate, and the overall trends are qualitatively similar.

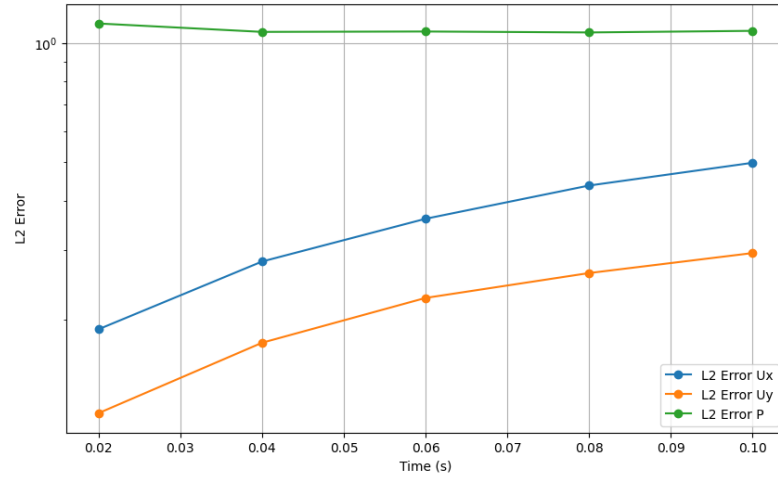


Figure 6.11: L2 error over the time in the supervised- d configuration

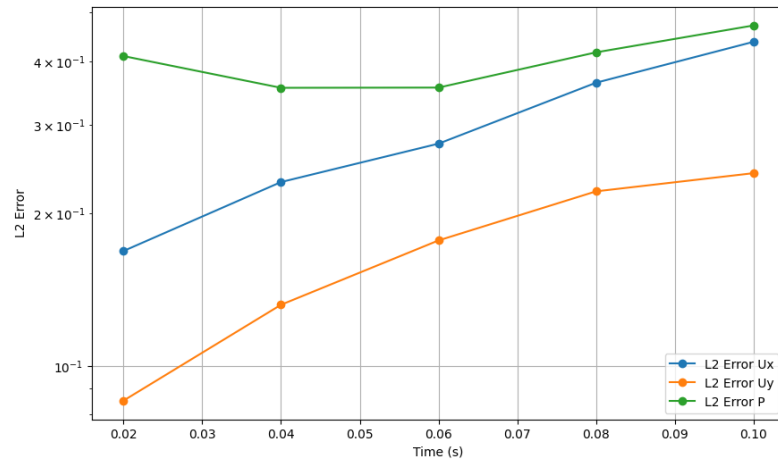


Figure 6.12: L2 error over the time in the imposed ALE configuration.

A qualitative inspection of the predicted fields at the final rollout step, corresponding to $t = 5$, reveals a substantially different picture. The snapshots reported in Figs. 6.13–6.15 and Figs. 6.16–6.18 explicitly refer to this time instant. In both configurations, the underlying flow dynamics are not effectively captured.

In the case where the displacement field $\mathbf{d}$ of the solid is only supervised and not explicitly imposed, the fluid–solid interaction appears to be poorly resolved. This limitation is particularly evident in the u_x component, where the expected modulation induced by the obstacle motion is largely absent. The wake structure does not reflect the correct kinematic coupling between the moving boundary and the surrounding fluid, suggesting that the learned model fails to internalize the dynamical interaction mechanism. When the mesh motion is explicitly imposed through the ALE formulation, a slight improvement can be observed. In particular, the u_x field exhibits a weak but noticeable signature of the fluid–structure interaction, indicating that the imposed kinematics provide additional consistency. Nevertheless, even in this case the overall temporal evolution remains qualitatively inaccurate, and the dominant dynamical features are not faithfully reproduced. This behaviour suggests that the observed limitations are not due to uncontrolled error growth, but rather to an insufficient representation of the coupled fluid–structure dynamics within the learned surrogate. In the configuration where the displacement field $\mathbf{d}$ is predicted rather than imposed, the model must learn not only the fluid evolution but also the structural motion that directly affects the domain geometry. This increases the dimensionality and coupling of the learned dynamical system. The difficulty may be attributed to the increased complexity of the fluid–structure interaction problem, where accurate prediction requires capturing both the boundary-induced velocity gradients and their feedback on the global flow evolution. In such settings, autoregressive error accumulation and limited long range information propagation can further hinder the model’s ability to reproduce coherent dynamical patterns.

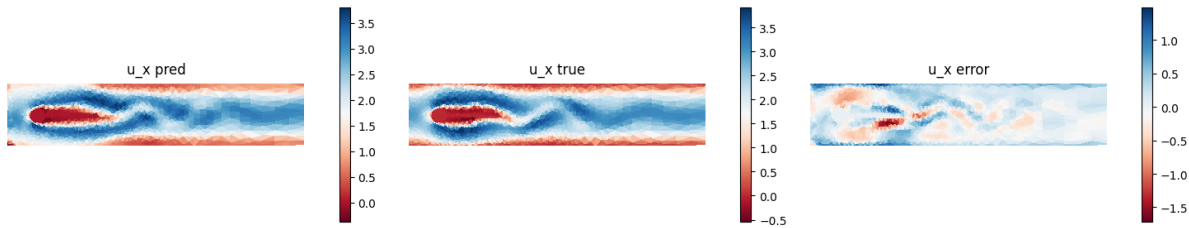


Figure 6.13: Predicted, L_2 error and OpenFOAM results for the velocity component along the x -direction (u_x) in the supervised- d configuration.

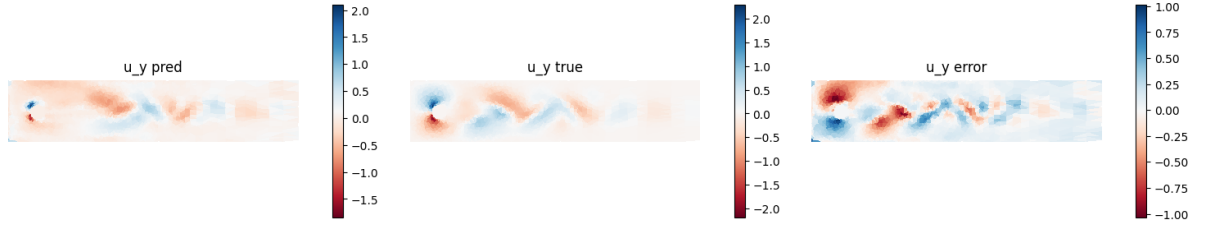


Figure 6.14: Predicted, L_2 error and OpenFOAM results for the velocity component along the y -direction (u_y) in the supervised- d configuration.

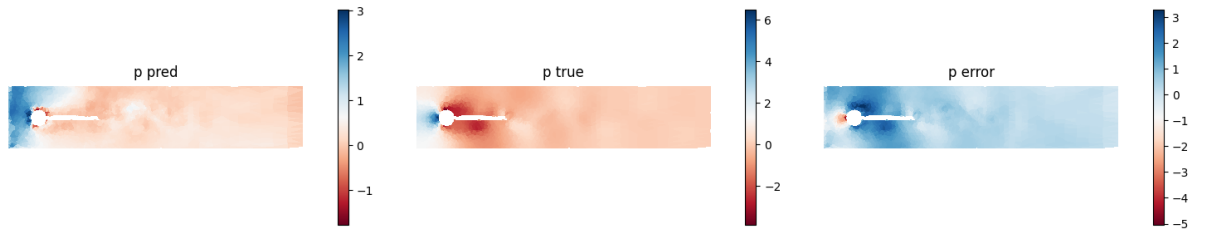


Figure 6.15: Predicted, L_2 error and OpenFOAM results for the pressure field (p) in the supervised- d configuration.

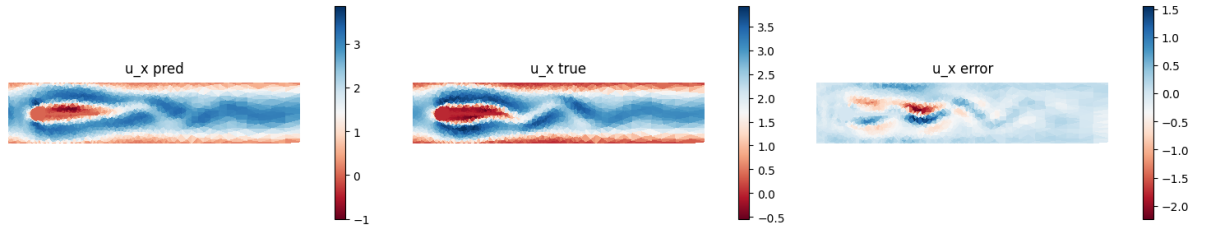


Figure 6.16: Predicted, L_2 error and OpenFOAM results for the velocity component along the x -direction (u_x) in the imposed ALE configuration.

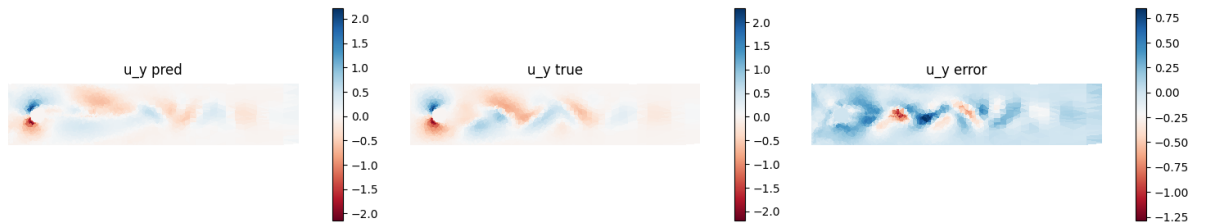


Figure 6.17: Predicted, L_2 error and OpenFOAM results for the velocity component along the y -direction (u_y) in the imposed ALE configuration.

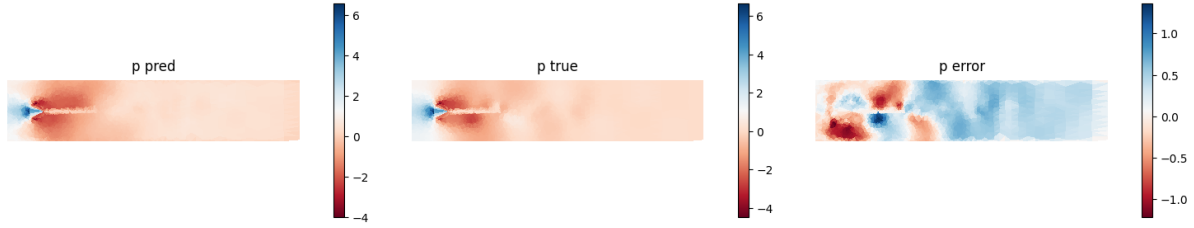


Figure 6.18: Predicted, L_2 error and OpenFOAM results for the pressure field (p) in the imposed ALE configuration.

6.4. Discussion and concluding remarks

The analysis presented in this chapter highlights a clear contrast between the supervised and partially unsupervised learning regimes in the context of fluid–structure interaction. When full supervision is available, the proposed MeshGraphNet framework proves capable of accurately learning the coupled fluid–solid dynamics over autoregressive rollouts, preserving the main physical flow features, including vortex shedding and structural oscillations. These results indicate that the graph-based formulation, together with the ALE geometric treatment, provides a consistent and reliable surrogate of the reference FSI solution in the fully supervised setting.

In contrast, the partially unsupervised configurations reveal significant limitations. Although the prediction errors remain bounded and do not exhibit numerical divergence, the qualitative structure of the flow is not correctly reproduced, and the dynamical coupling between fluid and structure is only weakly captured. This suggests that the difficulty lies in learning the coupled fluid–structure dynamics when supervision is reduced and geometric evolution must be inferred indirectly.

Overall, these findings indicate that, while supervised learning provides a solid and effective baseline for FSI surrogate modeling, extending the approach toward reduced-supervision or fully physics-driven regimes remains an open challenge.

7 | Scalability Limits of Message Passing Graph Neural Networks

In this section, we aim to investigate the accuracy of the MeshGraphNet when the spatial resolution of the computational domain is increased. To this end, we considered again the configuration described in Section 5.4, and progressively refined the mesh. In particular, the number of nodes was increased from 1574 (see Figure 5.16) to 8916 nodes (Figure 7.1).

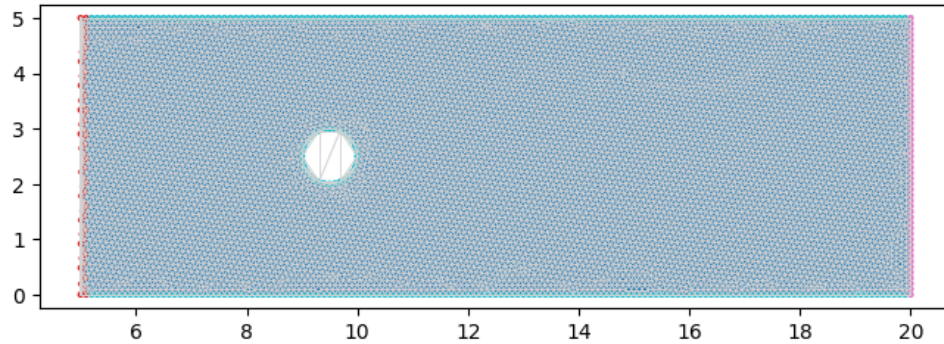


Figure 7.1: The new refined mesh.

In particular, the finer mesh coincides with the one employed by the OpenFOAM solver, while the coarser mesh is constructed by sampling a subset of its nodes.

From a numerical viewpoint, a finer discretization would normally be expected to improve accuracy, since the flow fields are sampled with higher spatial resolution. Surprisingly, the opposite behaviour was observed: the prediction error did not decrease, but increase.

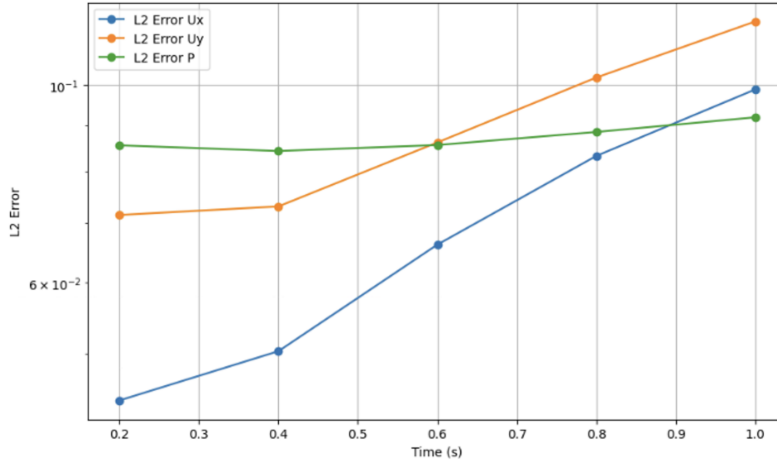


Figure 7.2: L2 error over the time for the finer mesh, for five timestep.

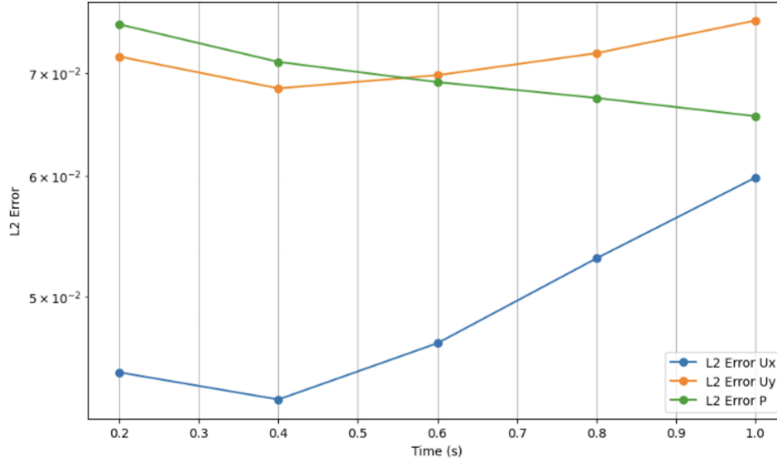


Figure 7.3: L2 error over the time for the coarser mesh, for five timestep.

Figure 7.2 and Figure 7.3 show the temporal evolution of the L_2 error for the two configurations, namely the finer mesh and the coarser mesh, respectively. As can be seen, in the finer case the error starts from a slightly larger value and increases more rapidly over time.

However, the most critical aspect emerges when inspecting the actual predictions at the final rollout step ($t = 5$). Despite the relatively moderate average L_2 error, the qualitative behaviour of the solution deteriorates significantly on the finer mesh (see Figures 7.4–7.6, which report the predicted and reference fields at $t = 5$). In particular, as visible in Figure 7.4, the curvature of the wake is not captured, especially in the region immediately downstream of the obstacle; instead, the velocity field progressively diffuses towards zero from the first to the last timestep, rather than preserving the expected downstream advection of the wake.

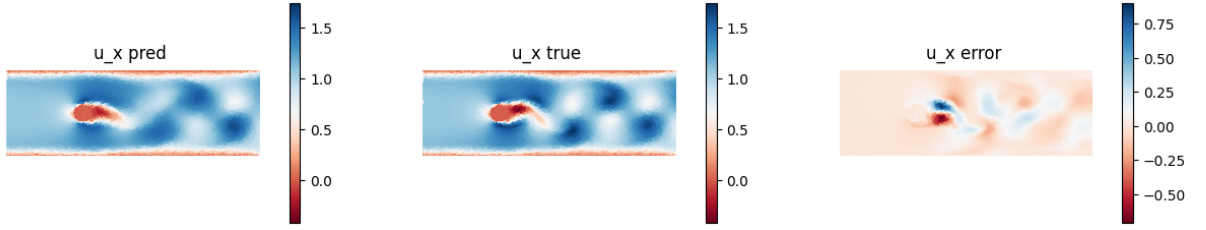


Figure 7.4: Comparison between MeshGraphNet predictions and OpenFOAM reference fields for the axes-x velocity for the finer mesh at the last timestep.

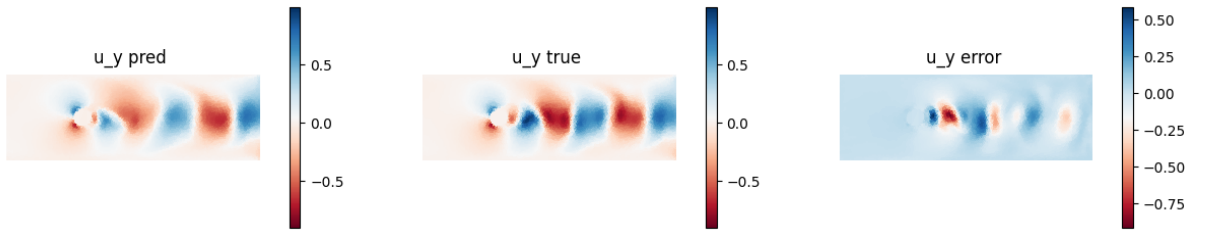


Figure 7.5: Comparison between MeshGraphNet predictions and OpenFOAM reference fields for the axes-y velocity for the finer mesh at the last timestep.

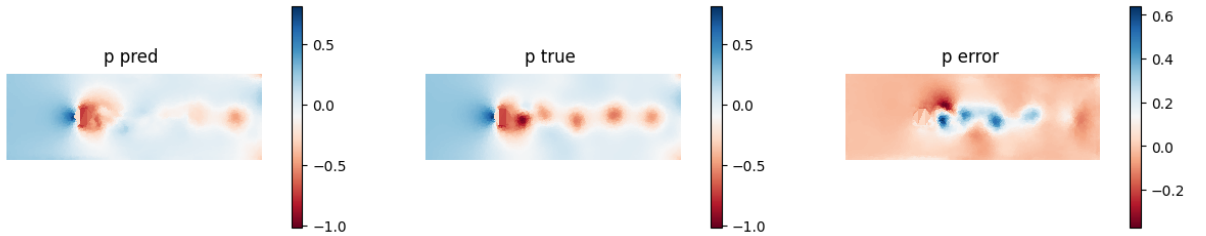


Figure 7.6: Comparison between MeshGraphNet predictions and OpenFOAM reference fields for the pressure for the finer mesh at the last timestep.

In contrast, in the coarser-mesh case, although the L_2 error also grows in time, the model is still able to reproduce the characteristic motion and deformation of the wake more reliably (See Figure 7.7, Figure 7.8, Figure 7.9) . This suggests that the finer discretization exacerbates the difficulties of the model in propagating coherent long-range information, ultimately leading to poorer qualitative predictions.

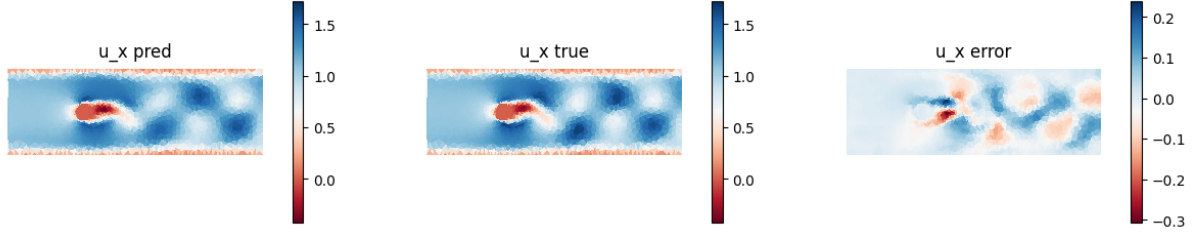


Figure 7.7: Comparison between MeshGraphNet predictions and OpenFOAM reference fields for the axes-x velocity for the coarser mesh at the last timestep..

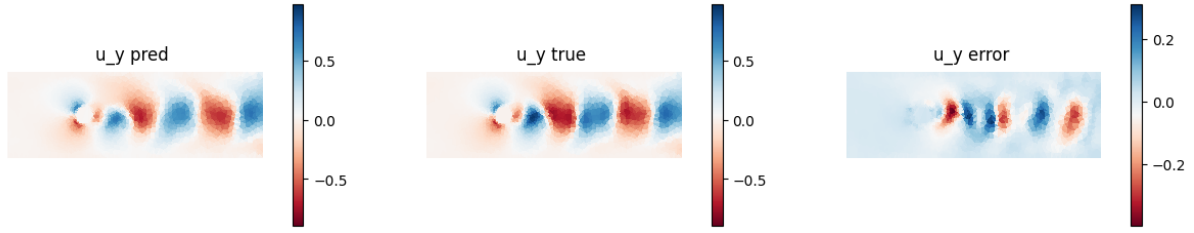


Figure 7.8: Comparison between MeshGraphNet predictions and OpenFOAM reference fields for the axes-y velocity for the coarser mesh at the last timestep..

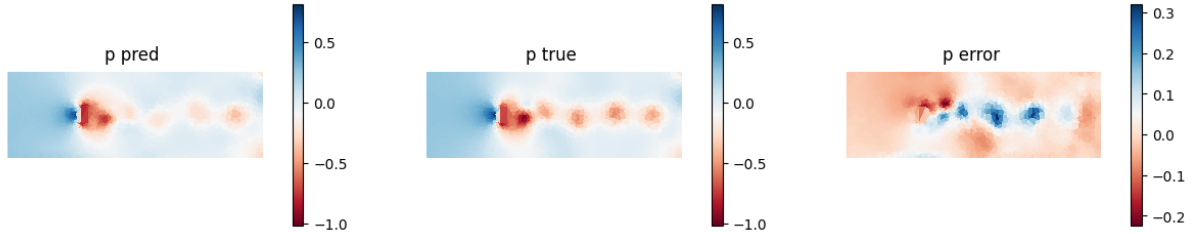


Figure 7.9: Comparison between MeshGraphNet predictions and OpenFOAM reference fields for the pressure for the coarser mesh at the last timestep..

This counterintuitive result aligns with known limitations of graph neural networks in the presence of large graphs requiring long-range information propagation. As highlighted in previous work [68], GNNs suffer from a bottleneck effect caused by the need to compress an exponentially growing amount of information into fixed-size node embeddings. This phenomenon, referred to as over-squashing, prevents effective propagation of messages when the number of nodes (and thus the receptive-field size) increases. In practice, signals originating from distant regions of the graph are squashed into a vector of limited dimensionality and cannot be transmitted faithfully through the message-passing layers.

In the present study, this behaviour must be interpreted in relation to the adopted learning regime. The degradation observed on refined meshes primarily affects the physics-

informed configuration, in which the model is required to reconstruct the global flow dynamics from spatially distributed residual constraints rather than from dense local supervision. In such a setting, accurate prediction relies on the propagation of physically meaningful information across long graph distances, making the model inherently sensitive to over-squashing effects.

By contrast, in the fully supervised regime the learning signal is strongly localized, since target states are directly provided at each node. As a consequence, the reconstruction of the solution does not depend on long-range message propagation to the same extent, and the practical impact of over-squashing is significantly reduced. The scalability limitation highlighted in this section should therefore be interpreted mainly as a constraint of the quasi-unsupervised formulation rather than as an intrinsic failure of the MeshGraphNet architecture under full supervision.

Refining the mesh substantially increases the number of nodes between any two physically interacting regions of the flow. As a consequence, the model is required to propagate information through much longer paths. According to the theoretical and empirical findings reported in [68], this dramatically amplifies over-squashing effects, making it increasingly difficult for the network to capture coherent long-range interactions such as the formation and evolution of vortices in the wake.

Therefore, the degradation in accuracy observed when increasing the mesh resolution is consistent with the over-squashing phenomenon: enlarging the graph makes it harder for the GNN to learn and preserve the relevant dynamics, despite the underlying physics being sampled more finely.

At the same time, these results highlight a practical advantage of MeshGraphNet when compared to traditional CFD solvers such as OpenFOAM. In the present configuration, MeshGraphNet is able to learn meaningful flow dynamics using a relatively coarse mesh, whereas a solver such as OpenFOAM would not converge, or would produce unreliable solutions, at such low spatial resolutions. This indicates that MeshGraphNet can operate effectively in regimes where classical solvers require significantly finer discretizations to remain stable, even though its performance may still degrade when the graph becomes excessively large.

8 | Conclusions

This thesis investigated a quasi-unsupervised deep learning strategy for incompressible fluid flows on irregular meshes, combining the MeshGraphNet architecture with discrete physical constraints derived from the governing equations. The objective extended beyond the reproduction of reference CFD fields and focused instead on robustness, data efficiency, temporal stability, and generalization across progressively more complex physical regimes.

The results obtained in laminar channel flow demonstrate that the hybrid grey-box formulation is able to recover physically meaningful solutions with accuracy comparable to fully supervised learning, even when supervision is spatially restricted. In this regime, residual based constraints effectively regularize the solution and preserve the expected physical structure of velocity and pressure fields. The only necessary external information concerns the pressure reference, whose weak anchoring resolves the intrinsic gauge indeterminacy of incompressible flows without altering the governing equations.

When obstacle induced wake dynamics are introduced, the physics-informed configuration shows particularly encouraging behaviour. While purely supervised learning achieves low training error, it does not systematically provide improved robustness. In contrast, the partially unsupervised strategy maintains competitive accuracy and demonstrates the ability to reproduce different geometric configurations with performance comparable to the supervised baseline. This result represents one of the most relevant positive outcomes of the thesis, as it indicates that the incorporation of physical constraints can enhance generalization within meaningful variations of the flow configuration.

The fluid–structure interaction case provides a clearer separation between what is currently achievable and what remains challenging. Under full supervision, the coupled dynamics are reproduced with satisfactory stability and physically consistent long-horizon rollouts, confirming that graph-based neural surrogates can approximate strongly coupled moving-boundary systems when sufficient data are available. However, the partially unsupervised configurations reveal that enforcing physics only through residual constraints is not yet sufficient to recover the correct qualitative coupling between fluid and structure. Importantly, this limitation is not associated with numerical divergence but with an

incomplete representation of the underlying dynamics, indicating that the main difficulty lies in the physics informed formulation rather than in the supervised learning paradigm.

A realistic assessment of computational cost is also essential. The original motivation for surrogate modeling lies in the high computational expense of CFD simulations. The present study shows that, although inference with a trained network is fast, the overall workflow still requires a non negligible amount of CFD-generated data and long training times. In practice, the total training cost can become substantial when accounting for both data generation and network optimization. Consequently, the computational benefit of the surrogate model is not immediate, but depends on its ability to generalize across multiple prediction scenarios. Taken together, these observations outline a balanced perspective. Quasi-unsupervised deep learning surrogates demonstrate clear potential: they improve robustness in relevant regimes, enable competitive predictions in geometrically varied obstacle configurations, and successfully reproduce complex coupled dynamics under full supervision. Despite these advances, the broader applicability of the approach is still conditioned by the difficulty of accurately capturing coupled multi-physics dynamics under limited supervision, achieving genuine generalization across domain topologies, and attaining a clear computational advantage over traditional simulation workflows.

Future progress will likely require tighter integration between physical modeling and neural architectures, broader and more diverse training distributions, and learning strategies capable of extracting transferable structure rather than interpolating within observed solution manifolds. Advances along these directions are necessary for transforming data-driven surrogates from promising research tools into practically reliable alternatives to traditional CFD solvers.

Bibliography

- [1] H. K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics*. Pearson, Harlow, UK, 2007.
- [2] C. Günther and S. Krajnović. Aerodynamic shape optimization of road vehicles using cfd. *Journal of Wind Engineering and Industrial Aerodynamics*, 171:35–45, 2017.
- [3] M. J. Churchfield and S. Lee. Large eddy simulation of wind turbine wakes using openfoam. *Journal of Physics: Conference Series*, 854(1):012033, 2017. doi: 10.1088/1742-6596/854/1/012033.
- [4] Y. Tominaga and T. Stathopoulos. Cfd simulation of pollutant dispersion in an urban environment. *Atmospheric Environment*, 79:716–730, 2013.
- [5] Y. Bazilevs, M. Hsu, and Y. Zhang. Fluid–structure interaction simulations of blood flow in patient-specific arteries using openfoam. *Computers & Fluids*, 141:128–145, 2016.
- [6] D. A. Nield and A. Bejan. *Convection in Porous Media*. Springer, New York, NY, 3rd edition, 2013.
- [7] R. Rocchetta, M. Broggi, and B. Sudret. Uncertainty quantification in computational fluid dynamics using openfoam. *Computer Methods in Applied Mechanics and Engineering*, 352:105–127, 2019.
- [8] J. B. Nagel and B. Sudret. Efficient uncertainty quantification and sensitivity analysis of cfd models using polynomial chaos. *International Journal for Numerical Methods in Fluids*, 89(5):187–213, 2019.
- [9] S. B. Pope. *Turbulent Flows*. Cambridge University Press, Cambridge, UK, 2000.
- [10] M. J. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53(3):484–512, 1984.
- [11] A. Quarteroni, A. Manzoni, and F. Negri. *Reduced Basis Methods for Partial Differential Equations*. Springer, Cham, Switzerland, 2015.

- [12] P. J. Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics*, 656:5–28, 2010. doi: 10.1017/S0022112010001217.
- [13] S. L. Brunton, B. R. Noack, and P. Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020. doi: 10.1146/annurev-fluid-010719-060214.
- [14] K. Duraisamy, G. Iaccarino, and H. Xiao. Turbulence modeling in the age of data. *Annual Review of Fluid Mechanics*, 51:357–377, 2019. doi: 10.1146/annurev-fluid-010518-040547.
- [15] N. Thuerey, K. Weißenow, L. Prantl, and X. Hu. Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA Journal*, 58(1):25–36, 2020. doi: 10.2514/1.J058291.
- [16] X. Guo, W. Li, and F. Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 481–490, San Francisco, CA, USA, 2016. ACM. doi: 10.1145/2939672.2939738. August 13–17, 2016.
- [17] J. Kim and C. Lee. Deep unsupervised learning of turbulence for inflow generation at various reynolds numbers. *Journal of Computational Physics*, 406:109216, 2020. doi: 10.1016/j.jcp.2019.109216.
- [18] Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. M. Stuart, and A. Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, Online, 2021. URL <https://arxiv.org/abs/2010.08895>. 3–7 May 2021.
- [19] Y. Bar-Sinai, S. Hoyer, J. Hickey, and M. P. Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, 2019. doi: 10.1073/pnas.1814058116.
- [20] M. Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *Journal of Machine Learning Research*, 19:1–24, 2018.
- [21] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. doi: 10.1016/j.jcp.2018.10.045.
- [22] G. E. Karniadakis, Y. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang.

- Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021. doi: 10.1038/s42254-021-00314-5.
- [23] N. Wandel, M. Weinmann, M. Neidlin, and R. Klein. Spline-pinn: Approaching pdes without data using fast, physics-informed hermite-spline cnns, 2022. URL <https://arxiv.org/abs/2109.07143>.
- [24] S. Wang, Y. Teng, and P. Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022. doi: 10.1016/j.jcp.2021.110768.
- [25] A. S. Krishnapriyan, A. Gholami, S. Zhe, R. M. Kirby, and M. W. Mahoney. Characterizing possible failure modes in physics-informed neural networks, 2021. URL <https://arxiv.org/abs/2109.01050>.
- [26] J. Ling, A. Kurzawski, and J. Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016. doi: 10.1017/jfm.2016.615.
- [27] S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, and S. Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2):525–545, 2019. doi: 10.1007/s00466-019-01740-0.
- [28] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. doi: 10.1109/MSP.2017.2693418.
- [29] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations (ICLR)*, Online, 2021. URL <https://arxiv.org/abs/2010.03409>. 3–7 May 2021.
- [30] A. Sanchez-Gonzalez, V. Bapst, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 8459–8468, Virtual, 2020. 13–18 July 2020.
- [31] P. W. Battaglia, J. B. Hamrick, V. Bapst, et al. Relational inductive biases, deep learning, and graph networks. *arXiv*, 2018. doi: 10.48550/arXiv.1806.01261.
- [32] B. Fornberg and N. Flyer. *A Primer on Radial Basis Functions with Applications to the Geosciences*. SIAM, Philadelphia, PA, 2015.

- [33] G. K. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, Cambridge, 1967.
- [34] D. S. Chandrasekharaiah and L. Debnath. Equations of fluid mechanics. In *Continuum Mechanics*, chapter 10. Academic Press, San Diego, CA, 1994. doi: 10.1016/B978-0-12-167880-7.50015-0.
- [35] R. Temam. *Navier-Stokes Equations: Theory and Numerical Analysis*. American Mathematical Society, Providence, RI, 2001.
- [36] O. A. Ladyzhenskaya. *The Mathematical Theory of Viscous Incompressible Flow*. Gordon and Breach, New York, NY, 1969.
- [37] C. Foias, O. Manley, R. Rosa, and R. Temam. *Navier-Stokes Equations and Turbulence*. Cambridge University Press, Cambridge, UK, 2001.
- [38] OpenFOAM. The open source cfd toolbox, 2025. URL <https://www.openfoam.com>. Accessed: 2025-12-08.
- [39] F. Moukalled, L. Mangani, and M. Darwish. *The Finite Volume Method in Computational Fluid Dynamics*. Springer, Cham, Switzerland, 2016.
- [40] J. H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer, Berlin, Germany, 2002.
- [41] C. Geuzaine and J. Remacle. *Gmsh Reference Manual*. Gmsh Development Team, 2023. URL <https://gmsh.info/doc/texinfo/gmsh.html>.
- [42] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, MA, 2016.
- [43] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. doi: 10.1038/nature14539.
- [44] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, pages 1097–1105, Lake Tahoe, NV, USA, 2012. 3–6 December 2012.
- [45] S. Hooker. The hardware lottery. *Communications of the ACM*, 63(12):58–65, 2020. doi: 10.1145/3399061.
- [46] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, and D. Cohen-Or. Meshcnn: A network with an edge. *ACM Transactions on Graphics*, 38(4):90:1–90:12, 2019. doi: 10.1145/3306346.3322959.

- [47] F. de A. Belbute-Peres, T. D. Economon, and J. Z. Kolter. Combining differentiable pde solvers and graph neural networks for fluid flow prediction. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 2402–2411, Virtual, 2020. 13–18 July 2020.
- [48] Z. Xiang, W. Peng, W. Yao, X. Liu, and X. Zhang. Solving spatiotemporal partial differential equations with physics-informed graph neural network. *Applied Soft Computing*, 155:111437, 2024. doi: 10.1016/j.asoc.2024.111437.
- [49] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 807–814, Haifa, Israel, 2010. 21–24 June 2010.
- [50] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization, 2016. URL <https://arxiv.org/abs/1607.06450>.
- [51] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical Review Letters*, 120(2):024102, 2018. doi: 10.1103/PhysRevLett.120.024102.
- [52] A. Karpatne, G. Atluri, J. H. Faghmous, M. Steinbach, A. Banerjee, A. R. Ganguly, S. Shekhar, N. F. Samatova, and V. Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on Knowledge and Data Engineering*, 29(10):2318–2331, 2017. doi: 10.1109/TKDE.2017.2720168.
- [53] J. Willard, X. Jia, S. Xu, M. Steinbach, and V. Kumar. Integrating physics-based modeling with machine learning: A survey, 2020. URL <https://arxiv.org/abs/2003.04919>.
- [54] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: A survey. *Journal of Machine Learning Research*, 18(153):1–43, 2018.
- [55] L. F. Richardson. The approximate arithmetical solution by finite differences of physical problems involving differential equations. *Philosophical Transactions of the Royal Society A*, 210:307–357, 1911.
- [56] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, 2006.
- [57] H. Gao, L. Sun, and J. Wang. Physics-informed graph neural networks. *Journal of Computational Physics*, 469:111510, 2023. doi: 10.1016/j.jcp.2022.111510.

- [58] J. Brandstetter, D. Worrall, and M. Welling. Message passing neural pde solvers. In *International Conference on Learning Representations (ICLR)*, Virtual, 2022. URL <https://arxiv.org/abs/2202.03376>. 25–29 April 2022.
- [59] S. Le Borne and W. Leinen. Guidelines for rbf-fd discretization: Numerical experiments on the interplay of a multitude of parameter choices. *Journal of Scientific Computing*, 95(8):1–40, 2023. doi: 10.1007/s10915-023-02123-7.
- [60] P. Marcandelli, N. Mathur, S. Markidis, M. Siena, and S. Mariani. Unsupervised physics-informed operator learning through multi-stage curriculum training, 2026. URL <https://arxiv.org/abs/2602.02264>.
- [61] E. H. Dowell et al. *A Modern Course in Aeroelasticity*. Kluwer Academic Publishers, Dordrecht, Netherlands, 2001.
- [62] A. Quarteroni, M. Tuveri, and A. Veneziani. Mathematical modelling of the cardiovascular system. In *Handbook of Numerical Analysis*. Elsevier, 2000.
- [63] L. Formaggia, A. Quarteroni, and A. Veneziani. *Cardiovascular Mathematics*. Springer, Milan, Italy, 2009.
- [64] J. Donea and A. Huerta. *Arbitrary Lagrangian–Eulerian Methods*. Wiley, Chichester, UK, 2004.
- [65] S. Turek and J. Hron. Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow. In Hans-Joachim Bungartz and Michael Schäfer, editors, *Fluid–Structure Interaction: Modelling, Simulation, Optimisation*, pages 371–385. Springer, Berlin, Germany, 2006.
- [66] solids4foam. Hron-turek fsi3 tutorial, 2024. URL <https://www.solids4foam.com/tutorials/more-tutorials/fluid-solid-interaction/HronTurekFsi3.html>. Accessed: 2026-02-11.
- [67] O. C. Zienkiewicz, R. L. Taylor, and J. Z. Zhu. *The Finite Element Method: Its Basis and Fundamentals*. Elsevier Butterworth-Heinemann, Oxford, UK, 6th edition, 2005.
- [68] U. Alon and E. Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations (ICLR)*, Online, 2021. URL <https://arxiv.org/abs/2006.05205>. 3–7 May 2021.

List of Figures

1.1	Schematic representation of a grey-box modeling approach combining high-fidelity CFD data and physical constraints within a graph neural network framework.	4
2.1	Example of a microfluidic channel device with rigid confining walls. The present study considers an idealised two dimensional approximation of a confined channel, abstracting from three dimensional effects and microscale specific phenomena.	7
2.2	Qualitative comparison between laminar and turbulent flow regimes: laminar flow (top) and turbulent flow (bottom).	11
2.3	Schematic velocity profile of plane Poiseuille flow between two parallel plates.	12
2.4	The configuration of the channel flow	15
2.5	Structured mesh generated using blockMesh for a channel flow configuration with Cartesian geometry.	16
2.6	Unstructured mesh generated using Gmsh for a channel flow configuration with a curved obstacle.	17
2.7	Unstructured mesh generated using Gmsh for a channel flow configuration with an airfoil obstacle.	17
3.1	Illustration of a typical feed-forward neural network architecture, in which artificial neurons (units) are arranged in successive layers and connected to adjacent layers.	22
3.2	Comparison between mesh-based predictions obtained using a graph-based representation (left) and a grid-based convolutional baseline (right). The graph-based approach enables a more localized and adaptive resolution of the flow field, allowing finer-scale structures to be captured more accurately [29].	24
3.3	Diagram of Graph Neural Network architecture [48].	24
3.4	Schematic representation of the encoder architecture	26
3.5	Example of message passing and aggregation in a graph neural network [29].	28

3.6	Example of stencil construction for RBF-FD discretization [48].	35
3.7	Example of a node distribution and the corresponding fill distance $h_{X,\Omega}$ and separation distance q_X	37
4.1	Two-dimensional computational domain of the channel flow problem. The rectangular domain has dimensions $2\text{ m} \times 1\text{ m}$ and is described in the Carte- sian coordinate system (x, y)	42
4.2	Velocity magnitude at the final simulation time, showing the fully devel- oped parabolic Poiseuille profile.	44
4.3	Pressure field at the final simulation time, showing the expected linear streamwise gradient.	45
4.4	Graph representation of the structured lattice used for the channel domain. Each node is connected to its immediate horizontal and vertical neighbors through bidirectional edges, forming a regular grid topology.	46
4.5	Schematic representation of the MeshGraphNet architecture adopted in this study. The input graph is first projected into a latent space through node and edge encoders. Information is then propagated through l message- passing layers before being mapped to the physical output fields by a node- wise decoder. The predicted fields are finally used to compute the training loss.	49
4.6	Predicted, L2 error and OpenFOAM results for the velocity component along the x -direction (u_x).	52
4.7	Predicted, L2 error and OpenFOAM results for the velocity component along the y -direction (u_y).	52
4.8	Predicted, L2 error and OpenFOAM results of the pressure (p) for the test case.	52
4.9	Predicted, L2 error and OpenFOAM results for the velocity component along the x -direction (u_x) for the finer mesh.	55
4.10	Predicted, L2 error and OpenFOAM results for the velocity component along the y -direction (u_y) for the finer mesh.	55
4.11	Predicted, L2 error and OpenFOAM results for the pressure for the finer mesh.	55
4.12	Division of the domain in supervised mask (20%) and unsupervised mask (80%)	57
4.13	Velocity loss over epochs	58
4.14	Predicted, L2 error and OpenFOAM results for the velocity component along the x -direction (u_x).	59

4.15	Predicted, L2 error and OpenFOAM results for the velocity component along the y -direction (u_y).	59
4.16	Predicted, L2 error and OpenFOAM results for the pressure (p).	59
4.17	Velocity loss over training epochs.	62
4.18	Predicted, L2 error and OpenFOAM results for the velocity component along the x -direction (u_x).	63
4.19	Predicted, L2 error and OpenFOAM results for the velocity component along the y -direction (u_y).	63
4.20	Predicted, L2 error and OpenFOAM results for the pressure (p). The pressure field exhibits oscillations and offset drift due to the indeterminacy of the gauge pressure.	63
4.21	Fully unsupervised case with boundary pressure anchoring. Velocity along x -axes.	64
4.22	Fully unsupervised case with boundary pressure anchoring. Velocity along y -axes.	64
4.23	Fully unsupervised case with boundary pressure anchoring. Prediction of the pressure.	65
5.1	$Re = 20$ — Steady laminar wake with a stable recirculation zone.	71
5.2	$Re = 200$ — Transitional regime with periodic vortex shedding.	71
5.3	$Re = 2000$ -Transitional regime with irregular vortex shedding.	71
5.4	$Re = 5 \times 10^5$ — Fully turbulent wake.	72
5.5	Comparison of the evolution of the velocity training loss across different phases of the multi-phase schedule, across one single phase and across the multi-phase without the restart.	73
5.6	Comparison of the evolution of the velocity training loss across different phases of the multi-phase schedule, across one single phase and across the multi-phase without the restart, for the last 1000 epochs.	73
5.7	Schematic illustration of the temporal rollout strategy. The model is first trained using short autoregressive rollouts (e.g., one timestep), and the prediction horizon is progressively increased after a fixed number of epochs, until the full rollout length is reached.	76
5.8	Structured mesh used for the $Re = 20$ laminar obstacle flow. The nodes are colored according to the categorical labels, which assigns a distinct node type to each region of the computational domain.	78

5.9	Domain subdivision for the partially unsupervised case. Supervised regions (blue) surround the obstacle and boundaries, while the interior flow is driven by physical constraints.	79
5.10	Predicted velocity along x-direction field (u_x) compared with the OpenFOAM reference.	80
5.11	Predicted velocity along y-direction field (u_y) compared with the OpenFOAM reference.	80
5.12	Predicted pressure field (p) compared with the OpenFOAM reference. . . .	80
5.13	Predicted velocity along x-direction field (u_x) compared with the OpenFOAM reference.	81
5.14	Predicted velocity along x-direction field (u_y) compared with the OpenFOAM reference.	82
5.15	Predicted pressure field (p) compared with the OpenFOAM reference. . . .	82
5.16	Unstructured mesh used for the $Re = 200$ obstacle flow. The nodes are colored according to the categorical labels, which assigns a distinct node type to each region of the computational domain.	83
5.17	The subdivision of the nodes in supervised and unsupervised mask. The blue one is the supervised mask and the orange one is the unsupervised mask.	84
5.18	Predicted velocity along axes-x for the $Re = 200$ case. The network correctly reproduces the periodic vortex shedding in the wake region.	86
5.19	Predicted velocity along axes-y for the $Re = 200$ case. The network correctly reproduces the periodic vortex shedding in the wake region.	86
5.20	Predicted pressure for the $Re = 200$ case. The network correctly reproduces the periodic vortex shedding in the wake region.	86
5.21	L2 error over the time, for the case test and trained with 30 timesteps . . .	87
5.22	L2 error over the time, for the first two periods trained and tested (periodic regime).	88
5.23	L2 error over the time, tested only on the last unseen period (periodic regime).	88
5.24	Behaviour of the velocity train loss for the supervised and for the unsupervised case.	89
5.25	L2 error over the time for the supervised and the unsupervised cases. . . .	90
5.26	Example of the trajectories of the dataset.	91
5.27	Example of the trajectories of the dataset.	91
5.28	Example of the trajectories of the dataset.	91

5.29	Comparison between the validation and the train loss of the velocity for our model.	92
5.30	Comparison between the validation and the train loss of the velocity for the supervised case.	92
5.31	A mean of the Mean Squared Error over the 8 test trajectories. Comparison between supervised case and our model.	93
5.32	Configuration of the two-obstacle case in OpenFOAM. The figure shows the magnitude of the velocity field.	94
5.33	Predicted velocity along x-direction field compared with the OpenFOAM reference at the last timestep ($t = 5$).	94
5.34	Predicted velocity along y-direction field compared with the OpenFOAM reference at the last timestep ($t = 5$).	95
5.35	Predicted pressure field compared with the OpenFOAM reference at the last timestep ($t = 5$).	95
5.36	Mean Squared Error over the time.	95
5.37	The subdivision of the nodes in supervised and unsupervised masks. The green one is the supervised mask and the grey one is the unsupervised. . .	97
5.38	Predicted velocity along axes-x for the $Re = 2000$ case. The network correctly reproduces the vortex in the wake region.	97
5.39	Predicted velocity along axes-x for the $Re = 2000$ case. The network correctly reproduces the vortex in the wake region.	98
5.40	Predicted velocity along axes-x for the $Re = 2000$ case. The network correctly reproduces the vortex in the wake region.	98
5.41	The mean of the L2 error on the computational domain over the time. . . .	98
5.42	Structured mesh used for the turbulent channel-with-obstacle case. The grid is denser in the wake region to better resolve vortex shedding and turbulent fluctuations.	99
5.43	Comparison between MeshGraphNet predictions and OpenFOAM reference fields of the velocity along x-direction at the timestep $t = 1$	101
5.44	Comparison between MeshGraphNet predictions and OpenFOAM reference fields of the velocity along y-direction at the timestep $t = 1$	101
5.45	Comparison between MeshGraphNet predictions and OpenFOAM reference fields of the pressure at the timestep $t = 1$	101
5.46	Evolution of the mean L_2 error over time during autoregressive rollout for the test of the turbulent case ($Re = 2 \times 10^6$). The error grows linearly with time due to temporal compounding.	102

6.1	Schematic representation of the Hron-Turek FSI3 configuration.	106
6.2	Velocity magnitude field of the Hron-Turek FSI3 configuration obtained with OpenFOAM. The vortex shedding behind the cylinder excites oscillations of the attached elastic beam.	107
6.3	Zoomed-in views of the elastic beam at two opposite displacement peaks within one oscillation cycle. The consistent color scale allows a direct qualitative comparison of the flow structure and beam deformation.	108
6.4	Computational mesh at the initial timestep of the Hron-Turek FSI3 configuration extracted from OpenFOAM. All mesh vertices are used as graph nodes in the supervised setting.	109
6.5	L2 error over the time of the velocity along x-direction, the velocity along y-direction, and the pressure	110
6.6	Predicted, L_2 error and OpenFOAM results for the velocity component along the x -direction (u_x) in the fully supervised configuration.	111
6.7	Predicted, L_2 error and OpenFOAM results for the velocity component along the y -direction (u_y) in the fully supervised configuration.	111
6.8	Predicted, L_2 error and OpenFOAM results for the pressure field (p) in the fully supervised configuration.	111
6.9	The subdivision of the nodes in supervised and unsupervised masks. The orange one is the supervised mask and the blue one is the unsupervised. . .	112
6.10	Node distribution of the coarsened mesh used for the FSI simulations, obtained from the original OpenFOAM geometry to reduce the computational complexity while preserving the main spatial features.	113
6.11	L2 error over the time in the supervised- d configuration	115
6.12	L2 error over the time in the imposed ALE configuration.	115
6.13	Predicted, L_2 error and OpenFOAM results for the velocity component along the x -direction (u_x) in the supervised- d configuration.	116
6.14	Predicted, L_2 error and OpenFOAM results for the velocity component along the y -direction (u_y) in the supervised- d configuration.	117
6.15	Predicted, L_2 error and OpenFOAM results for the pressure field (p) in the supervised- d configuration.	117
6.16	Predicted, L_2 error and OpenFOAM results for the velocity component along the x -direction (u_x) in the imposed ALE configuration.	117
6.17	Predicted, L_2 error and OpenFOAM results for the velocity component along the y -direction (u_y) in the imposed ALE configuration.	117
6.18	Predicted, L_2 error and OpenFOAM results for the pressure field (p) in the imposed ALE configuration.	118

7.1	The new refined mesh.	119
7.2	L2 error over the time for the finer mesh, for five timestep.	120
7.3	L2 error over the time for the coarser mesh, for five timestep.	120
7.4	Comparison between MeshGraphNet predictions and OpenFOAM reference fields for the axes-x velocity for the finer mesh at the last timestep. .	121
7.5	Comparison between MeshGraphNet predictions and OpenFOAM reference fields for the axes-y velocity for the finer mesh at the last timestep. .	121
7.6	Comparison between MeshGraphNet predictions and OpenFOAM reference fields for the pressure for the finer mesh at the last timestep.	121
7.7	Comparison between MeshGraphNet predictions and OpenFOAM reference fields for the axes-x velocity for the coarser mesh at the last timestep..	122
7.8	Comparison between MeshGraphNet predictions and OpenFOAM reference fields for the axes-y velocity for the coarser mesh at the last timestep..	122
7.9	Comparison between MeshGraphNet predictions and OpenFOAM reference fields for the pressure for the coarser mesh at the last timestep.. . . .	122

List of Tables

2.1	Typical flow regimes as a function of the Reynolds number for internal flows.	11
2.2	Boundary conditions used for the simplest test cases without obstacles.	18
2.3	Boundary conditions used for the cases involving obstacles and unsteady wake dynamics.	20
4.1	Summary of the supervised training setup for the MeshGraphNet model.	50
4.2	L_2 errors for velocity components and pressure.	51
4.3	L_2 errors for velocity components and pressure.	54
4.4	Mean L_2 errors for velocity and pressure fields in the partially unsupervised case ($U_{\text{in}} = 1.0$ m/s).	60
4.5	Comparison of mean L_2 errors with and without boundary pressure supervision in the fully unsupervised case ($U_{\text{in}} = 1.0$ m/s).	65
4.6	Comparison of mean L_2 errors for velocity and pressure fields under different supervision regimes.	66
5.1	Mean and variance of the L1 velocity error across different random seeds.	75
5.2	Mean L_2 errors for the laminar obstacle flow ($Re = 20$).	82
6.1	Parameters of the Hron–Turek FSI3 benchmark.	107

List of Symbols

Variable	Description	SI unit
$\mathbf{u}$	vector of the velocity	m s^{-1}
$\mathbf{u}_x$	velocity along the x-direction	m s^{-1}
$\mathbf{u}_y$	velocity along the y-direction	m s^{-1}
p	pressure	Pa
$\mathbf{d}$	structural displacement vector	m
ρ	fluid density	kg m^{-3}
μ	dynamic viscosity	Pa s
ν	kinematic viscosity ($\nu = \mu/\rho$)	$\text{m}^2 \text{s}^{-1}$
U_{in}	inlet velocity magnitude	m s^{-1}
L	characteristic length	m
t	time	s
Δt	time step	s
Re	Reynolds number ($U_{\text{in}} L / \nu$)	—
x_i	x-coordinate of point i	m
y_i	y-coordinate of point i	m
$\Delta \mathbf{u}_x$	velocity increment along the x-direction	m s^{-1}
$\Delta \mathbf{u}_y$	velocity increment along the y-direction	m s^{-1}
Δp	pressure increment	Pa

Acknowledgements

I would like to express my sincere gratitude to Professor Stefano Mariani, supervisor of this thesis, for the opportunity to work on a challenging and highly relevant problem at the intersection of CFD and artificial intelligence. His intellectual rigor, clarity of vision, and constant guidance have been fundamental throughout this work. This experience represented my first direct involvement in academic research, and his mentorship has played an important role in shaping the way I approach scientific questions.

I am also very grateful to Paolo Marcandelli for his constant support and collaboration. We worked closely throughout the project, discussing ideas, addressing challenges, and refining solutions together. His availability and thoughtful feedback were greatly appreciated and contributed meaningfully to the development of this thesis.

I would also like to thank Professor Francesco Costanzo of The Pennsylvania State University for his valuable insights and suggestions, which were essential for the development of the Fluid–Structure Interaction chapter.

