POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Aortic wall shear stress quantification from 4D flow data with physics informed neural networks

Tesi di Laurea Magistrale in
Mathematical Engineering - Ingegneria Matematica

Author: **Francesco Songia**

Student ID: 988635
Advisor: Dr. Stefano Pagani
Co-advisors: Dr. Francesco Regazzoni, Dr. Simone Saitta
Academic Year: 2022-23

Abstract

4D flow MRI provides non-invasive measurements of blood velocity in the ascending aorta. These velocity measurements enable patient-specific assessment of the wall shear stress (WSS), a widely adopted biomarker for characterizing and comprehending cardiovascular diseases. However, these recordings are usually affected by noise and do not have sufficient resolution to accurately reconstruct derived quantities like WSS.

The objective of this thesis is to develop a deep learning framework based on physics-informed neural networks (PINNs) to regularize and denoise 4D flow MRI data, enabling accurate WSS approximation. Well known physical laws can be considered as another source of information, as they described the flow evolution from a modelling point of view. This balance can be controlled by PINNs as they give the possibility to encode differential models into suitable terms of the loss function ensuring that the predicted velocity and pressure fields obey the prescribed physical laws.

An optimal configuration in terms of velocity reconstruction is found by testing the neural network on synthetic data that emulates real 4D flow measurements. A computational fluid dynamics (CFD) simulation is used to generate data and to evaluate the model performances on different synthetic test cases with an increasing noise level. The balance between loss terms and the imposition of the no-slip condition are the main features discussed when creating the models. Null wall velocity is obtained by adding one term in the loss function or, otherwise, by emulating the lifting procedure with an analytic function or with a further neural network representing the distance from the boundary.

In 3D test cases, we propose an estimate for wall shear stress starting from the predicted super-resolved velocity field. This biomarker is influenced by velocity profiles in the near wall region, and an accurate reconstruction of the related high velocity gradients remains one of the most challenging tasks in this work. We compared different strategies to address this complexity by forcing a steep velocity profile and by increasing the regularization of the governing physical laws near the boundaries. Finally, this methodology is applied to patient-specific 4D flow *in-vivo* data to estimate wall shear stress.

Keywords: physics informed neural networks, super-resolution, 4D flow MRI, wall shear stress

Abstract in lingua italiana

4D flow MRI fornisce misurazioni non-invadenti della velocità del sangue in un volume di interesse. Queste presentano dettagli specifici di un paziente che possono essere usati per calcolare lo sforzo sulla parete del vaso (WSS), per caratterizzare e comprendere malattie cardiovascolari. Tuttavia, queste misurazioni non hanno una risoluzione accettabile e, di conseguenza, non possono essere usate per calcolare rilevanti biomarcatori.
L'obiettivo di questa tesi è quello di sviluppare un ambiente di deep learning basato sulle reti neurali fisicamente informate (PINNs) per regolarizzare e rimuovere il rumore da dati 4D flow MRI. Ben note leggi fisiche possono essere considerate come un'altra sorgente di informazioni, dato che descrivono l'evoluzione del flusso da un punto di vista modellistico. Questo bilancio tra dati e teoria può essere controllato dalle PINNs in quanto hanno la possibilità di codificare modelli differenziali in alcuni dei termini della funzione obiettivo assicurando che i campi di velocità e pressione predetti rispettino le leggi fisiche prescritte. La configurazione ottimale in termini di ricostruzione di velocità viene trovata testando la rete neurale su dati sintetici che simulano reali misurazioni 4D flow. Una simulazione fluido-dinamica (CFD) è usata per generare questi dati e per valutare le prestazioni del modello sui diversi casi sintetici che presentano un livello crescente di rumore. Il bilancio tra i termini della funzione obiettivo e l'imposizione della condizione di non aderenza sono le principali caratteristiche discusse durante la creazione dei modelli. Una velocità nulla al bordo è ottenuta aggiungendo un termine nella funzione obiettivo o, altrimenti, simulando l'operazione di rilevamento con una funzione analitica o con un'ulteriore rete neurale rappresentante la distanza dal bordo.
Nei casi 3D, proponiamo una stima dello sforzo a parete partendo dalla raffinata velocità predetta. Questo biomarcatore è influenzato dai profili di velocità nella zona vicina al bordo, e un'accurata ricostruzione dei gradienti di velocità rimane uno dei compiti più difficili. Confrontiamo diverse strategie per affrontare questa complessità imponendo un ripido profilo di velocità e aumentando la rilevanza delle leggi fisiche vicino al bordo.

Parole chiave: reti neurali fisicamente informate, super risoluzione, 4D flow MRI, sforzo a parete

Contents

1 | Introduction

Over the past three decades, near-wall hemodynamics have received the utmost attention in cardiovascular fluid mechanics research. Clinical studies demonstrate its correlation with cardiovascular diseases [10, 39], and complex wall shear stress (WSS) patterns are related to disturbed blood flow dynamics in diseased arteries [1, 30].
In this work an accurate estimation of wall shear stress is carried out starting from blood velocity measurements in the aorta recorded through 4D flow MRI. These data are corrupted by several noise sources and they are affected by low spatial and temporal resolution, so they cannot be used to reliably compute precise biomarkers. Physics Informed Neural Networks (PINNs) address this problem by combining actual measurements of blood flow velocities with a modelling knowledge that describes through physical laws the blood behaviour. The resulting neural network proposes a super-resolved and regular velocity field and permits an accurate computation of velocity gradients through automatic differentiation, yielding to an estimate of wall shear stress.

The general structure and the main features of this analysis are reported in this introduction. Firstly, 4D flow MRI technique is described highlighting the acquisition procedure and the limitations encountered. PINNs recover these issues by proposing a super-resolved velocity, and the optimal neural network configuration is found working within a controlled framework build upon synthetic data. The latter emulate real measurements and, as they are generated from a CFD solution, there is a reference solution to evaluate performances. Finally, this methodology is applied on *in-vivo* data.
As the main objectives of this work, velocity and pressure fields are reconstructed improving the regularity and the resolution of the initial raw measurements and, in addition, WSS is estimated starting from the predicted velocity field.

1.1. 4D flow magnetic resonance imaging

4D flow magnetic resonance imaging (MRI) is used to obtain non-invasive patient-specific velocity measurements. However, they cannot be used to obtain reliable and accurate

biomarkers, as they suffer from limitations that will be investigated in this section. These data are, indeed, noisy and they do not have an adequate spatio-temporal resolution and many details are not captured.
The acquisition principle of 4D flow MRI is the same as the one for standard 2D PC-MRI. PC-MRI takes advantage of the direct relationship between blood flow velocity and the phase of the MR signal that is acquired during an MRI measurement. To eliminate unwanted background phase effects, two acquisitions with different velocity-dependent signal phase are typically needed to encode (using bipolar magnetic field gradients) and measure blood flow velocity along a single direction. Subtracting phase images from the two acquisitions removes background phase effects. The signal intensities in the resulting phase difference images are directly related to the blood flow velocity and can thus be used to visualize and quantify blood flow (Phase Contrast (PC) principle). PC-MRI encodes tissue velocity $\mathbf{v}(\mathbf{x}, t) \in \mathbb{R}^3$ at a spatial location x during cardiac phase t according to:

$$\rho_i(\mathbf{x}, t) = \rho_0(\mathbf{x}, t) exp \left(j\pi \frac{(\boldsymbol{\chi}\mathbf{v}(\mathbf{x}, t))_i}{\text{VENC}} \right), \tag{1.1}$$

where VENC is a manually set parameter determining the maximum velocity that can be recorded, $\rho_i = 0, ..., 3$ are the encoded magnitude and velocity components, and adopting a four-point velocity encoding, $\boldsymbol{\chi}$ is defined as:

$$\boldsymbol{\chi} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Hence, the measured tissue velocity component i is proportional to the phase shift of the reconstructed images ρ_i. Considering $\boldsymbol{\rho_{it}} \in \mathbb{R}^{N_r \times N_c \times N_s}$ a discretized complex PC image on a Cartesian grid $N_r \times N_c \times N_s$ corresponding to a cardiac phase t and velocity component i, the reconstructed image $\boldsymbol{H}(\boldsymbol{\rho_{it}}) \in \mathbb{R}^{N_r \times N_c \times N_s}$ can be modeled as:

$$\boldsymbol{H}(\boldsymbol{\rho_{it}}) = \boldsymbol{F^{-1}}(\boldsymbol{M}(\boldsymbol{F}(\boldsymbol{\rho_{it}}) + \varepsilon)),$$

where $\boldsymbol{F}$ is the Fourier transform, $\boldsymbol{M} \in \{0, 1\}^{N_r \times N_c \times N_s}$ defines the undersampling mask in *k-space*, and $\epsilon \in \mathbb{C}^{N_r \times N_c \times N_s}$ is the additive complex noise [37].
For cardiovascular applications, the 2D PC data is acquired over multiple cardiac cycles and measurements are synchronized with the cardiac cycle using ECG information. In 4D flow MRI, velocity is encoded along all three spatial dimensions throughout the cardiac cycle, thus providing a time-resolved 3D velocity field. As described above, quantita-

tive velocity measurements require two acquisitions and a subtraction for one-directional velocity encoding. For each time frame, four 3D raw datasets are collected to measure three-directional blood flow velocities (v_x, v_y, v_z) with a reference scan and three velocity-encoded acquisitions.

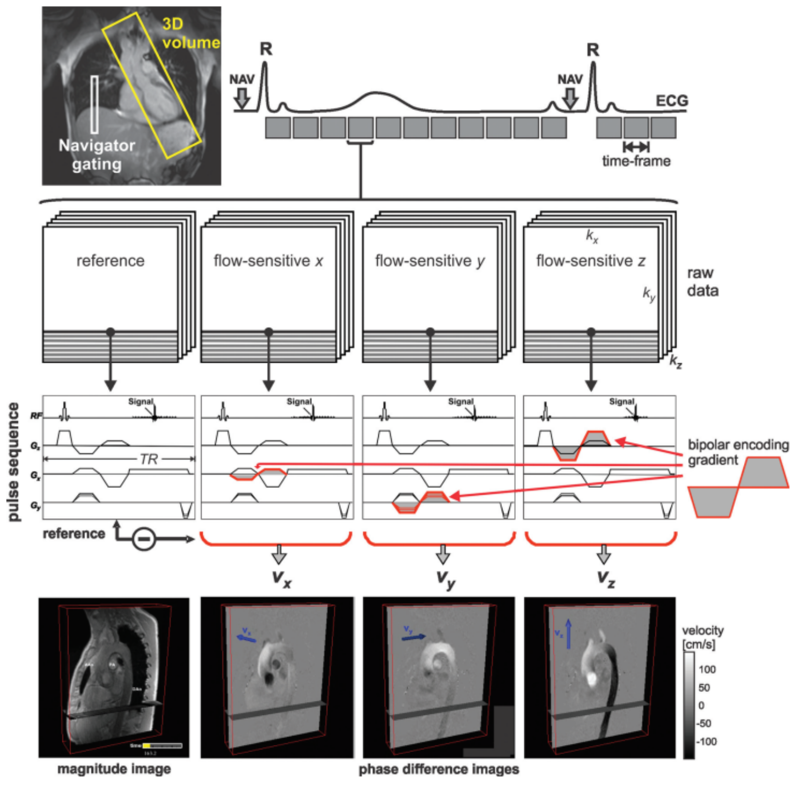

Figure 1.1: Schematic illustration of Cartesian 4D flow MRI of the thoracic aorta. For each time frame, four 3D raw datasets are collected to measure three-directional blood flow velocities (v_x, v_y, v_z) with a reference scan and three velocity-encoded acquisitions. Navigator gating of the diaphragm motion can be used for image acquisition during free breathing. The navigator pulse (NAV) is played out at the end of each cardiac cycle to update the current respiration phase which is used for respiratory gating. For applications in the aorta or pulmonary systems it is required an acquisition time of approximately 15-20 minutes with a temporal resolution of 40-50 msec. The image is taken from [25].

Scan times for 4D flow MRI can become prohibitively long, especially when high spatial resolution or large volumetric coverage are required. Studies involving compressed sensing techniques have enabled shortening of 4D flow MRI acquisition time [5, 36].
An alternative technique that is increasingly used to accelerate 4D flow MRI is radial data sampling. Radial acquisition schemes have been shown to permit 4D flow imaging with

improved scan time while providing large spherical volumetric coverage with high spatial resolution. [25, 41, 45].
Moreover, there are two main sources of inaccuracies:

1. the measurements come with an acquisition noise: it is generally considered to be zero-mean Gaussian in acquisition or frequency space. Since the inverse Fourier transform is linear, the noise maintains its characteristic when the k-space data is converted to complex image space data. However, converting the complex image space data to phase and magnitude is a non-linear operation that changes the noise characteristic [6, 12]. This can introduce a bias in the measured velocity that is dependent on magnitude of the actual velocity;

2. velocity aliasing problems: VENC should represent the maximum physiological velocity of the vessel of interest and when the velocity exceeds the VENC, velocity aliasing can occur which is typically visible as a sudden change from high to low velocity within a region of flow. To fix it, VENC can be increased and the acquisition is repeated to avoid aliasing. It is important to note that selecting a high VENC will also increase the level of velocity noise in flow velocity images because the spectrum of velocities allowed is enlarged and both high true velocities and potential noises present in the signal could be acquired. As a result, a compromise has to be found.

To summarize, 4D flow MRI data have noises inherent to their nature coming from a challenging acquisition in the k-space that can be negatively impacted by varying velocity ranges and by the practical need of subsampling the frequencies recorded. Moreover, the involvement of multiple cardiac cycles necessitates synchronization and an averaging procedures, which further increase the margin of error.

1.2. Physics informed neural networks

PINNs have been firstly introduced by Raissi et al. [33]. Such neural networks are constrained to respect any symmetries, invariances, or conservation principles originating from the physical laws that govern the observed data, as modeled by general time-dependent and nonlinear partial differential equations. This construction is capable of tackling a wide range of problems in computational science and leads to the development of new data-efficient and physics-informed learning models, as well as new approaches for model inversion and systems identification. The effectiveness of their proposal has been demonstrated over scientific applications in many areas [7, 24]. For instance, in fluid applications, PINNs have been used for fast surrogate modeling of idealized vascular flow problems in a forward parametric setting without training labels [43]. Moreover,

PINNs have also been formulated in an inverse modeling setting to extract unobservable information (e.g., blood flow velocity) from observable data (e.g., concentration data) in cardiovascular problems [1, 19, 34].
First attempts in computational physics in learning solutions of deterministic PDEs with neural networks in a space-time domain dates back at least to the early 1990s, e.g. [20, 27, 31]. It is exploited the fact that neural networks are universal approximators that can be used to approximate any continuous function and its derivatives [9, 17, 18, 21].

The central idea of PINNs is to take as input a set of points representing space-time coordinates and to return the corresponding output fields (e.g., velocity and pressure). The training aims to reduce the residual loss of the differential equations for the model output, over a set of collocation points sampled from the problem domain. This physics-informed loss function constrains the PINN from violating the differential equations, ensuring that its output obeys the governing physics. Together with this modelling knowledge, a data fidelity term is added in the loss definition and the distance between the neural net prediction and the data is minimized. The general structure is represented in Figure 1.2. To make PINN perform well, a balance between data-fidelity and physics-informed terms has to be found: data are needed as a starting guess and they propose some case-specific details of the solution, then physics will regularize and correct the possible noise present. The relevance of this trade-off will be discussed and analyzed in this work.

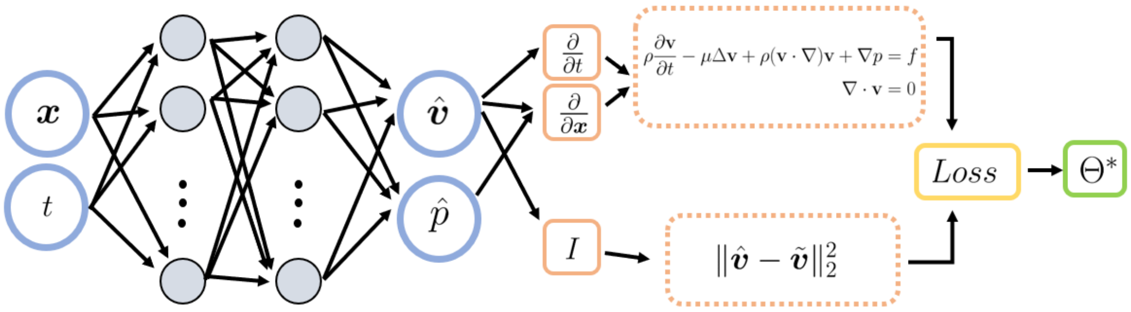
$\boldsymbol{x}$
t
$\hat{\boldsymbol{v}}$
$\hat{p}$
$\frac{\partial}{\partial t}$
$\frac{\partial}{\partial \boldsymbol{x}}$
I
$\rho\frac{\partial \mathbf{v}}{\partial t} - \mu\Delta\mathbf{v} + \rho(\mathbf{v}\cdot\nabla)\mathbf{v} + \nabla p = f$
$\nabla\cdot\mathbf{v} = 0$
$\|\hat{\boldsymbol{v}} - \tilde{\boldsymbol{v}}\|_2^2$
$Loss$
Θ^*

Figure 1.2: Network and loss general structure.

In hemodynamics applications, let consider velocity and pressure as desired output fields and consider 4D flow MRI as available measurements. The resulting net will be able to evaluate the desired fields in an arbitrary space-time point yielding to a super-resolved (SR) output. This is one of the main application of PINNs, since they are able to denoise and to increase the regularity of noisy, low resolute measurements. This quality output permits to investigate:

1. wall shear stress since its computation requires high resolute velocity fields in the near wall region;
2. pressure, that is not present in 4D flow MRI measurements, and it is completely given effortlessly alongside the regularization process of velocity field.

Moreover, leveraging on automatic differentiation does not introduce severe numerical errors both in WSS and PDEs residual computation.
The activation function plays a relevant role in representing the output fields. In this work, it is always preferred a sinusoidal activation yielding to a SIREN model structure, where SIREN refers to an MLP network combined with this non-linearity. Following [37, 40] a sinusoidal activation function is always preferred and SIREN refers to an MLP network combined with this activation function.
Periodic sinusoidal functions are employed in many applications to implicitly represent complicated signals such as images or 3D shapes. They are used also to capture model high-frequency information and higher-order derivatives in the context of solving differential equations. In contrast to conventional nonlinearities such as the hyperbolic tangent or the ReLU, the sine is periodic and therefore, non-local. Intuitively, this provides SIREN with a degree of shift invariance, as it may learn to apply the same function to different input coordinates. For example, velocity field has a similar behaviour in different regions of the geometry and this non-locality of the function used could help the learning speed and accuracy.
From [40] a strategy is taken to accurately initialize weights, in order to preserve the distribution of activations through the network so that the final output at initialization does not depend on the number of layers. This initialization produces an efficient and direct minimization starting from the very first iterations, avoiding recurrent oscillations in loss terms during the initial Adam epochs. Finally, each weight θ is initialized so that $\theta \sim \mathcal{U}(-\sqrt{6/c}, \sqrt{6/c})$, where c is the generic input feature size (c will be the number of neurons of the previous layer).
This strategy is heavily pursued in the 3D cases together with a correction of the first layer structure that is modified as $\sin(\omega_0 \cdot \Theta \boldsymbol{x} + \boldsymbol{b})$, where ω_0 is set to 30.

Despite the fascinating potential for a wide range of physic phenomena and applications, training an accurate PINN model in complex 3D applications remains a challenge. PINNs are computationally demanding and a large number of collocation points is required for matching the differential equations in order to train a good model. The physics introduced in the loss function succeeds in regularizing damaged data but fails in reconstructing specific flow details. Another crucial point that requires further development and research is the customization of the personalization of this neural network. An ad hoc model must be trained for each new patient since a new geometry and new specific data are prescribed. Finally, PINNs are typically constructed as a multi-layer-perceptron (MLP) network but, as an alternative, a convolutional neural network (CNN) could be considered [13, 14]. CNN has the capability to directly learn spatial relationship through kernels operations and specific kernels could be used also to compute derivatives with finite differences as an alternative to AD. For instance, Zhu et al. [47] developed a physics-constrained convolutional encoder-decoder to solve high-dimensional elliptic PDEs, and Geneva et al. [15] further extended this framework to dynamic hyperbolic PDEs with parametric initial conditions.

To further explore an alternative to AD, in [8, 11] the authors propose numerical differentiation methods to reduce the number of collocation points required to mitigate the computational resources used.

1.3. Synthetic data

Real patient specific 4D flow MRI data cannot be used in the first phase of the creation of an optimal strategy for regularizing data because there must be something true to compare and evaluate the prediction. Thus, synthetic data are created from an accurate CFD solution in a real geometry to simulate 4D flow measurements.

The degradation consists in a coarse frequency sampling in the corresponding *k-space* domain and in the addition of Gaussian noise. Moreover, 4D flow MRI measurements are usually taken with a frequency of about 40ms and to simulate this behavior a time average procedure is performed. The goal is to obtain corrupted velocity field as can be seen in Figure 1.3 where a synthetic data and a real measurement are visually compared.

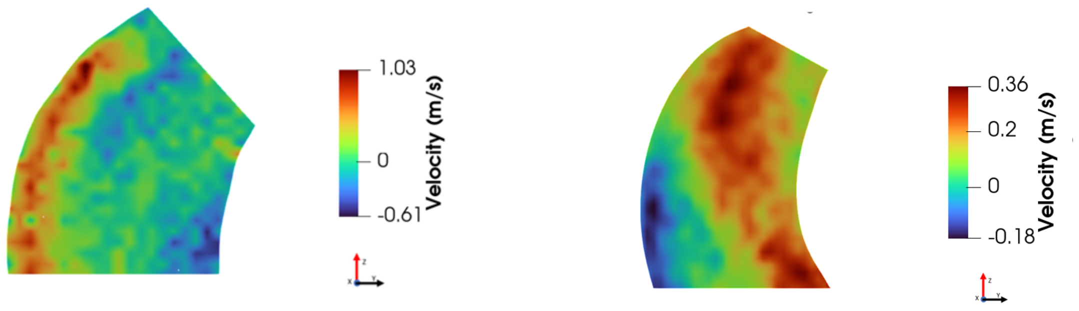

Figure 1.3: Visualization of velocity colormaps (foot-to-head component) on a sagittally oriented 2D aortic slice. On the left there are synthetic data on a reference geometry, while on the right real 4d flow MRI data on a BAV patient are represented.

The procedure used to create synthetic data is thought to emulate the 4D flow acquisition procedure: after a temporal averaging and a mapping in the *k-space*, a radial subsampling is performed, Gaussian zero-mean noise is added and finally the image is converted back from complex domain to real images. A more detailed description is provided in Section 4.2.

Organization of this work. The application, the mathematical models and methods are described in Chapter 2. Before working on real *in-vivo* data (Chapter 5), a synthetic 2D benchmark is considered in Chapter 3, and the optimal strategy to deal with 3D realistic data is found working on synthetic data in Chapter 4. Moreover, in the latter, we compare methods to compute WSS in order to accurately recover velocity gradients in the near wall region.
In Figure 1.4, it is presented a graphical summary of this thesis.

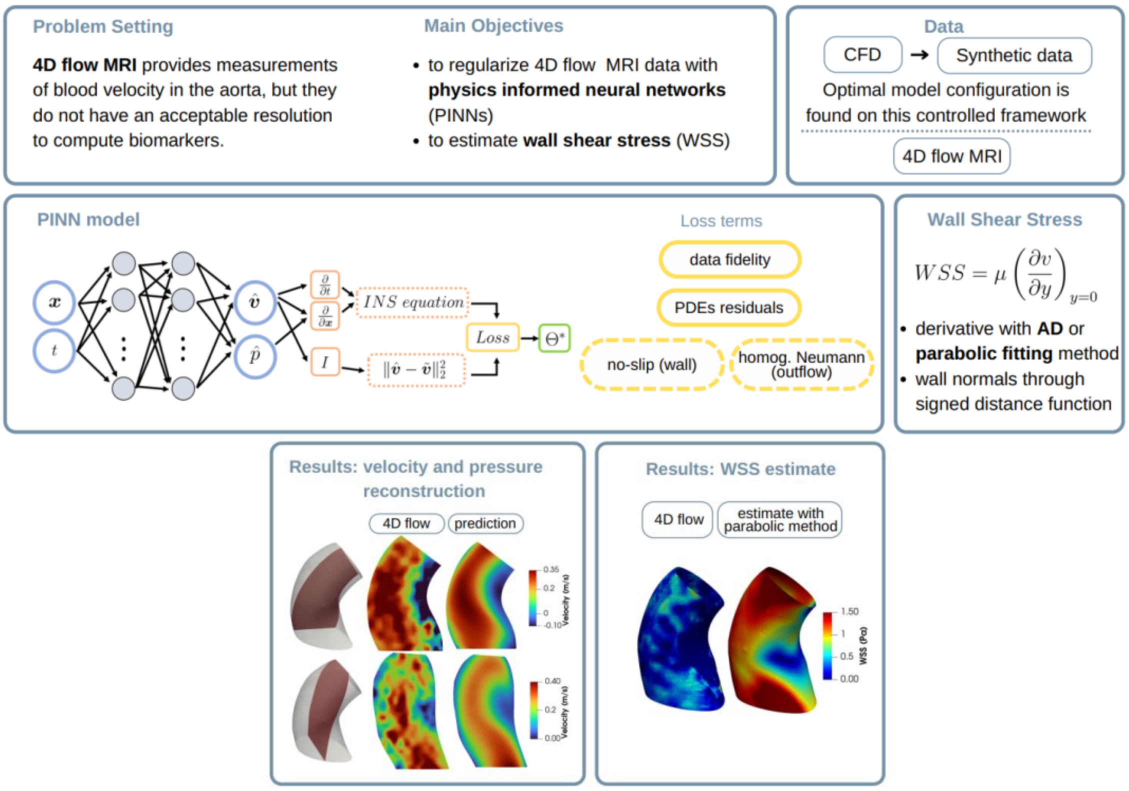
Problem Setting
4D flow MRI provides measurements of blood velocity in the aorta, but they do not have an acceptable resolution to compute biomarkers.
Main Objectives
• to regularize 4D flow MRI data with physics informed neural networks (PINNs)
• to estimate wall shear stress (WSS)
Data
CFD
Synthetic data
Optimal model configuration is found on this controlled framework
4D flow MRI
PINN model
$\boldsymbol{x}$
t
$\hat{\boldsymbol{v}}$
$\hat{p}$
$\frac{\partial}{\partial t}$
$\frac{\partial}{\partial \boldsymbol{x}}$
I
$INS\ equation$
$\|\hat{\boldsymbol{v}} - \tilde{\boldsymbol{v}}\|_2^2$
$Loss$
Θ^*
Loss terms
data fidelity
PDEs residuals
no-slip (wall)
homog. Neumann (outflow)
Wall Shear Stress
$WSS = \mu \left(\frac{\partial v}{\partial y}\right)_{y=0}$
• derivative with AD or parabolic fitting method
• wall normals through signed distance function
Results: velocity and pressure reconstruction
4D flow
prediction
Velocity (m/s)
Results: WSS estimate
4D flow
estimate with parabolic method
WSS (Pa)

Figure 1.4: Overview of this thesis. After defining the motivation and the main objectives, there are represented the model structure and few results for velocity reconstruction and WSS estimate.

2 | Methods

In this Chapter, there are presented the main features of the model used. PINNs are characterized by the definition of the training strategies and, above all, by the structure of the loss function with the description of all its terms. In Section 2.2, the methods to compute wall shear stress are formalized. Moreover, it is reported a deep learning approach involving the representation of a sign distance function to obtain the required wall normals.

2.1. Problem setting

PINNs regularize and improve the spatio-temporal resolution of 4D flow images within a deep learning framework. The neural network takes as input spatio-temporal coordinates and returns the velocity and pressure fields at those points. During the training, the net is led by a data-fidelity term built on the acquired data and by physical laws that regularize the output, forcing it to respect the governing PDEs. With this super resolute velocity field is possible to compute accurate biomarkers such as WSS through the analysis of velocity gradients near the wall.

In this applications one is only interested in analyzing the flow evolution during the time-interval $[t_1, t_2]$ in a small region of the aorta that will be denoted by the bounded domain Ω.
A sample $\mathbf{X}$ in $\Omega \times [t_1, t_2]$ is taken as input of the neural network. Consider a point $(\boldsymbol{x}_i, t_i)$, let the PINN output velocity field be $\boldsymbol{v}_i(\boldsymbol{x}_i, t_i) = (u_i(\boldsymbol{x}_i, t_i), v_i(\boldsymbol{x}_i, t_i), w_i(\boldsymbol{x}_i, t_i))$, where u, v, w stands as the x-, y-, z- component of the velocity and let $p_i(\mathbf{x}_i, t_i)$ be the PINN output pressure field. Finally, the complete output of the net stands in $\mathbb{R}^4$ with the velocity components and pressure concatenated.
To summarize, the net has to learn a continuous function $f : \mathbb{R}^4 \rightarrow \mathbb{R}^4$ that maps the spatio-temporal coordinates in the velocity and pressure fields. To approximate f an MLP is used with weights Θ and as activation function $a(\cdot)$ it is always chosen $\sin(\cdot)$.
In a generic layer k the input x_k represents the spatio-temporal coordinates and the fol-

lowing operation is performed:

$$x_{k+1} = a\left(\Theta_k x_k + b_k\right)$$

where Θ_k and b_k are the weights and biases of the kth-layer.

2.1.1. Governing PDEs

Blood is considered as a Newtonian fluid with constant density and its evolution is modelled by the incompressible Navier-Stokes (NS) equation. This theory is enforced in some of PINNs' loss terms forcing the predicted output to follow the Navier-Stokes equation together with the incompressibility constraint.

$$\begin{cases} \rho\dfrac{\partial \mathbf{u}}{\partial t} - \mu\Delta\mathbf{u} + \rho(\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = f & \Omega\times[t_1,t_2], \\ \nabla\cdot\mathbf{u} = 0 & \Omega\times[t_1,t_2], \\ \mathbf{u} = \mathbf{0} & \Gamma_{\text{wall}}\times[t_1,t_2], \\ \mathbf{u} = \mathbf{u}_{\text{in}} & \Gamma_{\text{in}}\times[t_1,t_2], \\ \mu\nabla\mathbf{u}\cdot\mathbf{n} - p\mathbf{n} = \mathbf{0} & \Gamma_{\text{out}}\times[t_1,t_2] \end{cases} \tag{2.1}$$

The non-dimensionalized version of NS equation is also taken into account to generalize the process to avoid a patient-specific choice of few hyperparameters. The Reynolds number is defined as $Re = \dfrac{\rho U L}{\mu}$.

$$\begin{cases} \dfrac{\partial \mathbf{u*}}{\partial t} - \dfrac{1}{Re}\Delta\mathbf{u*} + (\mathbf{u*}\cdot\nabla)\mathbf{u*} + \nabla p^* = f & \Omega\times[t_1,t_2], \\ \nabla\cdot\mathbf{u*} = 0 & \Omega\times[t_1,t_2], \\ \mathbf{u*} = \mathbf{0} & \Gamma_{\text{wall}}\times[t_1,t_2], \\ \mathbf{u*} = \mathbf{u*}_{\text{in}} & \Gamma_{\text{in}}\times[t_1,t_2], \\ \dfrac{\nu U}{L}\nabla\mathbf{u*}\cdot\mathbf{n} - \rho U^2 p^*\mathbf{n} = \mathbf{0} & \Gamma_{\text{out}}\times[t_1,t_2] \end{cases} \tag{2.2}$$

2.1.2. Training

Data are non-dimensionalized to facilitate the training, and the characteristic length L and velocity U are the same used in the non-dimensionalized Navier-Stokes equa-

tions. Those parameters are taken looking only at training data, leaving a subset of test data unseen to avoid any biases in the evaluation. L is taken as the maximum in the first two space coordinates that represent the lengths of an axial plane; U is the maximum velocity and T is defined as the ratio between L and U. Finally, the density ρ, the dynamic viscosity μ and $\nu = \frac{\mu}{\rho}$ are taken accordingly to the typical blood behavior in the aorta.
The following non-dimensionalization operations are performed for both synthetic and real data:

$$\boldsymbol{x}_i^* = \frac{\boldsymbol{x}_i}{L}, \quad t_i^* = \frac{t_i}{T},$$

$$\boldsymbol{u}_i^* = \frac{\boldsymbol{u}_i}{U}, \quad p_i^* = \frac{p_i}{\rho U^2}.$$

The training consists in the minimization of the loss function $\mathcal{L}$, that will be defined in the next section, with respect to the PINN's weights Θ:

$$\min_{\Theta} \mathcal{L}(f_\Theta).$$

As optimization strategy it is used a combination of Adam and Limited-memory BFGS (L-BFGS) [44] optimizers to try avoiding falling in local minima characterized by high-values of the loss function. Usually, few epochs of Adam are performed, then the training is concluded with the more accurate LBFGS optimizer [22]. The latter is a quasi-Newton method that uses also the Hessian matrix in the algorithm. The estimation of the second-order partial derivatives could be difficult in the first iterations, thus an initial step with a simpler first-order methods, such as Adam optimizer, is preferable.

Regarding the implementation of the neural networks and their training, it is used the `nisaba` library [35] implemented at MOX - Department of Mathematics, Politecnico di Milano.

2.1.3. Loss function

Building a meaningful loss function is the real and interesting challenge when dealing with neural networks: all the information at disposal must be combined and balanced at this stage. Hence, this function consists in terms combining data-fidelity with respect

to 4D flow data and the imposition of Navier-Stokes equations together with boundary conditions. A Tikhonov regularization term is also added to mitigate overfitting and to help network training discouraging the formation of large weights.
As notation, f_Θ stands for the PINN model used, $\tilde{}$ stands for 4D flow data and $\hat{}$ refers to the model prediction. The comparison between two quantities is performed with mean squared error.
With $\mathbf{X}$ is denoted a generic subset of points in the domain $\Omega \times [t_1, t_2]$ suitably adimensionalized. Two set of points with a potentially non-null intersection are usually used: the fitting points denoted by $\mathbf{X}_{\boldsymbol{fit}}$ in which 4D flow measurements are available and the collocation points $\mathbf{X}_{\boldsymbol{coll}}$ where the PDEs residual is evaluated. A generic subset $\boldsymbol{X}$ can be subdivided in $\mathbf{X}_{\boldsymbol{wall}}$ and $\mathbf{X}_{\boldsymbol{out}}$ corresponding to points at the lateral wall boundary of the geometry where a no-slip condition is imposed and corresponding to points at the outflow boundary respectively. Those two subset are used for boundary conditions imposition.
The complete loss can be written as:

$$\mathcal{L}(\Theta) = w_{fit}\mathcal{L}_{fit} + w_{wall}\mathcal{L}_{wall} + w_{out}\mathcal{L}_{neum} + w_{mass}\mathcal{L}_{mass} + w_{mom}\mathcal{L}_{mom} + w_{reg}\mathcal{L}_{reg}$$

where:

- $\mathcal{L}_{fit}(\mathbf{X}_{\boldsymbol{fit}}) = \dfrac{1}{N_{fit}}\sum_{\mathbf{X}_{\boldsymbol{fit}}} \|\hat{\boldsymbol{v}} - \tilde{\boldsymbol{v}}\|_2^2,$
- $\mathcal{L}_{wall}(\mathbf{X}_{\boldsymbol{wall}}) = \dfrac{1}{N_{wall}}\sum_{\mathbf{X}_{\boldsymbol{wall}}} \|\hat{\boldsymbol{v}}\|_2^2,$
- $\mathcal{L}_{neum}(\mathbf{X}_{\boldsymbol{out}}) = \dfrac{1}{N_{out}}\sum_{\mathbf{X}_{\boldsymbol{out}}} \left\|\dfrac{\nu}{UL}\nabla\hat{\boldsymbol{v}} \cdot \mathbf{n} - \rho U^2 \hat{p}\mathbf{n}\right\|_2^2,$
- $\mathcal{L}_{mass}(\mathbf{X}_{\boldsymbol{coll}}) = \dfrac{1}{N_{coll}}\sum_{\mathbf{X}_{\boldsymbol{coll}}} \left\|\dfrac{1}{L}\nabla \cdot \hat{\boldsymbol{v}}\right\|_2^2,$
- $\mathcal{L}_{mom}(\mathbf{X}_{\boldsymbol{coll}}) = \dfrac{1}{N_{coll}}\sum_{\mathbf{X}_{\boldsymbol{coll}}} \left\|\dfrac{\partial\hat{\boldsymbol{v}}}{\partial t} - \dfrac{1}{Re}\Delta\hat{\boldsymbol{v}} + (\hat{\boldsymbol{v}} \cdot \nabla)\hat{\boldsymbol{v}} + \nabla\hat{p}\right\|_2^2,$
- $\mathcal{L}_{reg} = \sum_{layers\ k} \Theta_k^2,$ where Θ are the network's weights,

and $\hat{\boldsymbol{v}}$ represent the first three component of $f_\Theta\left(\mathbf{X}_{(\cdot)}\right)$.

There is a balance between the physics and data-fidelity term, so the loss weights must be chosen accordingly. It is not suitable to rely always on data, since the goal is to modify them obtaining a more accurate and regular solution, on the other hand, the governing

PDEs allow also a zero constant solution and the physics terms tend to smooth and remove all the details toward a constant value for every point. It is optimal for both terms to collaborate effectively so that each can bring their advantages, in order to achieve the best result during the trade-off. A strategy could be to dynamically modify the weights also relying on the gradient of each loss term [26], but this will lead to a heavier training in terms of resources used and could be not optimal for all test cases. The general idea is to weight more the data-fidelity term at the beginning and then rely more on physics.
One strategy could be a more intense first step where the fit loss weight is about one order of magnitude larger with respect to the momentum and mass weights and a small second step, around 15% of the first round epochs, with the same magnitude between data-fidelity and physics terms. As an alternative, one can consider also to add an initial step in which the model only interpolates 4D flow data and thus without PDEs' residual terms. In the latter the pressure cannot be correctly reconstructed since the only useful information comes from the momentum equation, thus if many interpolation-only epochs are performed it could happen that, once physics is added, the velocity field could be wrongly modified since the pressure has assumed unusual and random values. The classic and more straightforward solution remains to use the same weights starting from the beginning relying more on fit data. Moreover, those weights do not show evidence that one term is more relevant with respect to the other since they cannot be equally compared: a PDE's residual is different from a measured distance from a data-fidelity value.
The PDEs are imposed in an adimensionalized version to facilitate the generalization of the whole process to different patients and test cases. This is because, once the loss weights are tuned in a reference adimensionalized framework, they are less likely to be changed and they do not depend on test case specific features.
About boundary condition, we impose a no-slip condition in the wall boundary and a homogeneous Neumann boundary condition in the outflow. This permits to fix a reference value for the pressure that otherwise is defined up to a constant.
In addition to the main role of regularizing the 4D flow data, the minimization of the PDEs' residuals in the loss function is the only way to correctly reconstruct the pressure field. This can be also a meaningful marker to check during the training in order to understand whether physics is bringing an added value or the net is simply interpolating data.
In this context, the physics loss terms have a crucial role and they must be built upon an accurate computation of gradients with respect to the spatio-temporal coordinates. This is usually not trivial in many applications but with PINNs it becomes extremely easy thanks to automatic differentiation (AD) [2].

2.1.4. Error quantification

To quantify errors obtained in experiments and simulations, two metrics are used. Differences between a reference vector field $\boldsymbol{u_{ref}}$ and another generic vector field $\boldsymbol{u}$ are evaluated by computing magnitude and vector normalized-rootmean-squared-errors (mNRMSE and vNRMSE, respectively). They are defined as follows:

$$mNRMSE = \frac{1}{max|\boldsymbol{u_{ref}}|}\sqrt{\frac{1}{K}\sum_{k=1}^{K}(|\boldsymbol{u}| - |\boldsymbol{u_{ref}}|)_k^2},$$

$$vNRMSE = \frac{1}{max|\boldsymbol{u_{ref}}|}\sqrt{\frac{1}{K}\sum_{k=1}^{K}(\boldsymbol{u} - \boldsymbol{u_{ref}})_k^2},$$

where K is a generic number of points belonging to $\Omega \times [t_1, t_2]$ where the two velocity fields are evaluated.

2.2. Wall Shear Stress analysis

Computing clinically relevant hemodynamic biomarkers, such as wall shear stress, is one of the application of the regularization of 4D flow data, which is especially relevant in congenital vascular pathologies. It is defined as

$$WSS = \mu\left(\frac{\partial v}{\partial y}\right)_{y=0},$$

where μ is the dynamic viscosity, v is the component of the velocity vector that is locally parallel to the wall, and y is the Euclidean distance from the wall.
This derivative can be computed following two approaches: fitting a parabolic profile on three points near the wall and calculate the analytical derivative or through automatic differentiation. In the first one, discrete normals coming from the mesh structure are used, while in the second approach a continuous distance function is evaluated to obtain the normals.

2.2.1. Normal computation through a signed distance function

An accurate computation of wall normals is essential for computing tangential velocity and for identifying the correct direction in which calculate the derivative. This can be done with a deep learning approach exploiting again automatic differentiation. Once the distance from the nearest wall boundary is evaluated in each point of the domain Ω, normals $\boldsymbol{n}$ can be obtained from the norm of the gradient of this distance:

$$\boldsymbol{n} = \frac{|\nabla\phi|}{\|\nabla\phi\|}$$

where ϕ is the sign distance function.
ϕ is obtained through the neural network SDF that takes as input the spatial coordinates of a generic point and returns the corresponding distance with sign from the wall,

$$\phi = \mathrm{SDF}(\boldsymbol{x}), \;\; \boldsymbol{x} \in \Omega.$$

For all the geometries analyzed the same MLP network architecture is used: it has 5 hidden layers with 32 neurons each and $\sin(\cdot)$ is always taken as activation function.
To create the training data, a point cloud sampling P is generated in a parallelepiped, embedding the domain Ω plus an additional margin.

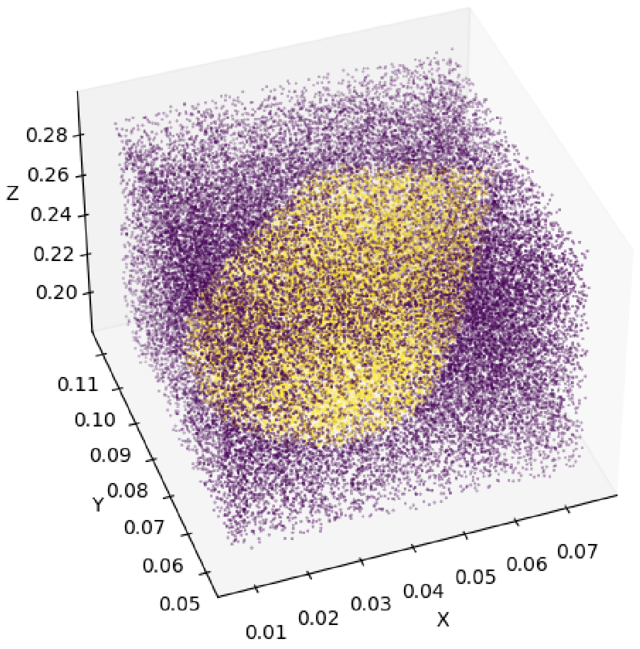

Figure 2.1: Sampling in the parallelepiped containing the reference aorta geometry. The yellow points represent the boundary of Ω.

In P two subsets are defined: P_w is the set of points defying the wall boundary, they will have a zero distance $|\phi_w|$; the remaining points are contained in P_s and to define their distance from the wall a distance matrix M of size $(|P_s| \times |P_w|)$ is created, and $M_{ij} = \mathrm{dist}(P_{s,i}, P_{w,j})$. The final distance $|\phi_s|$ is obtained taking the minimum in each row of M.

SDF has to learn the sign distance function but it is trained only looking at the absolute value of the distances. Indeed, the loss function is defined as:

$$\sum_{x_i \in P_s} \| \, |\mathrm{SDF}(\boldsymbol{x_i})| - |\phi_{s,i}| \, \|_2^2 .$$

It is not predictable if the net will approximate positive or negative distances inside the geometry and this can be controlled a posteriori by changing the output sign. For normal computation purposes, it is better to consider positives distances outside to end up with outwards normals.
Inlet and outlet points are considered as zero wall distance points, even if they are not wall points. This will create a more continuous distance in those boundary regions, improving normals estimation. Then, when computing WSS, those points will be excluded since they do not represent a real wall boundary.

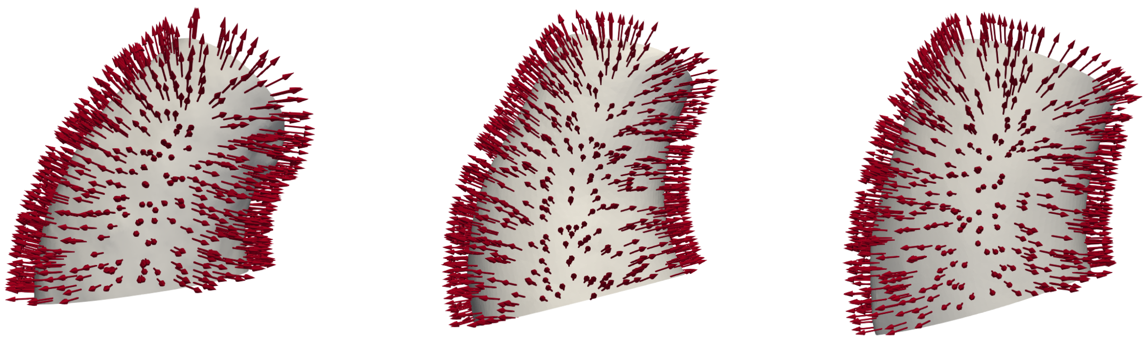

Figure 2.2: Normals computed through a case-specific SDF net. The first one is the reference geometry used for the CFD simulation and for synthetic data, the others are geometries segmented from measuraments on real patients. They all represent a region of the ascending aorta.

Following this procedure, normals can be easily computed thanks to a continuous representation of a sign distance function. However, the net must be trained specifically for a single geometry and this is not optimal when dealing with many patients or with different areas of the aorta. Following [28], it is possible to add a latent representation of the geometry and train multiple instances together. The net will be able to quickly compute sign distance values for a new geometry, furthermore it will be possible to explore the latent space to discover and represent new possible geometries.

2.2.2. WSS computation

Parabolic fitting. Let a wall point be p_{i0} and let the corresponding discrete normal coming from the mesh be $\boldsymbol{n_i}$. On the normal direction, consider two further points p_{i1} and p_{i2} at distance d_n and $2d_n$, respectively. Exploiting PINNs interpolation properties we can evaluate velocity in those points and compute the tangential component as:

$$\boldsymbol{v_{i0,\,tan}} = \boldsymbol{v_{i0}} - (\boldsymbol{v_{i0}} \cdot \boldsymbol{n_i})\boldsymbol{n_i},$$

$$\boldsymbol{v_{i1,\,tan}} = \boldsymbol{v_{i1}} - (\boldsymbol{v_{i1}} \cdot \boldsymbol{n_i})\boldsymbol{n_i},$$

$$\boldsymbol{v_{i2,\,tan}} = \boldsymbol{v_{i2}} - (\boldsymbol{v_{i2}} \cdot \boldsymbol{n_i})\boldsymbol{n_i}.$$

The magnitudes of those vectors are computed and a parabolic function $f = ay^2+by+c$ is fitted among those values. Finally, the derivative is calculated analytically as $\frac{df}{dx} = 2ay+b$. This is one among the approaches described in [29] that presents also linear interpolation between the points on the inward wall normal and a Fourier Velocity Encoding method.

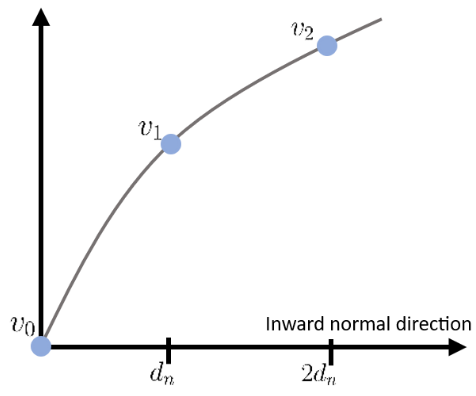

Figure 2.3: Parabolic profile fitted on three points.

Once the normal derivative is computed, WSS is obtained by evaluating the derivative at the wall and multiply it by μ.
This procedure together with an interpolation scheme can be applied also for real 4D flow data or reference CFD solutions. In those cases it is not possible to evaluate any point since only fixed mesh points are available and a nearest-neighbor interpolation is required yielding to a potential noisy behaviour in WSS field.

Automatic differentation. One can evaluate WSS also by exploiting automatic differentiation to compute velocity gradients. In this case there is no need to choose an optimal d_n and derivatives are easily obtained through the network structure. Firstly, the model is evaluated in boundary wall points, and the tangential velocity components are calculated based on the wall normals obtained as described in Section 2.2.1. The gradient of the tangential velocity with respect to x-, y-, z- coordinates is computed through AD, and the final normal derivative is obtained by multiplying the gradient by $\boldsymbol{n}$.

With the first approach, a parabolic profile is super-imposed in all regions, instead with automatic differentiation the real output velocity profile is implicitly considered. From a theoretical point of view, the second method is more accurate and realistic but it requires a very precise reconstruction at boundaries. For example, if in the near wall region a flat or a negative velocity profile is predicted, then the WSS estimate will be imprecise. Fitting a parabola using points far away from the wall could fix this issue.

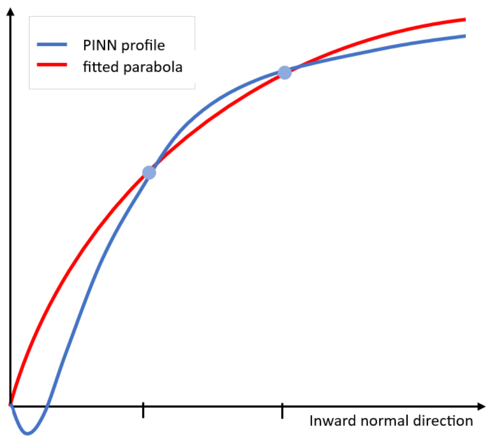

Figure 2.4: Example of a PINN and parabolic velocity profile fitted on three points. This shows an overfitting behaviour due to the enforcing of no-slip condition that yields to an inaccurate WSS calculation.

The no-slip condition heavily forces the PINN profile to the behaviour in Figure 2.4. Moreover, even if the gradient sign does not change, it could be smoothed losing the typical velocity slope at the boundaries. This is an overfitting behaviour, and it could be balanced tuning the wall loss term weight. One can decide to reduce the force used to impose the no-slip condition and, through AD, obtaining WSS that could be slightly smoothed since velocity does not arrive at zero. Otherwise, the fitted parabola will avoid overfitting problems and the no-slip condition could be heavily forced, but the parabolic profile obtained in this way could be unrealistic and, again, potentially too smoothed. To further improve the estimate, a more refined sampling of collocations points near the wall could yield to a better velocity profile since it will be guided by the momentum NS equation. This balance is discussed in 3D test cases reported in Chapter 4 and Chapter 5, as it is expected that automatic differentiation will provide a more accurate computation of WSS, but requires a clear and realistic profile at the wall. The quality of the provided data must be taken into consideration and 4D flow measurements usually suffers in accuracy at boundaries, due to the acquisition and the segmentation procedures.

3 | Test Case 1: 2D flow around a cylinder

In this Chapter, it is analyzed a toy reference test case of a 2D flow around a cylinder. We consider this starting benchmark for its reduced computational complexity to construct the computational pipeline for the more complex 3D patient specific case. In this stage the implementation of the problem is verified together with the procedure to generate synthetic data. In Section 3.2.1, it is reported a special method to enforce Dirichlet boundary condition in PINNs that recalls the lifting procedure. The obtained results do not show the final complexity that will be faced in the 3D cases where the flow will be extremely more detailed, and the applied strategies together with the net architecture must be enriched with more complexity.

3.1. CFD simulation

The Navier-Stokes equation is considered with a no slip condition on the wall boundary Γ_{wall}, a parabolic profile on inlet Γ_{in} and homogeneous Neumann boundary condition on the outlet Γ_{out}. The channel has a size of 0.7 m $\times$ 0.4 m, and the cylinder has a radius of 0.05 m.

A CFD simulation in `Fenics` [23] is performed to have an accurate reference to evaluate performances and to create synthetic data to emulate 4D flow MRI measurements. For the numerical simulation we exploit the incremental Chorin Themam projection method of the first order. Physical parameter are set as follows: density $\rho = 1000\frac{\text{kg}}{\text{m}^3}$, dynamic viscosity $\mu = 0.1\frac{\text{kg}}{\text{ms}^2}$ and $Re \sim 2000$ with $\text{U} = 0.29\ \frac{\text{m}}{\text{s}}$ and $\text{L} = 0.7\text{m}$.

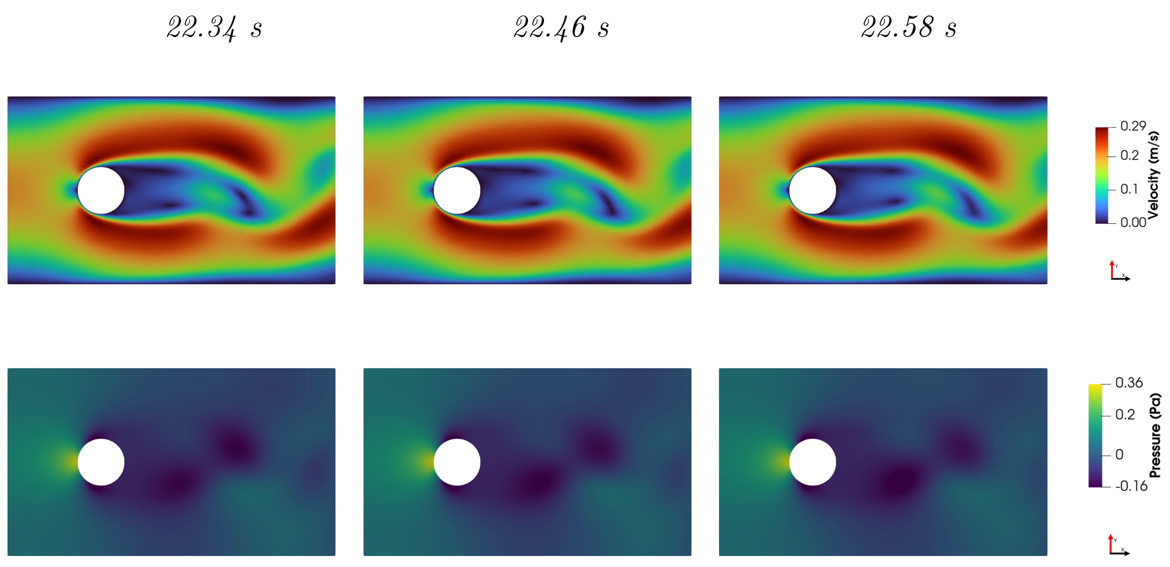

Figure 3.1: Visualization of velocity magnitude colormaps (first row) and pressure (second row) in three different timesteps for the CFD reference solution.

3.2. Model description

Complexity is slowly added in few steps: at first only CFD data are considered, then pressure is removed from the dataset seen by the net and finally synthetic noisy 4D flow data are used during training. The time dependency of the solution is the final element to face: at the beginning, it is considered a time interval where the solution is almost stationary and then, it is moved forward to reach the wake evolution stage. To emulate what happens with real measurements that are typically recorded with a 40ms frequency, also training data are sampled by a similar time length. In this test case 3 of those timesteps are considered yielding to a time interval of 1.2 seconds.
The loss function is built with the data-fidelity term and the physics terms (mass and momentum NS equation) described in Section 2.1.3. About boundary conditions, the Neumann homogeneous outflow loss term is added only when pressure is not included in the training data. The no-slip condition is not enforced in the loss term and an alternative approach is followed to emulate the lifting procedure in CFD simulations. Moreover, regularization and SIREN initialization are not applied.

3.2.1. Lifting method to enforce no slip condition

To enforce no slip condition, we implement a strategy based on multiplying the velocity output of the net by a continuous function that is null on the wall boundaries and increases

in proportion with the distance from the wall. In this way not only the no-slip condition is automatically imposed, but also the net is forced to follow the typical increasing velocity profile near the wall. The advantage of this solution is that the loss function is lightened: it has one less term to take into account during the minimization procedure since the no-slip condition is completely managed by the lifting operator.
Consider the PINN as $f_\Theta(\cdot)$, the net is evaluated in various space-time coordinates $(\mathbf{x}, t) \in XT = [X \times [t_1, t_2]]$, and it returns the velocity and pressure field $[\mathbf{v}, p]$. Let be $\Psi(\mathbf{x}) : X \to \mathbb{R}$ the lifting operator, then, the final output will be:

$$\begin{bmatrix} \mathbf{v} \\ p \end{bmatrix} = f_\Theta(\mathbf{x}, t)\, \Psi(\mathbf{x}),$$

where $\Psi(\mathbf{x}) = [\psi, \psi, 1]^T$ with ψ representing the distance from the wall boundary.

This multiplication can be alternatively seen as the final layer at the end of the original architecture.
Moreover, the net is aware of the lifting's action since this operation is seen in the loss gradients, and thus in the entire back-propagation procedure:

$$\nabla_{\mathbf{x},t}\mathcal{L} = \nabla_{\mathbf{x},t} f_\Theta(\mathbf{x}, t)\Psi(\mathbf{x}) + f_\Theta(\mathbf{x}, t)\nabla_{\mathbf{x}}\Psi(\mathbf{x}).$$

There is a cooperation between the two, as they are not independent. This mean that, potentially, the net could propose a wall velocity different from zero if Ψ is not exactly null. However, this will be not the case since it is reasonable to impose null velocity at the wall, and it is not likely that fidelity data will conduct to a very different behaviour.

The distance from the wall $\Psi(\cdot)$ can be defined analytically or can be approximated by a further neural network. Since the domain for this 2D test case is very simple, the wall boundaries can be defined analytically as a set of different edges. With this geometry representation, the continuous distance can be easily computed as in [3, 4, 42]. In this work the idea to enforce Dirichlet boundary condition is employed in a lifting fashion within a PINN framework.
Let be d the signed distance function from $\mathbf{x} = (x, y)$ to the line defined by the segment AB of length L with vertices $A = (x_A, y_A)$ and $B = (x_B, y_B)$:

$$d(\mathbf{x}) = \frac{(x - x_A)(y_B - y_A) - (y - y_A)(x_B - x_A)}{L}.$$

Then, let be $(x_c, y_c) = ((x_A + x_B)/2, (y_A + y_B)/2)$ the center of AB and let define tr as the following trimming function:

$$tr(\mathbf{x}) = \frac{1}{L}\left[\left(\frac{L}{2}\right)^2 - \|(x, y) - (x_c, y_c)\|^2\right]$$

Note that $tr \geq 0$ defines a circle of center (x_c, y_c). Finally, the distance function ψ is defined as:

$$\Psi(\mathbf{x}) = \sqrt{d^2 + \left(\frac{\sqrt{tr^2 + d^4} - tr}{2}\right)^2}. \tag{3.1}$$

A graphical representation of $d(\mathbf{x})$, $tr(\mathbf{x})$ and $\psi(\mathbf{x})$ for a generic segment is show in Figure 3.2.

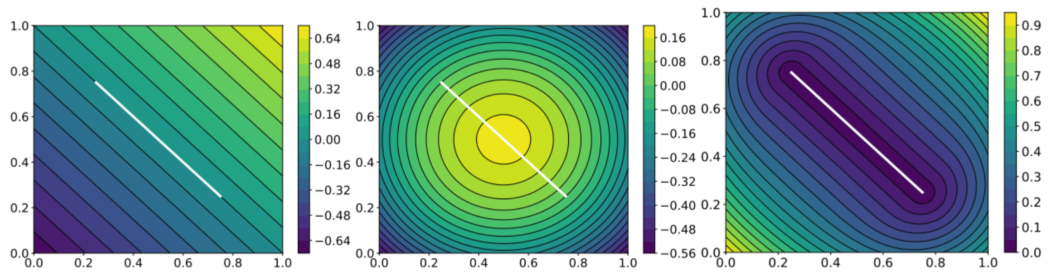

Figure 3.2: Representation of the signed distance function $d(\mathbf{x})$ to a straight line (left), the trimming function $tr(\mathbf{x})$ (middle) and the approximate distance function $\Psi(\mathbf{x})$ to a segment (right). The image is taken from [3].

Assuming that the wall boundary can be expressed ad the union of n_s segments $\{s_1, ..., s_{n_s}\}$, then Ψ final distance from the wall, normalized up to order ≥ 1, is defined as:

$$\Psi = \frac{1}{\sqrt[m]{\frac{1}{\psi_1^m} + \frac{1}{\psi_2^m} + ... + \frac{1}{\psi_{n_s}^m}}},$$

where ψ_i is the distance from the segment s_i. About the normalization, it guarantees that for every regular point of the wall boundary, the following holds:

$$\Psi = 0, \qquad \frac{\partial \Psi}{\partial n} = 1, \qquad \frac{\partial^k \Psi}{\partial n^k} = 0 \quad (k = 2, 3, ..., m).$$

3.2.2. Model parameters

The following parameters represent the model architecture and training strategy used in 2D test cases:

- model architecture: 4 hidden layers with 20 neurons each, and 3 neurons in both input and output layers;
- sin(·) as activation function;
- lifting applied with an analytical closed function that defines the distance from the wall (3.1);
- first minimization loop: 500 epochs with Adam optimizer (learning rate: 0.01);
- second minimization loop: 5000 epochs with L-BFGS optimizer;
- 5000 collocations points where the physics loss terms are evaluated and 3000 data-fidelity points both spread in the entire time interval are considered;
- loss weights are reported in Table 3.1:

	w_{fit}	w_{mass}	w_{mom}	$w_{neumann}$
weights	1	0.01	0.01	0.05

Table 3.1: Loss weights for 2D test case.

3.3. Results

CFD clean data are used to train an interpolation-only network in order to fix a reference architecture that could represent the entire complexity of the flow. Pressure must be included in the fitting data since it cannot be otherwise reconstructed without the physics loss terms. With 4 hidden layers composed by 20 neurons each, all velocity details are correctly reconstructed as expected since the data are super accurate, and they brought only very clear information.
In those tests, it is possible to verify the physics loss residual to understand if clean data can satisfy the NS equations when gradients are computed through AD. Even if the solution fits perfectly the CFD ground-truth, the PDEs residual is not null as PINNs cannot reach accuracy of classic CFD solvers. However, the goal of this work is to regularize existing data and, even if all details are not reconstructed, this is a way to extract information from the available measurements.
When removing pressure from the data fidelity loss term, physics starts to be relevant and

crucial for its reconstruction. Momentum equation, together with the outflow boundary condition, reconstructs a relative pressure field with a fixed reference value in the outflow that comes from the homogeneous Neumann boundary condition.
The final step is to consider noisy and corrupted data to emulate 4D flow measurements. CFD ground-truth data are time-averaged within a time window of 40 ms, and a final 0.6s time interval is considered. In this period of time the solution is not stationary, and it is possible to appreciate the wake evolution behind the cylinder.
A Cartesian grid with voxel size of 0.002×0.002 m^2 is taken to sample data before moving to the Fourier domain where the 98% of frequencies are radially sampled. Moreover, a fully sampled calibration region of 5×5 in the center of k-space is maintained and no noise is added in this test case. Finally, complex images were converted back to real images of velocity fields to obtain noisy synthetic measurements.

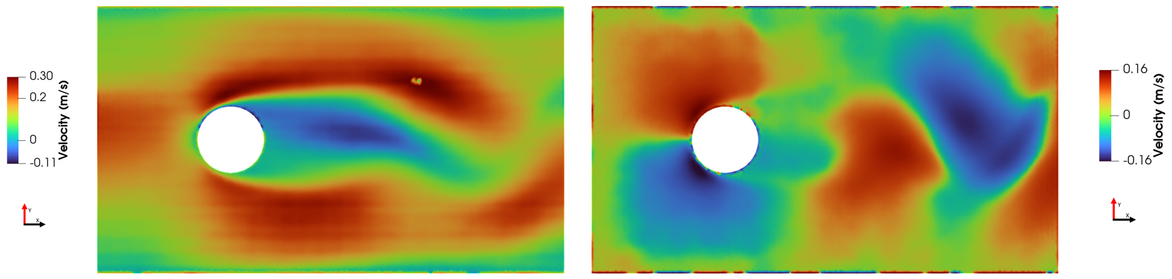

Figure 3.3: Visualization of synthetic velocity colormaps in a generic timestep. X- and Y- components on the left and on the right, respectively.

These data are not a faithful representation of real 4D flow MRI measurements, as they usually present a more noisy behaviour, characterized by a speckled representation. Obstacles have been encountered in the synthetic data creation process in 2D since the flow does not present many complex details: this does not permit to have a wide range where noise level could vary. Slightly changing one parameter yields to unusable noisy data, and thus it is challenging to find an optimal choice of the generation parameters. When moving to 3D test cases this procedure will become more realistic and easier to apply.

3.3.1. Loss evolution

As it can be seen in Figure 3.4, the data-fidelity term maintains a main role during the entire training but it reaches a plateau after 1500 epochs. Physics terms, instead, keep

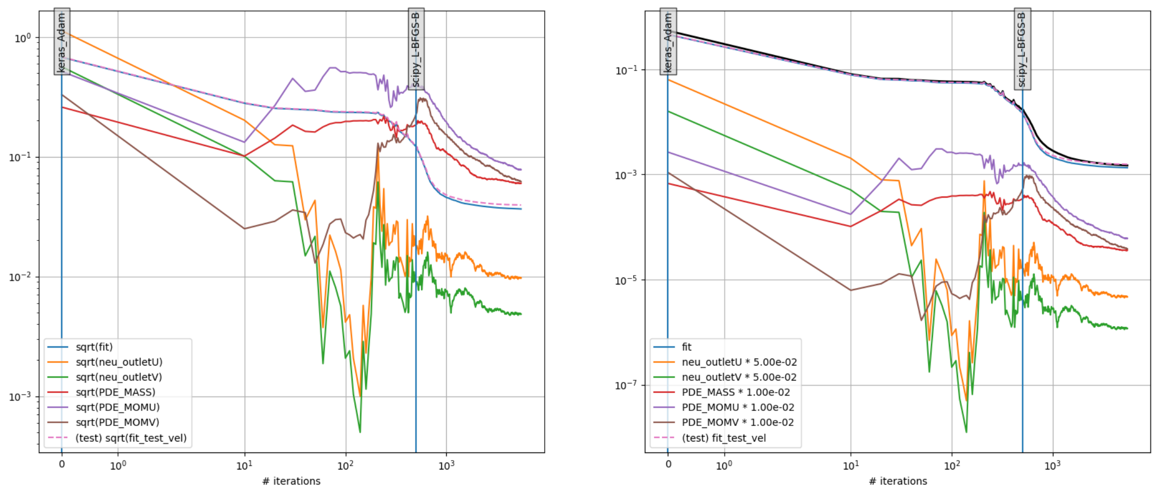

Figure 3.4: Loss evolution for the model presented in this section. On the left are visualized each loss term with their real magnitude before taking the square and loss' weights are not considered. On the right, each term is multiplied by its corresponding weight and here it can be seen their influence on the global loss value. PDE_MASS refers to the residual of the incompressibility constraint and PDE_MOM refers to the momentum NS equation.

decreasing. This means that, at a certain point, the net does not learn anymore from synthetic data, but it continues the regularization process through the governing PDEs. Starting from these results, we investigate the effect of increasing the relevance of physics terms starting from the beginning, or as soon as the data-fidelity term reaches a plateau. Both strategies have been tested, but no better results were achieved. More relevant physical terms lead to a flatter output since it is convenient for the net to propose a zero constant value during the minimization process. Physics terms need to be accompanied by another term that forces the presence of details in the velocity field. Even if those details are not accurate, they are required to avoid a zero constant output and, later on, accuracy will be taken care of by the action of the governing PDEs.

3.3.2. Visualization and performances evaluation

A further neural network is trained to only interpolate synthetic data and the physics is completely switched-off in the loss function. The PINN output is then compared to this other prediction to evaluate the added value of PINNs that force velocity and pressure fields to follow specific physical laws.
vNRMSE is computed taking as reference the reference CFD solution for both the PINN

and the interpolating network. The errors are reported in Table 3.2.

	Vel. X	Vel. Y	Pressure
PINN	0.056	0.118	0.045
Interpolation	0.103	0.194	1.765

Table 3.2: vNRMSE velocity and pressure errors.

As expected, pressure cannot be reconstructed and it is completely random in the interpolation-only model, as only velocities are seen by the data-fidelity term. Moreover, velocity fields are regularized by the PINN, and the noises introduced by synthetic data are removed. This is possible only relying on PDEs in the minimization process, as it is performed in physics informed neural networks.

Finally, in the following figures, the model prediction is visualized and compared with CFD solution and with synthetic data used during the training stage:

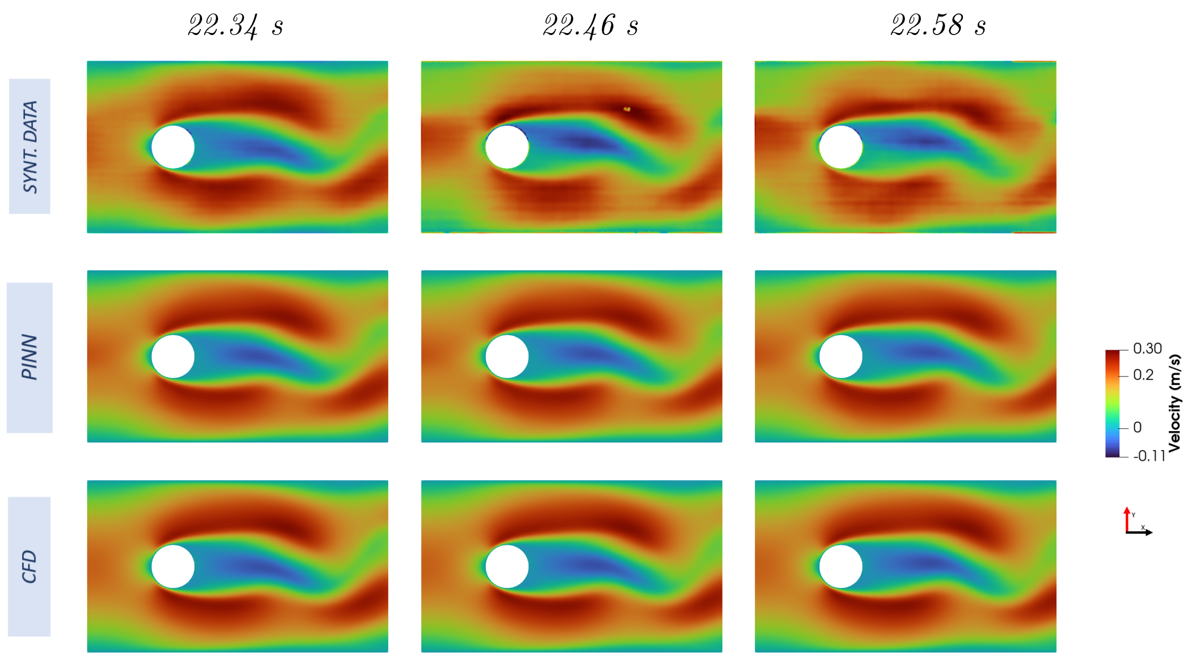

Figure 3.5: X velocity component. In the first row there are synthetic data in three timesteps, then there is the model prediction and the ground-truth CFD solution in the last row.

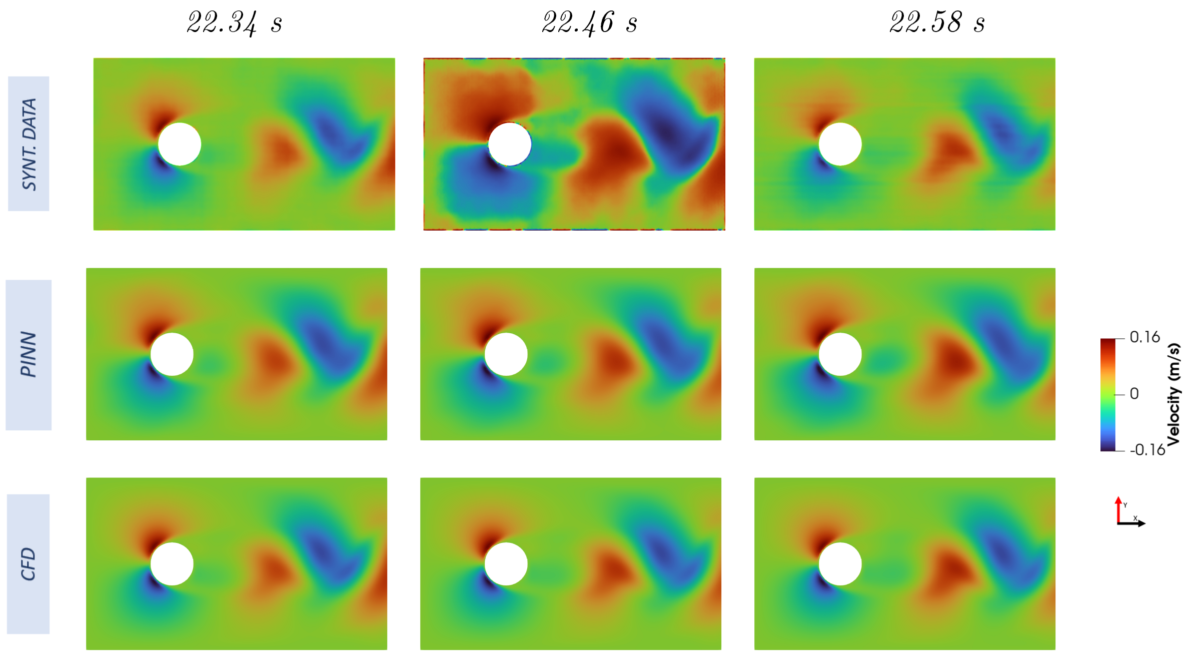

Figure 3.6: Y velocity component. In the first row there are synthetic data in three timesteps, then there is the model prediction and the ground-truth CFD solution in the last row.

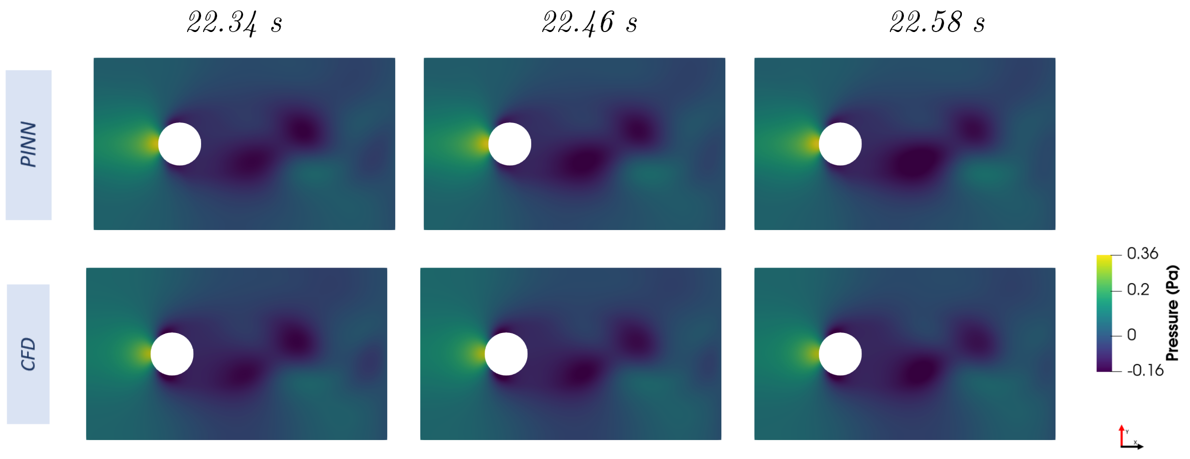

Figure 3.7: Pressure. In the first row there is the model prediction in three timesteps and the ground-truth CFD solution in the last row. No synthetic measurements are available for pressure that is completely reconstructed through the physics loss terms.

4 | Test Case 2: Synthetic data for a 3D flow in the aorta

In this Chapter, a real aorta geometry is taken into account to create a suitable model to regularize data and to estimate WSS. This is done in a controlled framework, as synthetic 4D flow data are used to train the model, and a CFD solution represents the reference ground truth.
The process to generate synthetic data is described in Section 4.2, where they are also compared to the CFD solution. The models used are characterized in Section 4.3 and there are compared two strategies to impose the no-slip condition. Finally, in Section 4.5, the methods used to compute WSS are evaluated based on a reference field achieved through an analysis on a smaller geometry of the domain.

4.1. CFD simulation

For the simulation we considere the ascending aorta of a subject with thoracic aorta aneurysm (TAA) [37], whose geometry is segmented from 3D MRA images using `ITK-SNAP` open-source software [46]. The segmented domain Ω is divided into 3 subdomains: inlet Γ_{in}, outlet Γ_{out} and wall Γ_{wall}. A 3D tetrahedral mesh with a base size of 0.6 mm is generated using `vmtk` library and the final volumetric mesh consisted of $\approx$ 800k nodes. Time-varying 3-directional velocity profiles are prescribed as inlet boundary condition, enforcing a realistic TAA inlet velocity on Γ_{in}. A zero-pressure condition is enforced on Γ_{out} and a homogeneous Dirichlet boundary condition (no-slip) is assumed on Γ_{wall}. Blood is modeled as a Newtonian fluid with constant density $\rho = 1060\frac{\text{kg}}{\text{m}^3}$ and dynamic viscosity $\mu = 0.0035$ Pa·s. A finite volume simulation is run at a fixed timestep of 0.001 s. Results are exported at every timesteps within the interval (0.2 s - 0.26 s) [37].
From now on, the analyzed area is reduced: the new geometry takes into account only the upper part of Ω, keeping only points with the foot-to-head coordinate ≥ 0.24 m. This is done to reduce the computational complexity of the problem, to shorten the training time of the model. Many tests with different conditions must be run and this simplification is

needed to make it feasible. With this reduction the definition of Γ_{in} is lost, but it is not required in PINN framework.
It is necessary to define the orientation of the classical Cartesian axes X, Y, Z within the aorta. The X axis refers to the posterior-to-anterior direction, the Y axis to the left-to-right direction and the Z axis refers to the foot-to-head one. Finally, to give a glimpse about the spatial dimensions, the reduced Ω could be contained in a cube with edge size of 0.06 m and it is visualized in Figure 4.1.

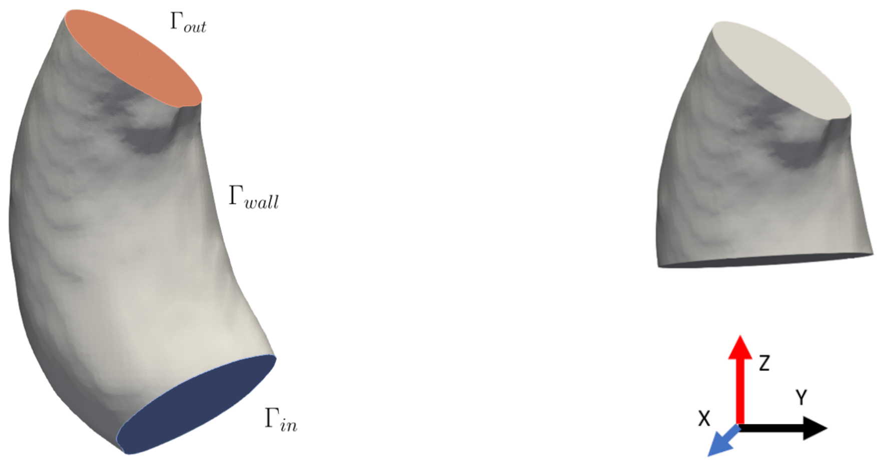

Figure 4.1: Original computational domain for CFD simulation on the left, and reduced geometry on the right.

4.2. Synthetic data creation

Three synthetic test cases with different parameters are created from a CFD simulation to evaluate the model performances with respect to the different levels of degradation and noise added. In all of them, not only there is a subsampling in the *k-space*, but additive noise is always considered, unlike in the 2D case.
The following steps are applied on high resolution CFD velocity fields:

1. data are temporally downsampled with a moving average to obtains measurements each 40ms, as this frequency is typical in 4d flow MRI measurements;
2. the velocity in each timestep is converted to an uniform Cartesian grid with voxel size of $VS \times VS \times VS$ mm^3 using a linear interpolation scheme to assign velocity vector values to grid cells. This procedure yields to a sequence of Cartesian grids where each element belongs to $\mathbb{R}^{N_r \times N_c \times N_s}$;

3. each velocity grid is converted to a complex tensor containing magnitude and phase images using suitable VENC values, as formalized in (1.1). VENC components are always chosen 10% larger than the maximum velocity recorded to avoid velocity aliasing issues;
4. the fast Fourier transform is applied to obtain the corresponding k-space data;
5. 3D k-space data is truncated in the high frequencies to effectively decrease the spatial resolution by a factor of 2;
6. a zero-mean Gaussian noise is added with standard deviation $\sigma = \sqrt{M/10^{\frac{SNR}{10}}}$, where M is the square of the mean magnitude of the signal, and SNR is the parameter related to the level of the noise added. The noise decreases when SNR increases;
7. a randomized radial sampling is performed to keep only $S\%$ percent of the *k-space*. Moreover, a fully sampled calibration region of $CR \times CR \times CR$ is kept in the center of k-space to maintain the main features of the signal;
8. the inverse Fourier transform is applied to the undersampled, noise-corrupted k-space, yielding a complex tensor of magnitude and phase images;
9. complex images are converted back to real images of velocity fields using VENC values consisted with step 3, obtaining a sequence of noisy synthetic velocity measurements.

A synthetic dataset is thus characterized by voxel size VS, the level of noise SNR, the sampled percentage S in the *k-space* and the size CR of the fully sampled center region.

	VS	SNR	S	CR
mild	0.3	40	99	5
medium	0.3	25	95	7
extreme	1	2	75	5

Table 4.1: Parameter description for each synthetic test case.

Two different Cartesian grids are used to generate different synthetic data: a very refined one with voxel of size $0.3 \times 0.3 \times 0.3$ mm^3 and a coarser grid with voxel of size $1 \times 1 \times 1$ mm^3. Data are sampled in those grids before moving to the Fourier domain. In the second case, once synthetic data are created, a nearest-neighbours interpolation is performed to

arbitrary sample points in every position inside the domain. Different strategies are used because, with the coarser grid, it is easier to vary the sampling percentage in the *k-space* but an interpolation, that could flatten some details, is required to have a reasonable sampling resolution. On the other hand, with a refined grid, it is difficult to reduce the sampling in the *k-space* in order to vary the quality of data and, consequently, tuning optimal parameters is not trivial and results are often unusable. Moreover, it is interesting also to differentiate visually the results: with the coarser grid the resulting data have the typical speckled behavior of real 4D flow data instead, the other data presents a different noise texture, more thin than the first one. The difference in resolution is also clear between the two grids.
Moving from this rough assessment, a rigorous evaluation based on magnitude and vector normalized-rootmean-squared-errors is performed using as reference the CFD solution, and the errors are reported in Figure 4.2. As expected from the chosen parameters, there are three different levels of degradation in the three synthetic cases considered: indeed, the NRMSE for the extreme case is 10 times larger than the mild one.

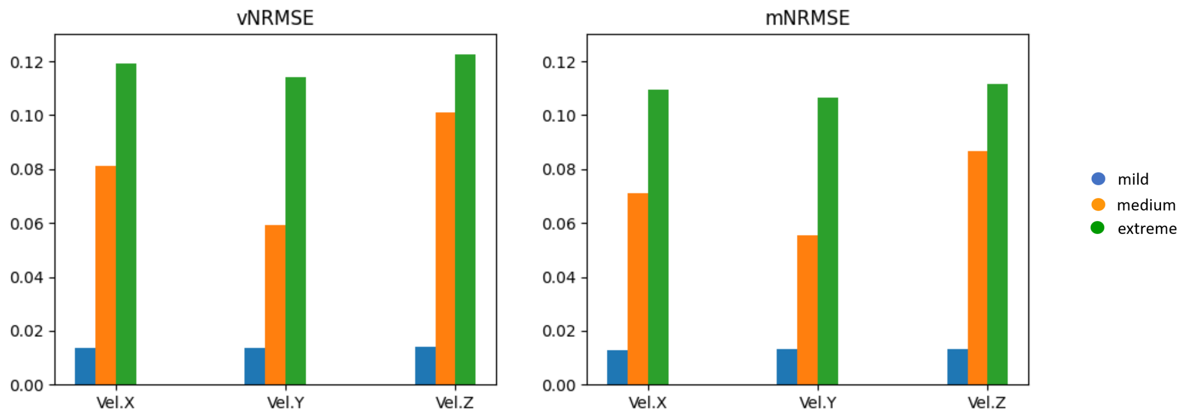

Figure 4.2: vNRMSE and mNRMSE are reported to quantify the level of degradation in the proposed synthetic data for each velocity component.

CFD solution together with all the synthetic data generated are represented in different 2D slices in Figures 4.3, 4.4 and 4.5. The first slice in each figure represents a sagittally oriented slice, the second a coronal oriented one and the last one an axially oriented slice.

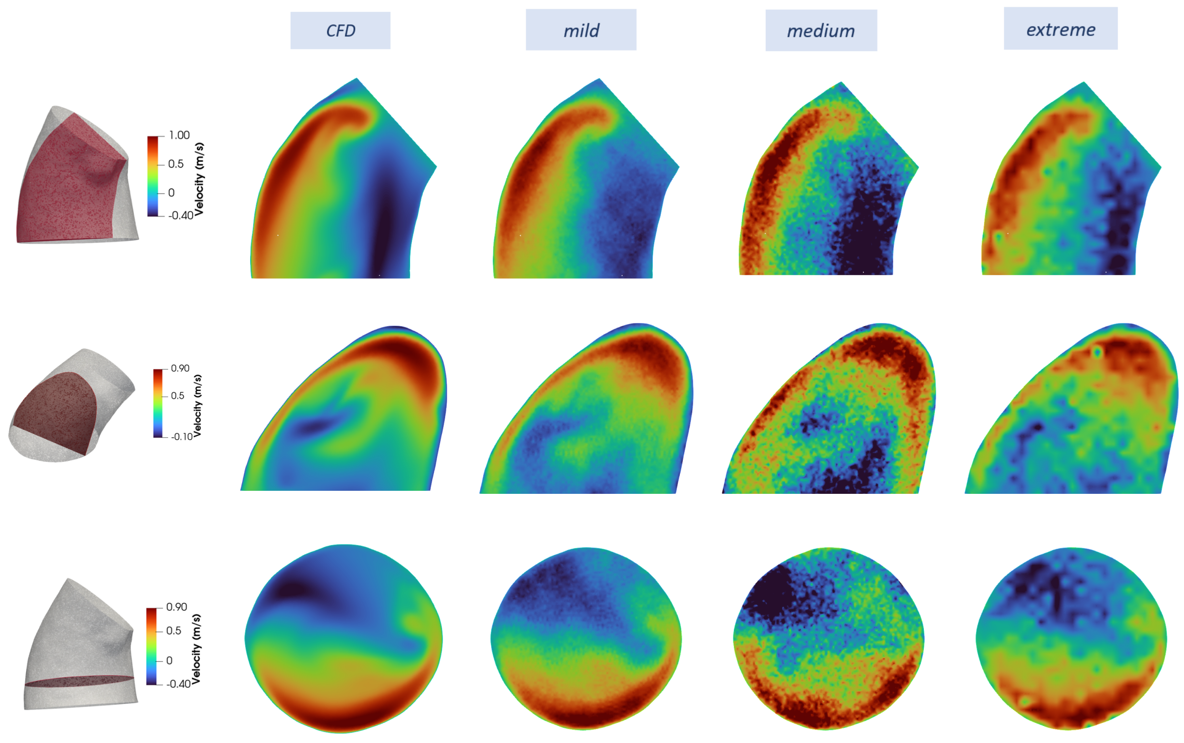

Figure 4.3: Velocity X component colormaps in different 2D slices at the same timestep.

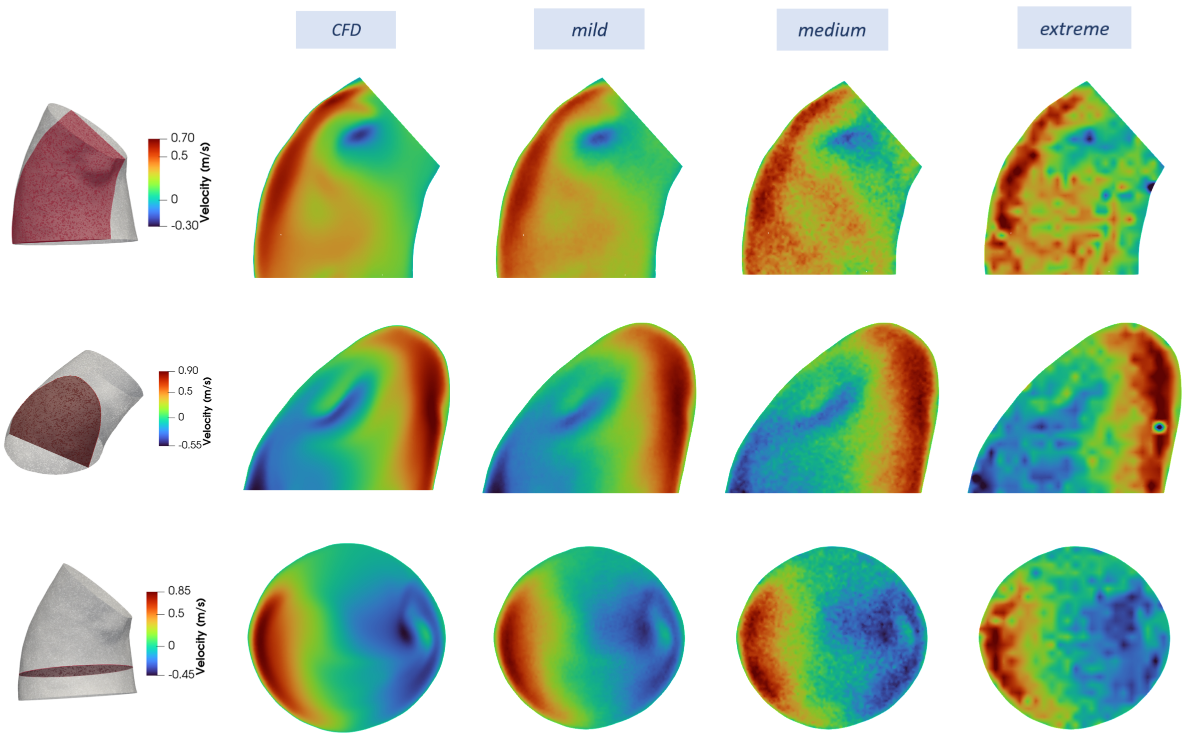

Figure 4.4: Velocity Y-component colormaps in different 2D slices at the same timestep.

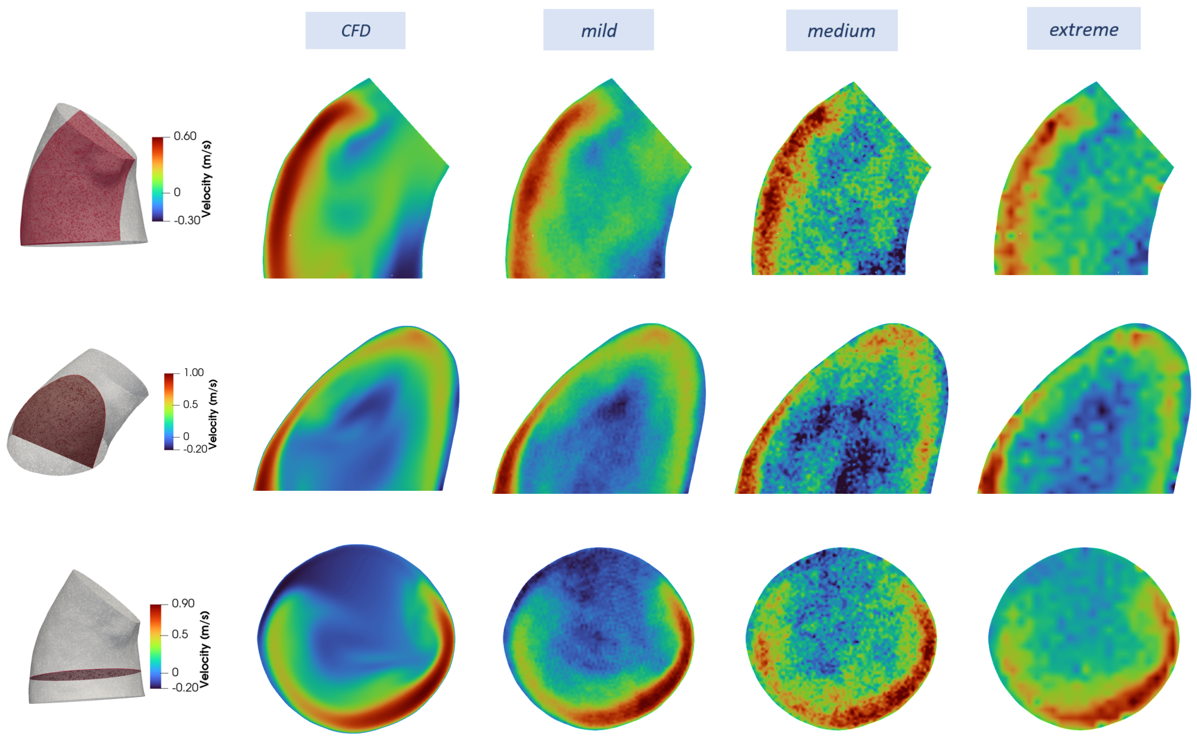

Figure 4.5: Velocity Z-component colormaps in different 2D slices at the same timestep.

4.3. Model description

To deal with the very detailed time dependent solution, the model for the 3D case is extremely enriched with more parameters, with respect to the 2D case, in terms of layers and neurons employed, while the model structure and loss terms remain similar.

4.3.1. Imposing the no-slip condition

The main difference with respect to the 2D case lies in the imposition of the no-slip condition at the wall boundary. With a simpler geometry that could be described as a set of edges, it was easier to follow the lifting approach with the continuous distance function described in Section 3.2.1 and in [3]. Here, instead, the geometry complexity forces to compute the distance from the wall with a distance matrix based approach. The distance between every point and a subset of wall boundary points is calculated, then the final distance from the wall, for a point p_i, is taken as the minimum of all the distances between p_i and every wall points. Alternatively with respect to the normal computational case, inlet and outlet points are not considered as zero wall distance points, as the no-slip

condition is not imposed.
At this stage there are two alternatives to impose zero velocity at the wall boundary:

- train a further neural net to learn the distance from the wall ϕ starting from the distance matrix, following the same procedure applied in the first step of normals computation (Section 2.2.1). Then, multiply ϕ by the velocity output of the net simulating the lifting procedure as done in Section 3.2.1;
- directly impose a zero wall velocity by adding a term in the loss function that forces a zero output velocity for a subset of wall points.

The first method has the advantage to end up with a simpler loss function that could help the minimization process. On the other hand, the loss' gradients with respect to the lifting ϕ could lead the optimizer to a wrong direction. This is because there could not be enough confidence that the lifting neural network is continuous, and that it will surely return an accurate distance from the wall. The tests performed shows that the best strategies is to add a zero wall velocity term in the loss function. Even if the lifting is correctly computed, and it assumes a near zero value at wall points, the velocity tends not to be exactly null. The output of the main net before the lifting layer tends to be very high and even if it is lowered by ϕ, it does not assume a zero value. This happens since there is nothing that penalizes a non-zero wall velocity and ϕ does not assume an exact null value. A possible solution could be to use both the ideas: the model could be potentially helped by the lifting action because ϕ tends to lower wall velocity, and a further term in the loss function could conduct the model to the final desired output. In this chapter we consider only the wall loss term, reserving further investigation for future studies.

4.3.2. Model parameters

A reference architecture is found by training an interpolation-only neural network with CFD data, without any imposition of physical laws, and the goal is to obtain an architecture that could represent the complexity of the velocity field. Eight hidden layers composed by 32 neurons each are chosen, and altogether the model consists in 7684 parameters. A sinusoidal activation function is employed, weights are initialized as described in Section 1.2 and there is the first layer adjustment.
The tuning procedure of loss terms weights yields to a general main role for the data-fidelity term, as happens in the 2D case. There remain two choices for the wall weight that will be discussed during the WSS analysis, as the velocity profile near the wall is

heavily influenced by the force used to impose a zero wall velocity. Finally, Neumann loss weight is two order of magnitude smaller than the data fidelity one. When its relevance increases not only the outflow points, but also the surrounding area tends to assume a zero pressure value. This does not respect the physics behind the problem but follows the typical behaviour of the network: it flattens out all details to a constant value that still satisfies the PDEs.
To summarize, the typical model architecture and training strategy for this test case can be completely characterized by those features:

- model architecture: 8 hidden layers with 32 neurons each, and 4 neurons in both input and output layers;
- sin(·) as activation function;
- first minimization loop: 500 epochs with Adam optimizer (learning rate: 0.01);
- second minimization loop: 3000 epochs with L-BFGS optimizer;
- 20k collocations points where the physics loss terms are evaluated and 20k data-fidelity points both spread in the entire time interval are considered;
- loss weights are reported in Table 4.2:

	w_{fit}	w_{mass}	w_{mom}	$w_{neumann}$	w_{wall}	w_{reg}
weights	1	0.1	0.1	$1 \cdot 10^{-4}$	0.1 or 1	$1 \cdot 10^{-5}$

Table 4.2: Loss weights for 3D synthetic test cases.

4.4. Results

In this Section we present the loss evolution for the best model and its velocity and pressure reconstructions.

4.4.1. Loss evolution

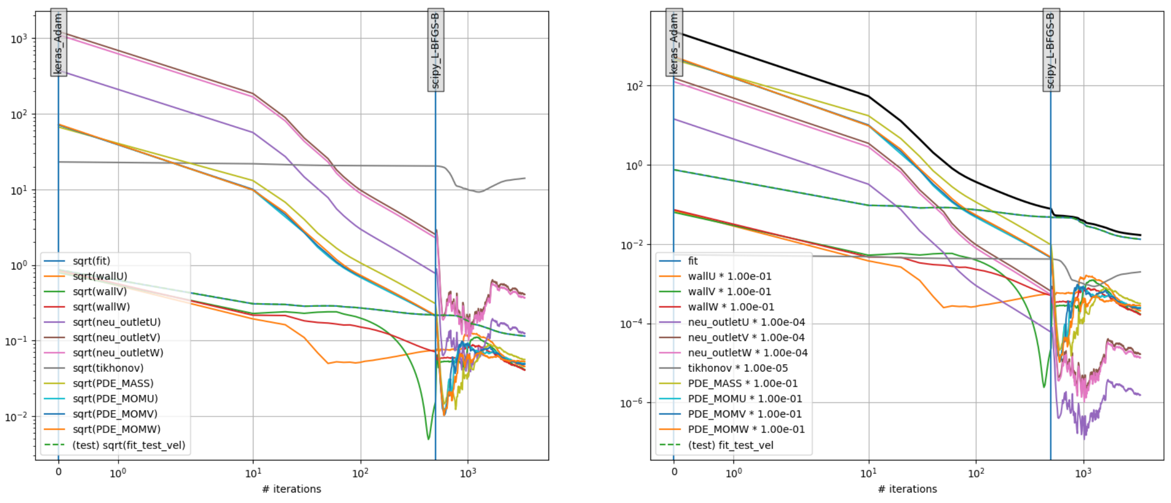

Figure 4.6: Loss evolution for the model trained with medium synthetic data. On the left are visualized each loss term with their real magnitude before taking the square and loss weights are not considered. On the right, each term is multiplied by its corresponding weight and here it can be seen their influence on the global loss value. PDE_MASS refers to the residual of the incompressibility constraint and PDE_MOM refers to the momentum NS equation.

The behaviour is very similar to the 2D case. Data-fidelity maintains the main role and the PDEs residual are kept low. All terms have not reached a plateau and the minimization process could continue probably yielding to better results. Indeed, in all the performed tests there is the possibility to further improve the performances, but due to the computational time request, only 3000 L-BFGS epochs are carried out.

4.4.2. Velocity and pressure reconstruction from different noisy synthetic data

Different models are trained for each synthetic case and, despite the clear difference between the three level of degradation, the predictions are similar, as PINNs succeed in

regularizing the input data. Indeed, in the first row in Figure 4.7, there is a maximum error difference of 4% between the three cases for each component.
To evaluate the performances, more independent training runs are performed for each synthetic case since the net output is not deterministic.

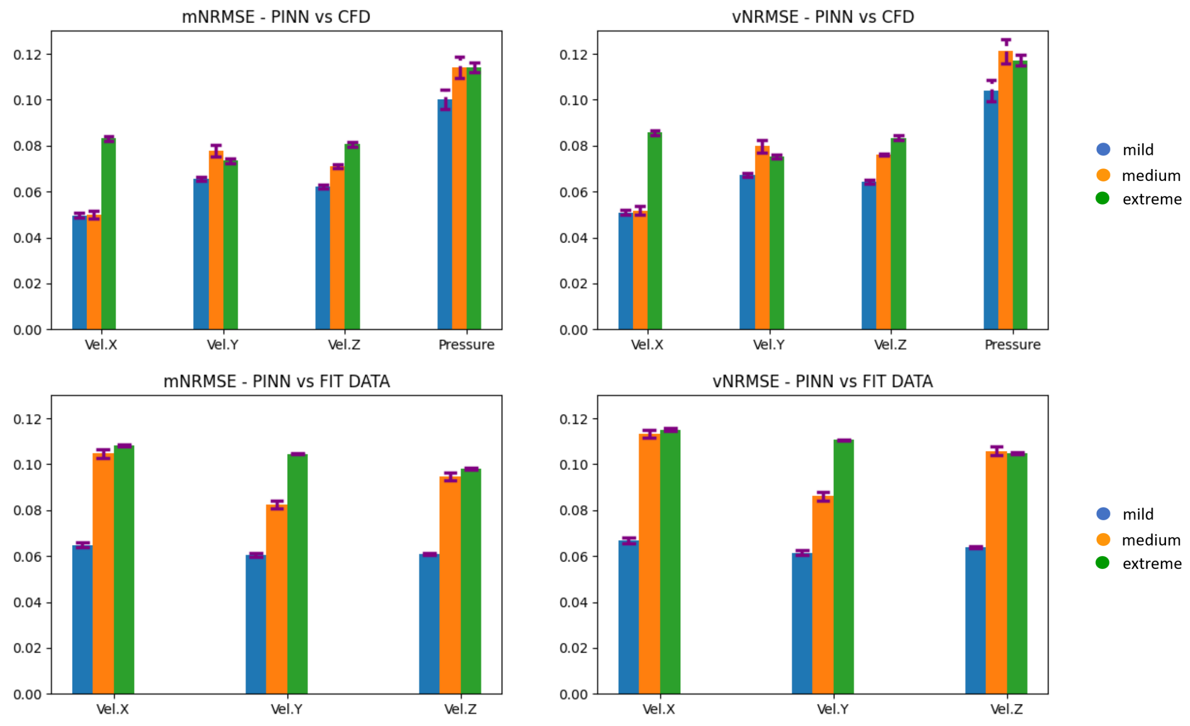

Figure 4.7: Errors for models trained on different synthetic cases. Three independent training runs for each model are performed and it is represented the mean error with the standard deviation (purple line above the bars). In the first row, there are the errors between the model prediction and the reference solution, while on the bottom, there are the errors with respect to fitting data (synthetic data).

As expected, slightly better performances (first row in Figure 4.7) can be detected for the mild case in which the level of degradation is very limited. It is interesting to note that the error with respect to the input synthetic data increases with the level of noise considered (second row in Figure 4.7). The model moves further away from the fitting data when they are very corrupted by noise. Even if the weight of the data-fidelity term is the same, the net learns to not focus on the degraded data. This result suggest to not modify the data-fidelity weight for cases in which input data are very noisy.
Another credit on PINNs is the correct pressure reconstruction: Navier-Stokes equations permit to build the pressure field effortlessly in addition to the regularization process of velocity fields.

In general, the model succeeds in reconstructing the main features of velocity and pressure but details, such as vortices visible in the reference CFD solution, are lost. The net tends to smooth the output as it is convenient for the physics loss terms in the minimization process. This can be better seen when the degradation level increases (extreme synthetic case): since the input data are very noisy, the net learns to rely more on physics terms yielding to a flatter output. For pressure this is even more evident in all cases as the reconstruction is based only on the governing PDEs.

In the following figures the same 2D dimensional slices considered in Section 4.2 are used to visualize the predicted output. In each row, corresponding to a different slice, there are presented the CFD ground-truth, the medium and the extreme synthetic cases. In the CFD column there is the reference solution, in the medium and extreme columns there are the synthetic input data used during training on the left and, alongside, there is the model super-resolved prediction.

In Figures 4.8, 4.9, 4.10 there are visualized the X-, Y-, Z- velocity components. The velocity reaches a maximum value of 1 $\frac{\mathrm{m}}{\mathrm{s}}$ in the CFD simulation, while the PINN prediction reaches a value 10% lower. The pressure drop between inlet and outlet is visualized in Figure 4.11, and it decreases in the model reconstruction, as it cannot reach the CFD value of 200 Pa.

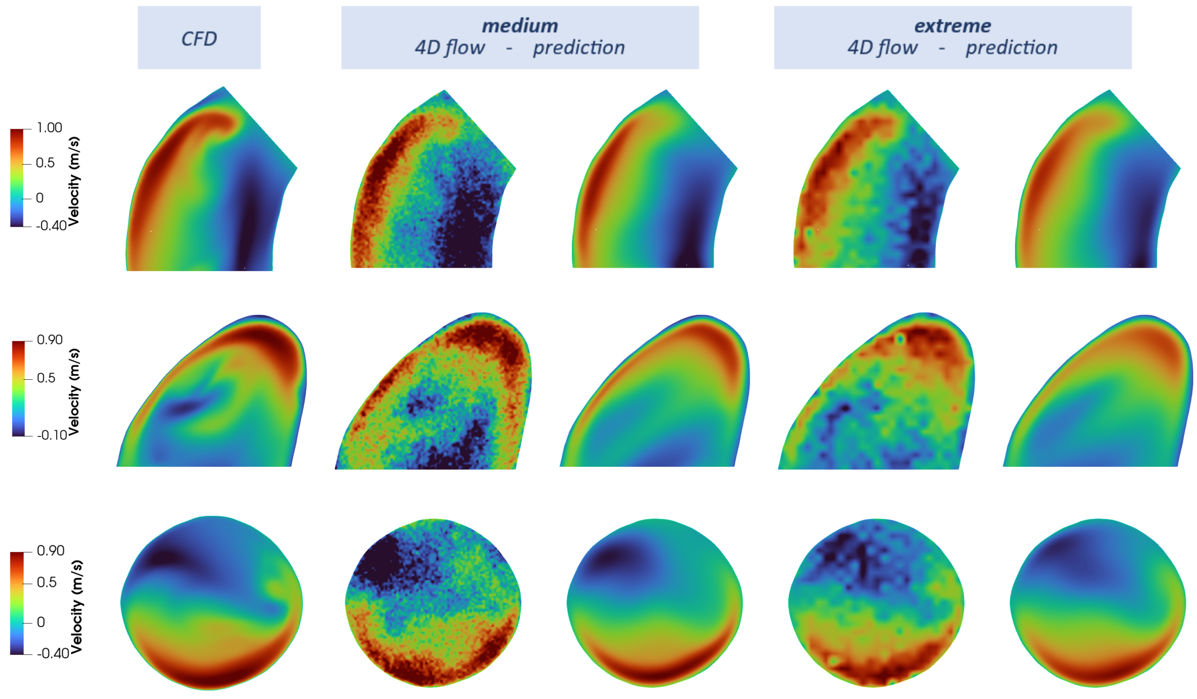

Figure 4.8: X-component velocity colormaps in different 2D slices at the same timestep.

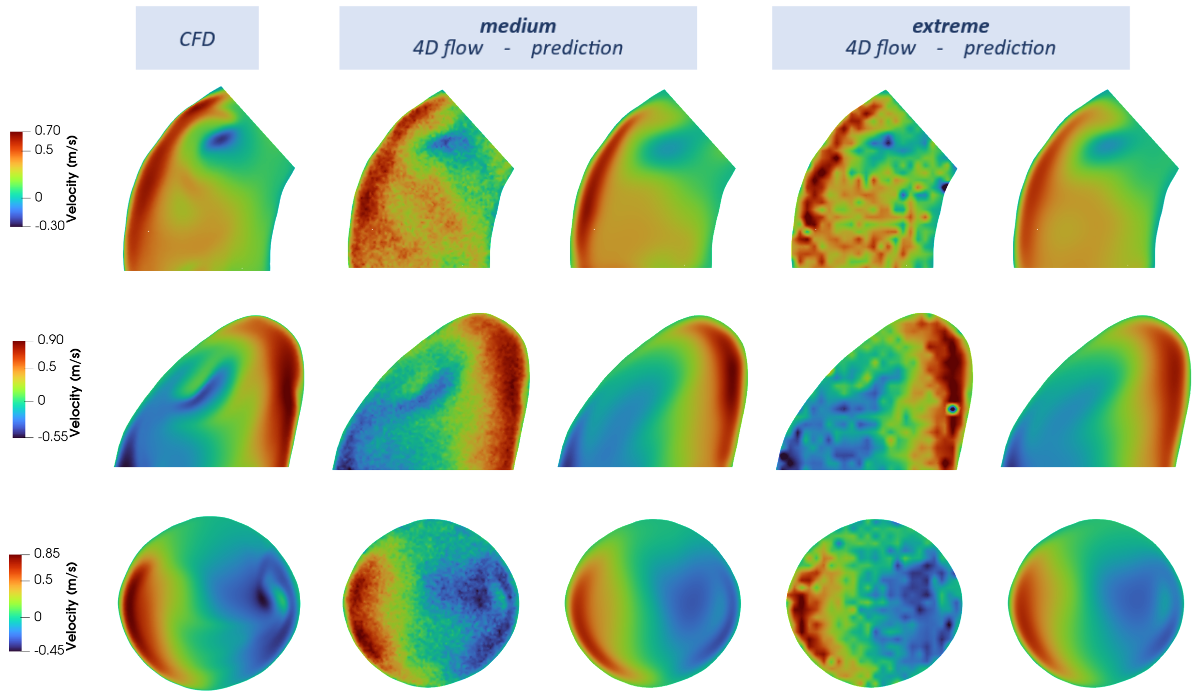

Figure 4.9: Y-component velocity colormaps in different 2D slices at the same timestep.

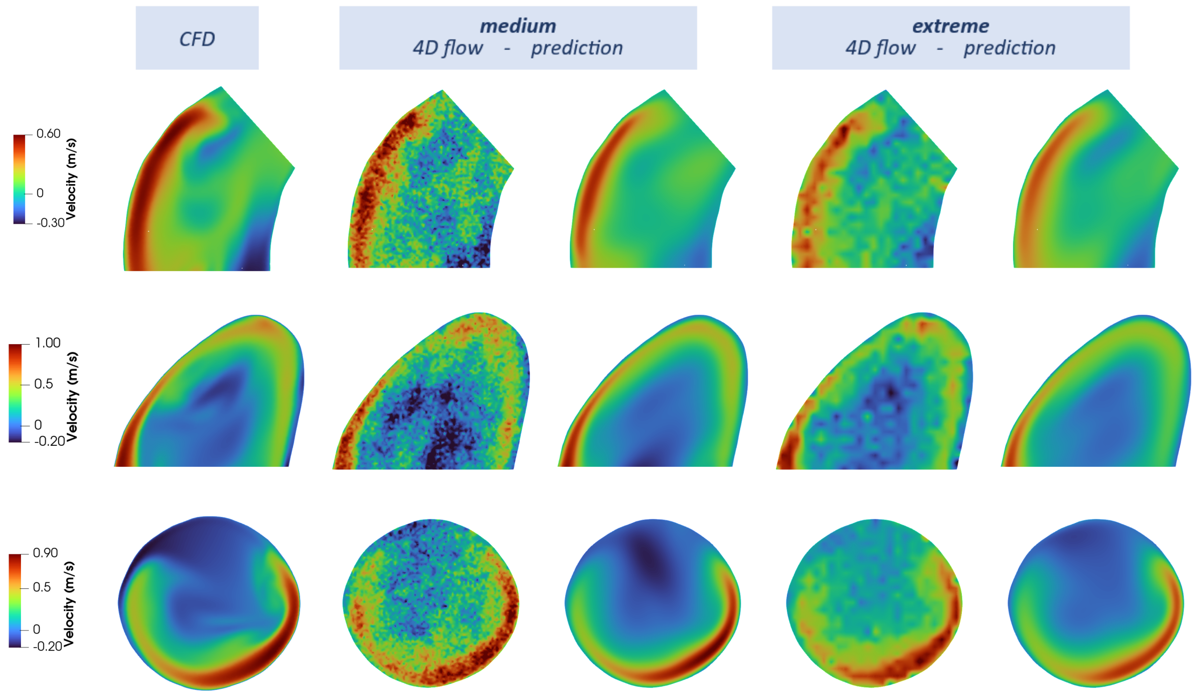

Figure 4.10: Z-component velocity colormaps in different 2D slices at the same timestep.

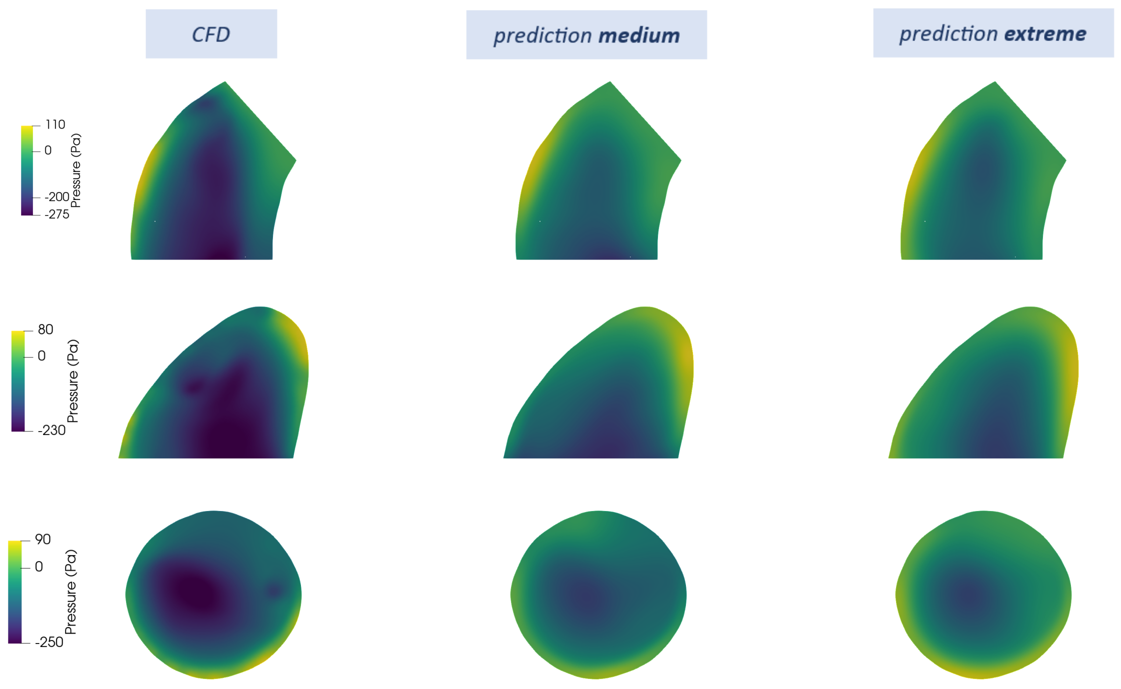

Figure 4.11: Pressure reference and reconstructed colormaps in different 2D slices at the same timestep.

4.5. Wall Shear Stress analysis

In this section, we investigate different strategies for computing wall shear stress. The velocity profile in the near wall region could be influenced by how many collocations point are sampled and by the wall loss term weight w_{wall}. Moreover, WSS can be computed through AD or with the parabolic fitting method. The latter introduces another hyperparameter d_n that represents the distance from the wall of the fitting points. A ground-truth WSS field is needed to evaluate the performances with respect to those parameters, and this is obtained by considering a test case, reported in the following, that restricts the investigation to a cube Λ of dimensions 1 cm^3 embedded in the geometry (Figure 4.12).

4.5.1. WSS on a reference cube

A further interpolation-only neural network is trained with CFD data to represent the velocity field in Λ. Since an intense training, in terms of epochs and data used, is carried out, the net provides a very precise velocity field leading to an accurate WSS computation through AD. Λ has a contained size due to computational costs, indeed, in this small region it is possible to perform such an accurate training in a feasible time.

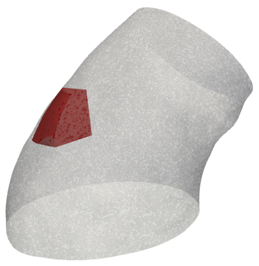

Figure 4.12: Λ represented together with the reduced reference CFD geometry.

Firstly, to test the network expressiveness, different architectures are employed to understand how many parameters are required, at least, for representing Λ's complexity. Four architectures all composed by 5 hidden layers and, respectively, 5, 10, 20, 40 neurons each, are considered, and they are all trained for 15k L-BFGS epochs. Their performances are evaluated with respect to CFD ground-truth in Λ and they are reported in Figure 4.13. It turns out that, except for the first case, all the architectures succeed in well representing the velocity field, and thus, at least 10 neurons in 5 hidden layers are required. From now on, it is chosen to continue the analysis with the bigger net (5 layers $\times$ 40 neurons).

With this optimal architecture, a first reference WSS field is already available as AD accurately estimates WSS in Λ. Trough this ground-truth field it is possible to obtain a second reference WSS: it is computed with the parabolic fitting method starting from CFD data, and the optimal value for d_n is tuned relying on the first WSS reference.
We compare three choices for d_n ($2 \cdot 10^{-4}$ m, $5 \cdot 10^{-4}$ m and $7 \cdot 10^{-4}$ m) to use within the parabolic fitting method. The velocity data come from the net prediction or the CFD solution where a nearest neighbours interpolation is performed, since only mesh points can be used. These two estimates are compared to the reference one computed through AD, and they are visualized in Figure 4.15. The mNRMSE for the estimates provided are reported in Figure 4.14.

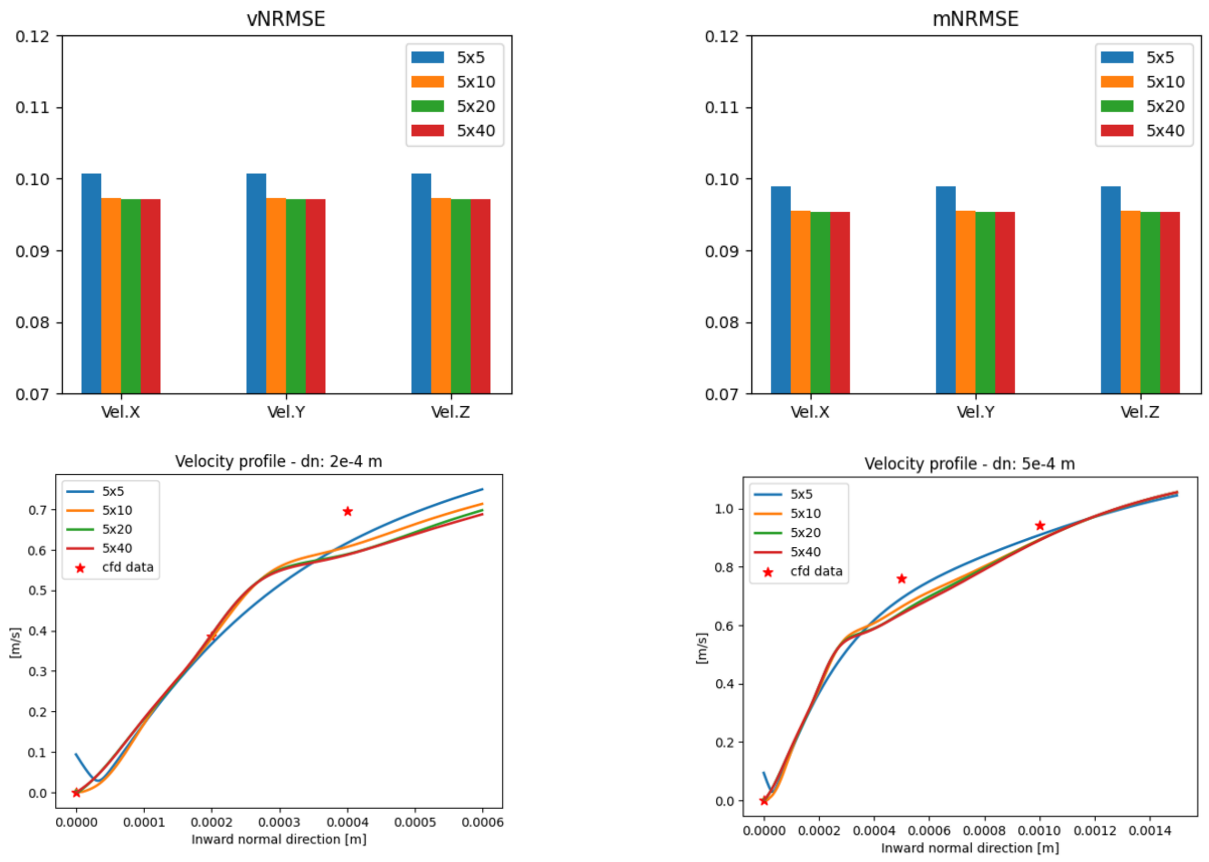

Figure 4.13: Velocity errors for the different models architectures. In the second row, there are represented 2D plots of velocity profiles starting from a wall point and going inward in the normal direction.

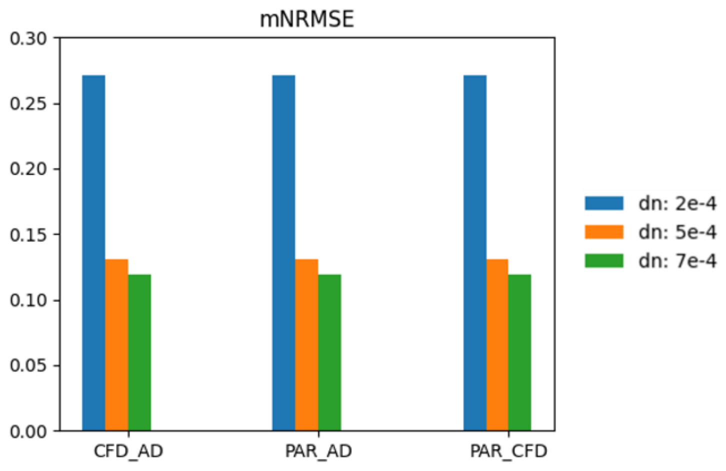

Figure 4.14: mNRMSE with respect to different d_n choices. About notation in this figure: A_B means that A and B are compared taking as reference B.

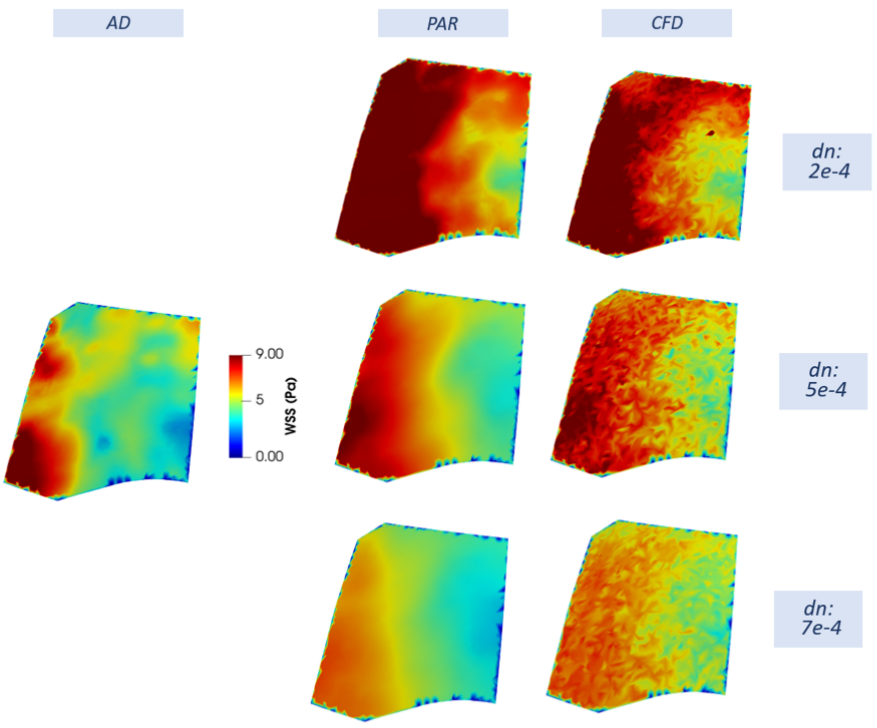

Figure 4.15: On the left there is the reference WSS field computed with AD. In the second column there are the ones calculated with the parabolic fitting method starting from points evaluated by the net. In the last column there is the WSS field computed with the parabola and the CFD solution. The rows refer to distinct d_n choices.

The detailed velocity profiles at the boundary are completely captured by the interpolating net and, consequently, WSS computed through AD represents a precise reference. A very similar result is proposed by the parabolic methods that uses a large d_n: relying more on points far away from the boundary permits avoiding looking at inaccuracies present in the near wall region, where the net suffers in capturing high frequencies. Instead, when using $d_n : 2 \cdot 10^{-4}$ m, the parabola recreates a very steep profile, reaching too high values for WSS and severe errors as reported in Figure 4.14.

In Figure 4.16, we represent the profiles of the module of the tangential velocity that are used by the parabolic methods. There is also the one obtained by the neural network's prediction, and it must be noted that this is not the one used by AD, as it does not consider the velocity module. Finally, the stars indicate the CFD reference points obtained through interpolation when they are not available at that specific coordinate. If the mesh points used for the interpolation are too far away from the required point, then a lower color intensity is applied.

In Figure 4.17, there are the corresponding derivative profiles for the first point chosen. The blue profile represents the derivative computed through AD of the magnitude of the tangential velocity and, even if the WSS estimate through AD does not consider the magnitude, the derivative at the wall is the same. Hence, it could be used to visually compare what AD computes with the estimates obtained from the parabolic profiles: there is more accordance when using a large d_n.

This analysis concludes that WSS could be exactly computed through AD only if the net proposes an accurate velocity in the near wall region. Moreover, CFD reference data show that the velocity does not present a clear parabolic profile, thus AD is preferable since it does not super-impose a specific profile. However, with a large d_n, very similar velocity gradients are estimated.
In more challenging test cases, where the net has poor accuracy near the boundaries, it will not be convenient to rely on AD. An acceptable result could be recovered with a parabolic method that uses a large d_n to avoid looking at the problematic near wall region. This will be discussed in the next sections considering synthetic and real test cases.

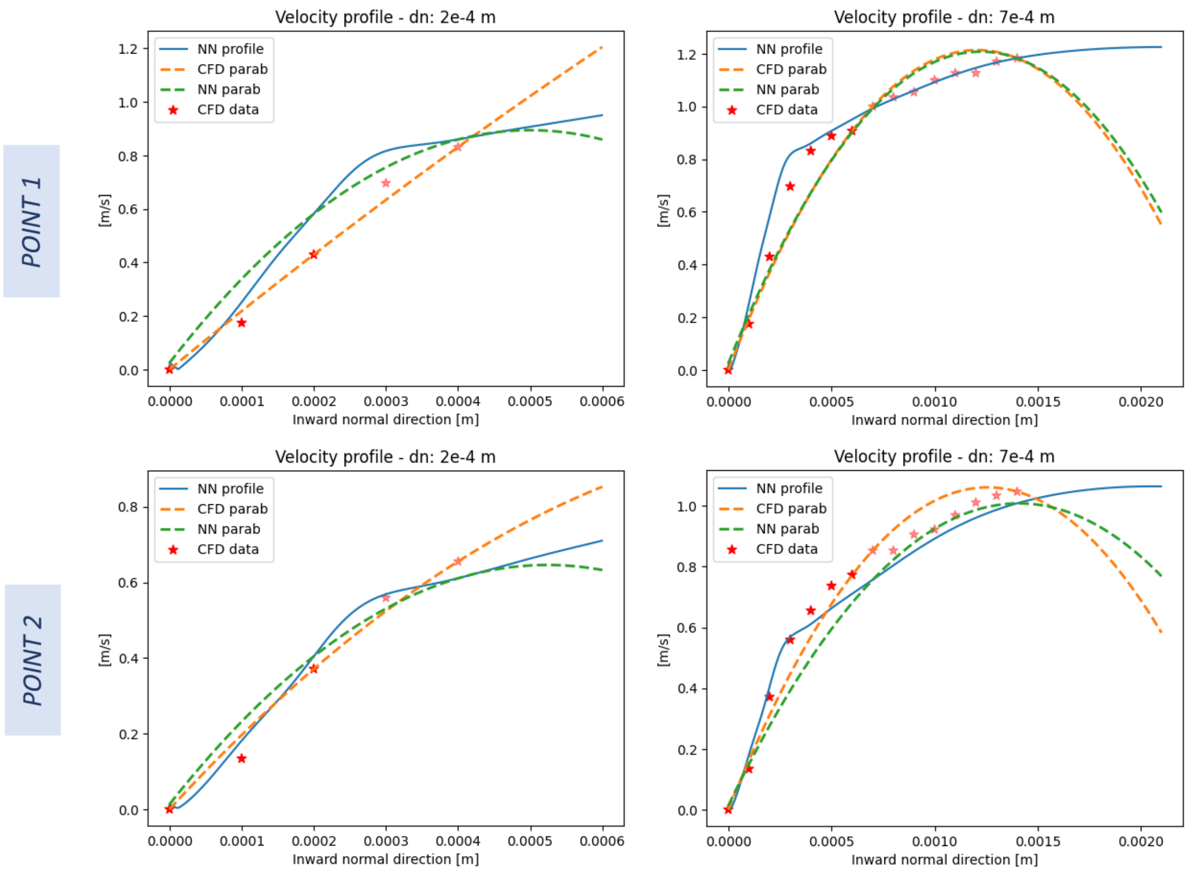

Figure 4.16: In each row there are represented 2D velocity profiles for different wall points in Λ. In each column there are different choices for d_n.

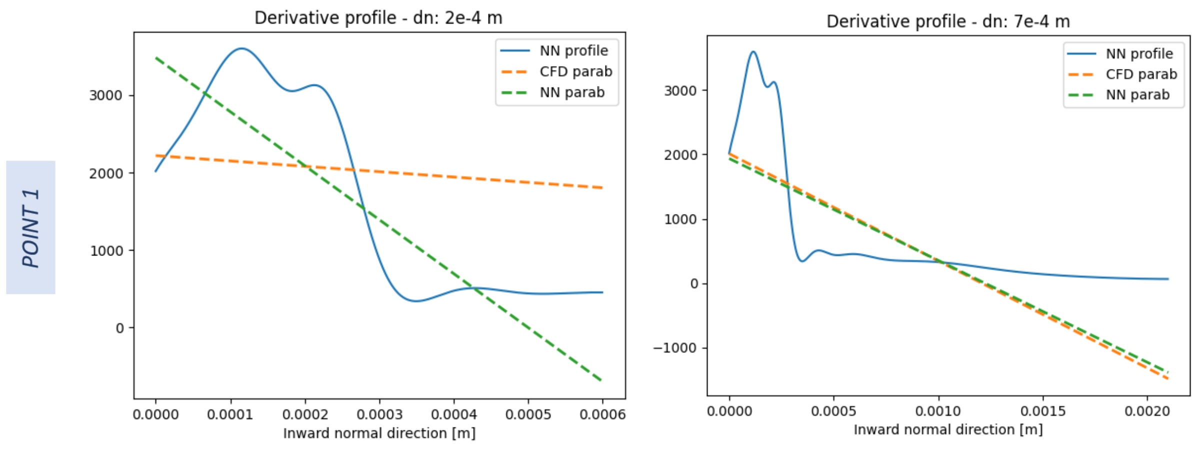

Figure 4.17: Velocity derivative profiles for the first wall point represented in Figure 4.16. In each column there are different choices for d_n.

4.5.2. Analysis on synthetic data

Four different models are now evaluated in terms of velocity and pressure reconstruction, and in terms of WSS estimate. For the latter, there are used two references obtained in the previous section: WSS computed with AD in Λ and WSS calculated in the entire geometry starting from CFD solution with a parabolic fitting method with $d_n = 7 \cdot 10^{-4}$ m. The proposed models are all trained with the same synthetic dataset, since this analysis wants to identify differences between them, then cleaner synthetic cases will provide better performances.

The number of collocation points in the near wall region and the weight w_{wall} in the loss function are the two degrees of freedom that differs in the following models:

	w_{wall}	More wall collocation points
W01	0.1	×
W1	1	×
W01_{coll}	0.1	✓
W1_{coll}	1	✓

Table 4.3: Models evaluated in this analysis. When more wall collocation points are required, 15% of the total number of collocation points required are surely sampled in the near wall region.

The neural network trained to learn wall distances for normals computation (Section

2.2.1) is used to identify the near wall region. If a more intense sampling of collocation points is needed, at least 15% of them are sampled within the near wall region (distance $\phi \leq 0.0035$), and the remaining 85% points are randomly sampled in the whole domain.

mNRMSE and vNRMSE are calculated to evaluate the velocity and pressure reconstruction in the whole domain, and no significant differences can be seen between the four models, as can be seen in Figure 4.18.

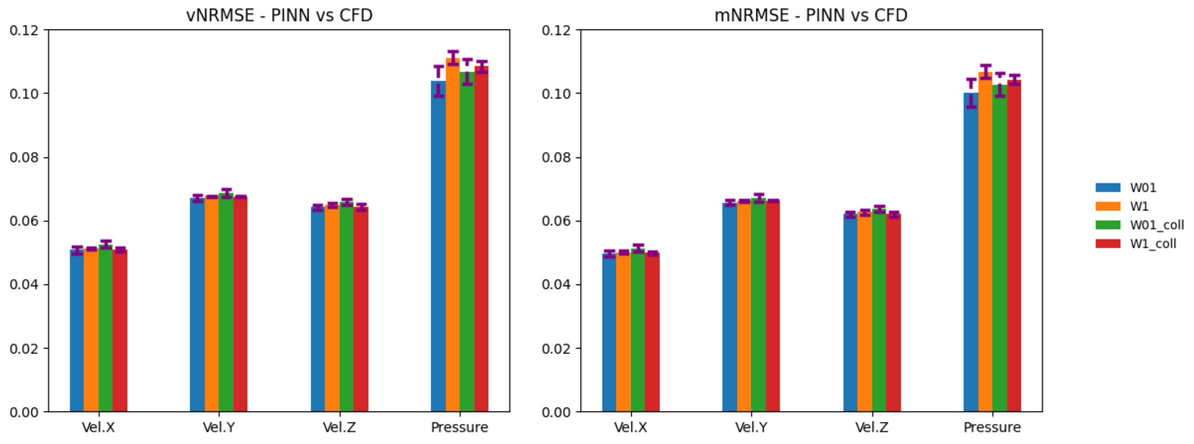

Figure 4.18: Velocity and pressure errors. Three independent training runs for each model are performed and it is represented the mean error with the standard deviation (purple line above the bars).

The real discriminant among those models is WSS computation starting from the predicted velocity. For each model, five methods are exploited to compute the normal derivative:

- AD: automatic differentiation;
- PAR2, PAR7: parabolic fitting method with $d_n = 2 \cdot 10^{-4}$ m and $d_n = 7 \cdot 10^{-4}$ m, respectively;
- PAR2 ZERO, PAR7 ZERO: parabolic fitting method with the two choices for d_n, but the first point used to fit the parabola is forced to be null.

Errors with respect to the two references for WSS are reported in Figure 4.19, together with the velocity profile in an arbitrary point of the domain (Figure 4.20). For the latter, it is taken a wall point and, going inward in the normal direction, the magnitude of the tangential velocity is represented in a 2D plot. Moreover, in Figure 4.20, there are also drawn the parabolic profiles used in the described methods.

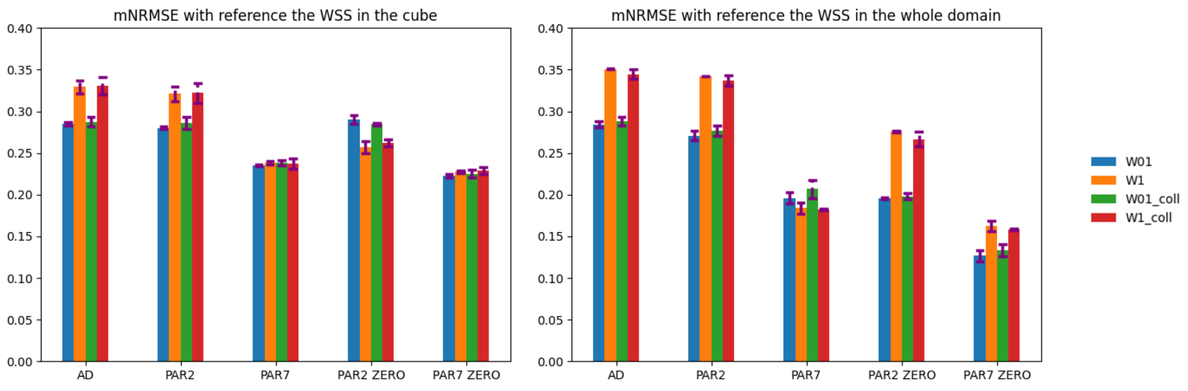

Figure 4.19: Errors for different methods exploited to compute WSS. Three independent training runs for each model are performed and it is represented the mean error with the standard deviation (purple line above the bars).

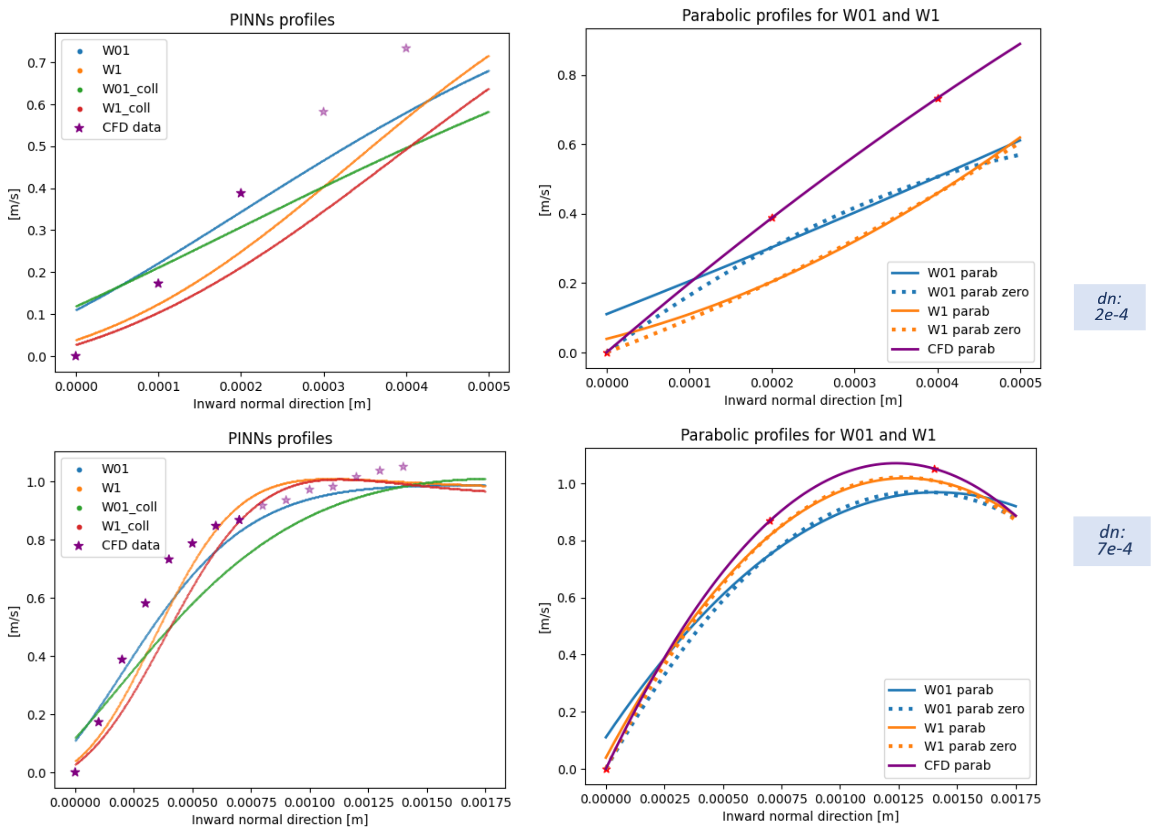

Figure 4.20: PINNs and parabolic profiles for the proposed models starting from an arbitrary wall point. In each row there is a different choice for d_n. In the second column, there are represented the parabolic profiles fitted starting from three points evaluated in the corresponding models (only for W01 and W1). The dotted profiles refer to the parabola forced to start from zero.

Before discussing the methods exploited, the different models are analyzed. A more intense sampling in the near wall region does not introduce any improvement and the result is very similar to the standard case (W01). The profiles for W01$_{\text{coll}}$ and W1$_{\text{coll}}$ are lower and smoother: implicitly relying more on physics, with more collocation points, ends up in a flatter profile, as a constant velocity is favored by the loss. About the no-slip condition, increasing w_{wall} succeeds in predicting a lower value at the wall boundary, but it does not improve WSS estimate. This is because the velocity profile arrive at the wall with a smoother profile and, in some cases, the concavity is inverted. The initial idea was to force more the no-slip condition to end up with a steeper profile, but the results show the opposite behaviour, yielding to severe errors in almost all the cases, as reported in Figure 4.19. Hence, the baseline model (W01) remains the best one.
Great differences could be seen when computing WSS with AD or through the parabolic fitting method. PINNs trained in the whole domain suffer in correctly reconstructing the gradients in the near wall region. Differently from what is done in Λ in Section 4.5.1, the predicted flat profiles are not adequate to estimate WSS through AD. To recover the high gradients at the boundary, it could be used a parabola forced to start from zero: this strategy super-imposes a profile, but it relies also on the model prediction when taking the inner fitting points. To confirm the advantages of this approach, errors in PAR ZERO are lower both in Λ and in the whole domain (Figure 4.19). However, when taking as reference the CFD WSS in the whole domain, there is a bias in PAR7 ZERO since there is the same choice for d_n.
Finally, d_n heavily influences the fitted parabolic profile, and thus, the WSS estimate. Taking a larger lens with $d_n = 7 \cdot 10^{-4}$ m permits considering innermost points that are more reliable. They succeed also in recovering a steep velocity profile at the wall, since the velocity magnitude is significantly higher going inward in the domain. On the other hand, with a lower d_n, the estimates are extremely variable: small areas with very high WSS value are surrounded by regions characterized with a lower magnitude. This noisy behaviour is something unrealistic that is not expected.
In general, regions that present a high WSS field are always detected, but WSS is always underestimated by all methods except for the noisy and variable PAR2 ZERO. However, all of them succeed in extremely improving the result obtained from raw and unprocessed 4D flow data. As a conclusion, the parabolic fitting method with $d_n = 7 \cdot 10^{-4}$ m is the more robust choice that could recover a precise estimate even when the prediction is not accurate at the wall. Indeed, AD is not trustworthy in those situations. The result are visualized in Figure 4.21 and in Figure 4.22.

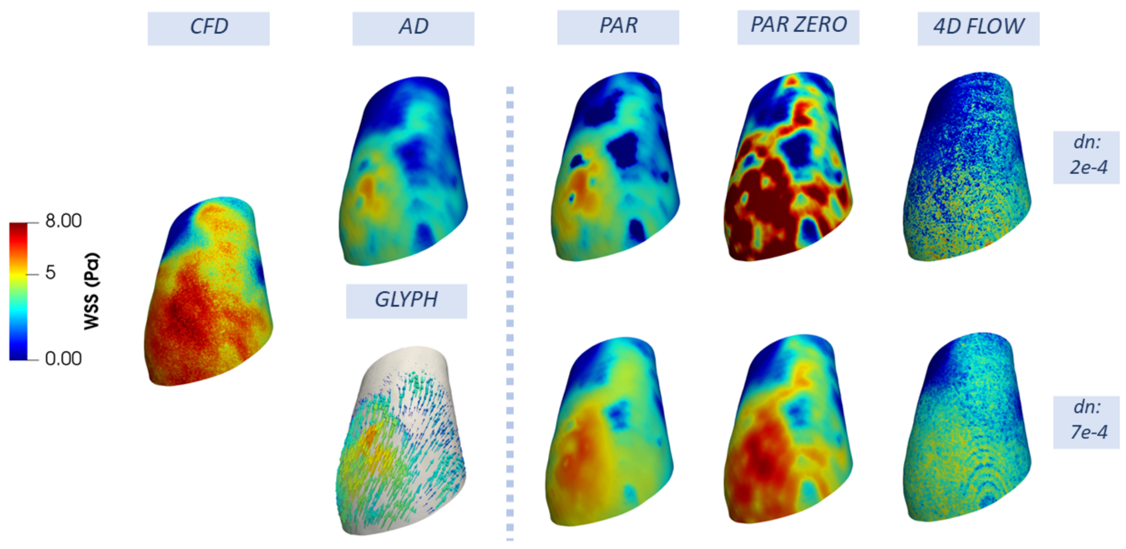

Figure 4.21: WSS estimate for mild synthetic case. There are represented the results using all the described methods, with different d_n choices. In the last column, there is the estimate starting from synthetic data, thus without the PINN's action. There is a vectorial representation only for the WSS field computed through AD.

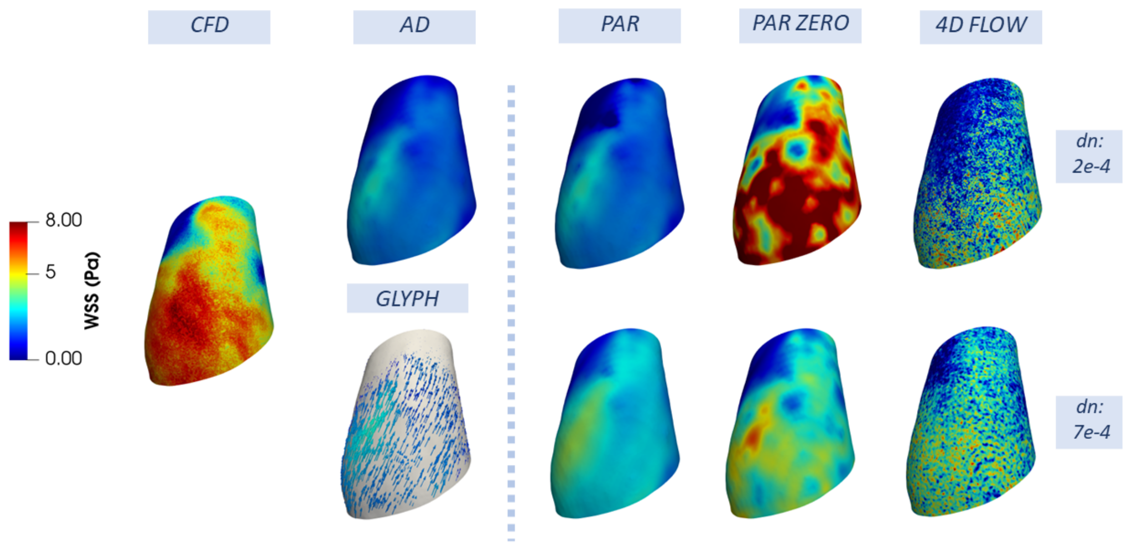

Figure 4.22: WSS estimate for medium synthetic case. There are represented the results using all the described methods, with different d_n choices. In the last column, there is the estimate starting from synthetic data, thus without the PINN's action. There is a vectorial representation only for the WSS field computed through AD.

5 | Test Case 3: 4D flow *in-vivo* data in the aorta

In this Chapter, we apply the model structure, studied within the synthetic controlled framework on real 4D flow data. We analyzed also patients with bicuspid aortic valve: for these subjects a WSS estimate could provide an added value in a risk stratification study. In Section 5.3.2, velocity and pressure fields are reconstructed and visualized. Finally, in Section 5.3.4, it is proposed an estimate of wall shear stress for two different patients.

5.1. *In-vivo* data preprocessing

We consider four measurements on different patients with bicuspid (BAV) and tricuspid (TAV) aortic valves.

ID	Aortic valve	Age	M/F
T1026	TAV	33	M
B0001	BAV	31	F
B2000	BAV	28	M
B0003	BAV	26	M

Table 5.1: Patients characteristics.

Those data are fully deintentified and provided by Weill Cornell Medicine, (NY, USA). A thoracic 4D flow MRI scan of a subject with ascending thoracic aortic aneurysm is retrospectively retrieved. A respiratory compensated technique is adopted with the following settings: spatial resolution (voxel size) = 1.14 mm $\times$ 1.14 mm $\times$ 0.9 mm, field of view = 360 mm, flip angle = 15°, VENC = 200 cm/s in all 3 directions, time between consecutive frames = 30 ms, for a total of 20 frames per cardiac cycle [37].
DICOM images were processed using open-source code [38] to compute PCMRA image,

and segmentation is performed with `ITK-SNAP` [46]. PCMRA is derived from the magnitude images $M(\cdot)$ and velocity components as [16]:

$$\text{PCMRA} = \sqrt{\frac{1}{N}\sum_{t=1}^{N} M^2(t)\left(v_x^2(t) + v_y^2(t) + v_z^2(t)\right)}.$$

The use of both magnitude and velocities permits to easily identify the domain to be segmented, as can be seen in Figure 5.1.

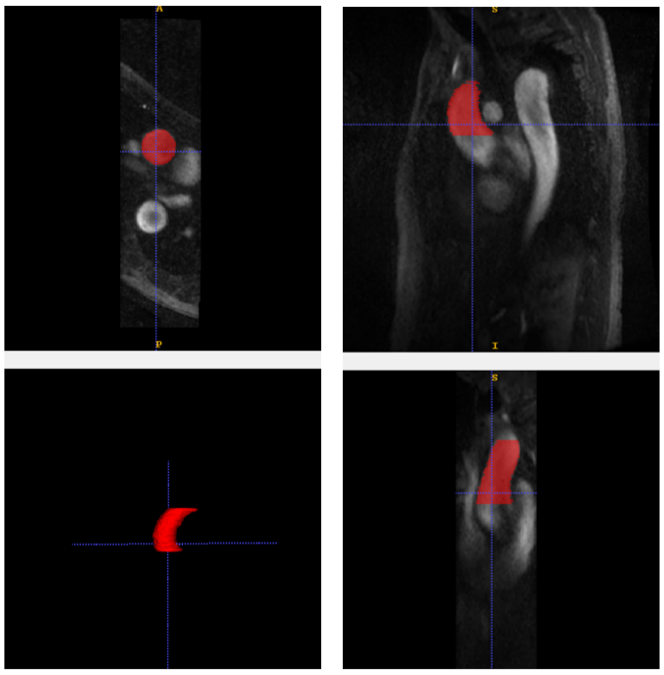

Figure 5.1: `ITK-SNAP` environment with a slice on each plane and the current segmented object in the lower left section.

The resulting surface is then tagged to define wall, inlet and outlet regions and a very refined mesh ($\approx$ 300k nodes) is created with `vmtk`. The mesh structure will be used to sample points for the model training, as done with synthetic data. DICOM images are visualized in `Paraview` and then sampled in the previously defined mesh, yielding to *vtk* files describing velocities at different timesteps interspersed by 40 ms. The resulting files are visually analyzed to identify the systole peak: this step is needed since in data acquisition there could be phases of completely useless noisy recordings, due to technical problems and instrument calibration. Finally, four files describing the blood activity form a dataset with data spread in a 1.6 s time interval.

5.2. Model description

The same model architecture and training strategy employed in Chapter 4 is used for real test cases characterized by:

- model architecture: 8 hidden layers with 32 neurons each, and 4 neurons in both input and output layers;
- sin(·) as activation function;
- weights are initialized as described in Section 1.2 and there is the first layer adjustment;
- first minimization loop: 500 epochs with Adam optimizer (learning rate: 0.01);
- second minimization loop: 3000 epochs with L-BFGS optimizer;
- 20k collocations points where the physics loss terms are evaluated and 20k data-fidelity points both spread in the entire time interval are considered;
- loss weights are reported in Table 5.2:

	w_{fit}	w_{mass}	w_{mom}	$w_{neumann}$	w_{wall}	w_{reg}
weights	1	0.1	0.1	$1 \cdot 10^{-4}$	0.1	$1 \cdot 10^{-5}$

Table 5.2: Loss weights for real 4D flow test cases.

5.3. Results

In this Section we present the loss evolution for the best model and its velocity and pressure reconstructions. To support the physical regularization process we verify the imposition of the mass conservation principle. Finally, in Section 5.3.4 we propose an estimate for WSS.

5.3.1. Loss evolution

The model training is very challenging when working with real data yielding to a difficult minimization process of loss terms represented in Figure 5.2. The data-fidelity term maintains the main role, while, physical terms are always kept low and there is not a clear descent in their evolution. The same issues, encountered before, and related to the balance between loss terms, are seen in those real cases: due to the high level of corruption of data, the network does not find an optimal trade-off between data fit and physics constraints.

Relying more on data yields to unphysical outputs, while relying more on physics leads to flatter velocity and pressure fields.

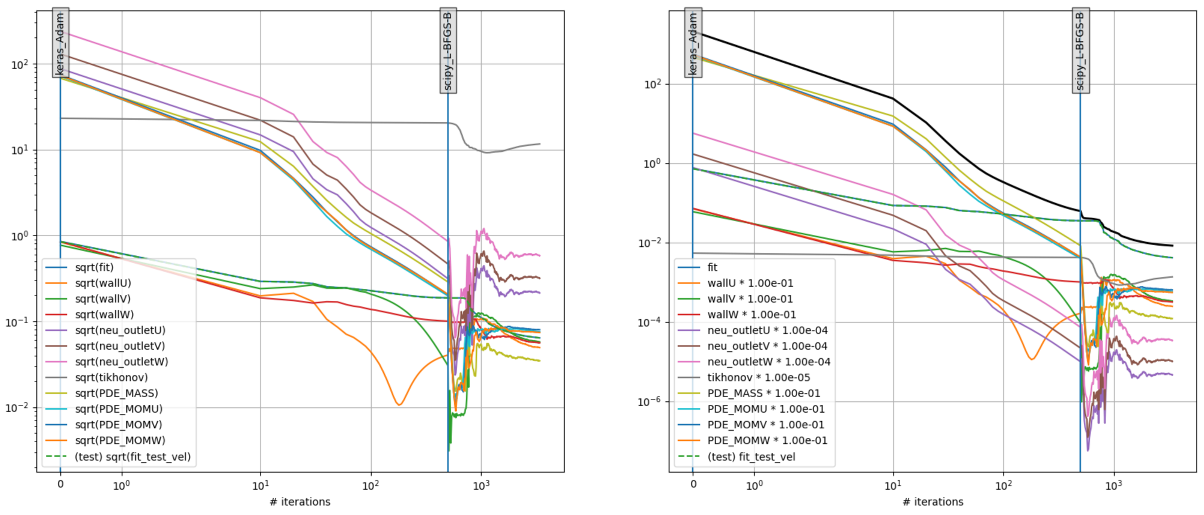

Figure 5.2: Loss evolution for the model trained for patient B2000. On the left are visualized each loss term with their real magnitude before taking the square and loss weights are not considered. On the right, each term is multiplied by its corresponding weight and here it can be seen their influence on the global loss value. PDE_MASS refers to the residual of the incompressibility constraint and PDE_MOM refers to the momentum NS equation.

5.3.2. Velocity and pressure reconstruction

Despite training difficulties, PINNs succeed in obtaining super-resolved and regular velocity and pressure fields, yielding to a significant improvement with respect to initial 4D flow MRI recordings. Nevertheless, there remains some secondary limitations to note: the predicted flow does not show many details especially in wall boundaries where the no-slip condition is not completely imposed, and high velocity gradients do not always appear at the wall. This must be taken into consideration when computing WSS in Section 5.3.4.

The results for patients B0031 and B2000, in three different slices at the same timestep, are shown in the following figures. In Figures 5.3, 5.4, 5.5 we represent the velocity for patient B003: looking at the Z-component, there is clear vertical flux that reaches a maximum velocity of 0.40 $\frac{\text{m}}{\text{s}}$. In Figure 5.6, it is represented the pressure field with a pressure drop between inlet and outlet of about 100 Pa. The velocity reconstruction for patient B2000 is reported in Figures 5.7, 5.8, 5.9: lower values are obtained with respect to the other patient and the no-slip condition is not correctly imposed. Finally, in Figure 5.10, we can appreciate an inner region with lower pressure values as in the synthetic test cases.

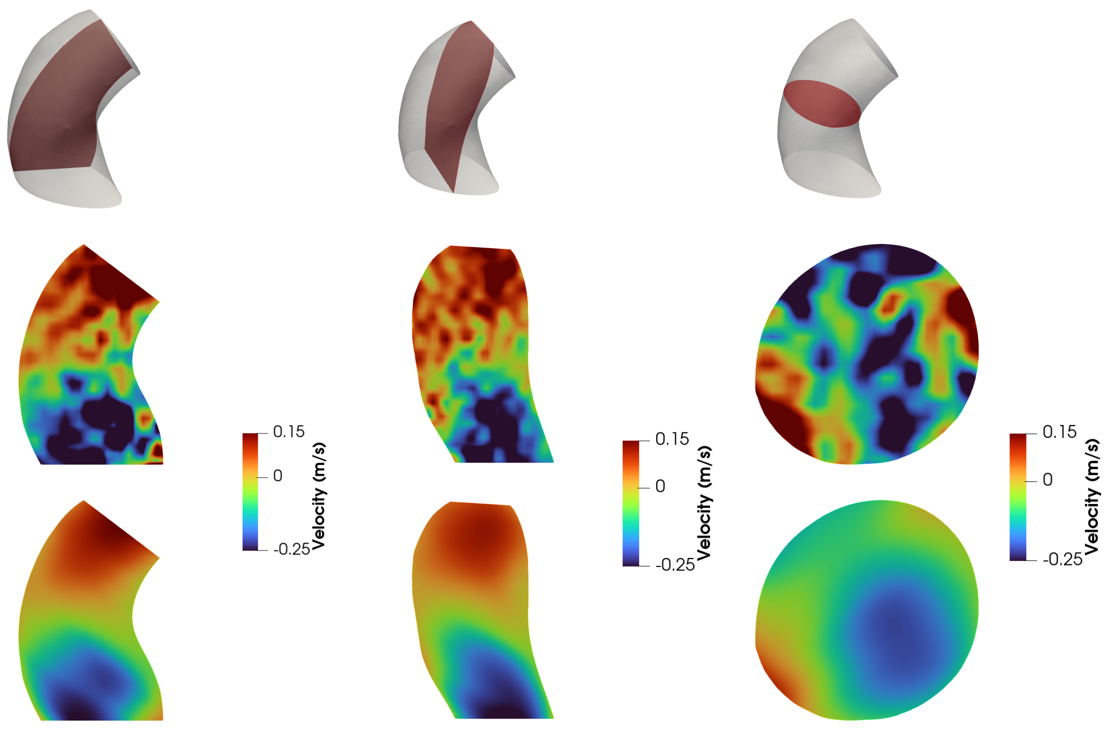

Figure 5.3: B003. X velocity component colormaps in different slices at the same timestep. On the top there are 4D flow MRI data and, on the bottom, the model prediction.

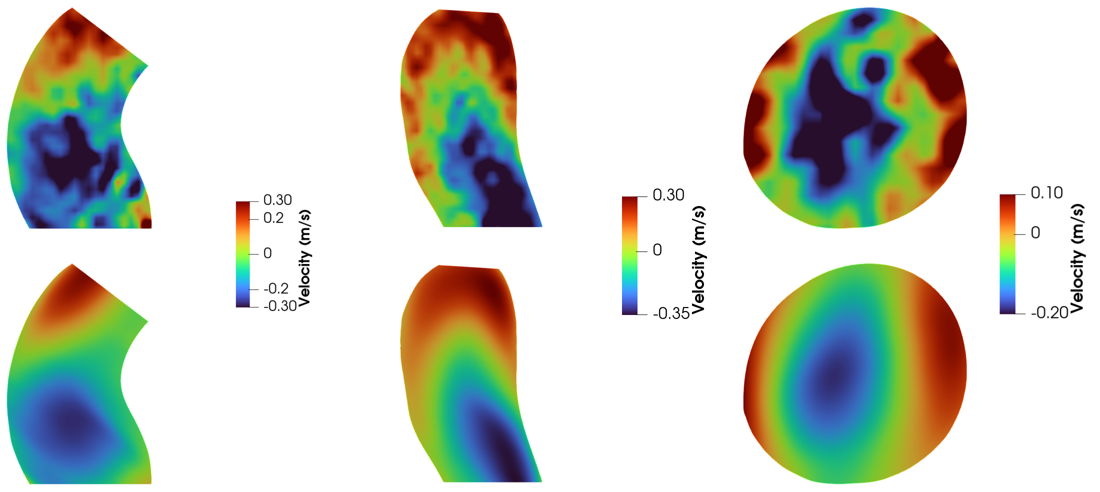

Figure 5.4: B003. Y velocity component colormaps in different slices at the same timestep. On the top there are 4D flow MRI data and, on the bottom, the model prediction.

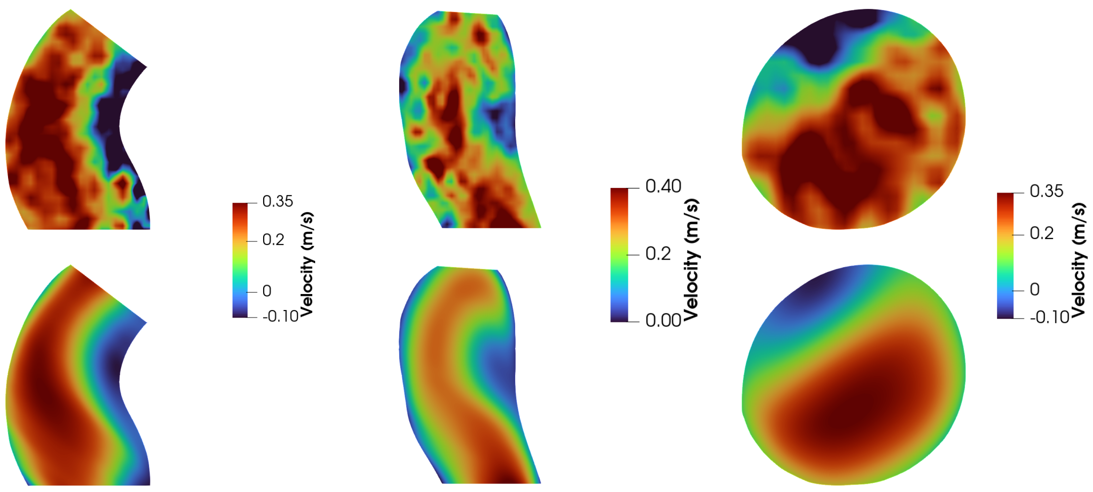

Figure 5.5: B003. Z velocity component colormaps in different slices at the same timestep. On the top there are 4D flow MRI data and, on the bottom, the model prediction.

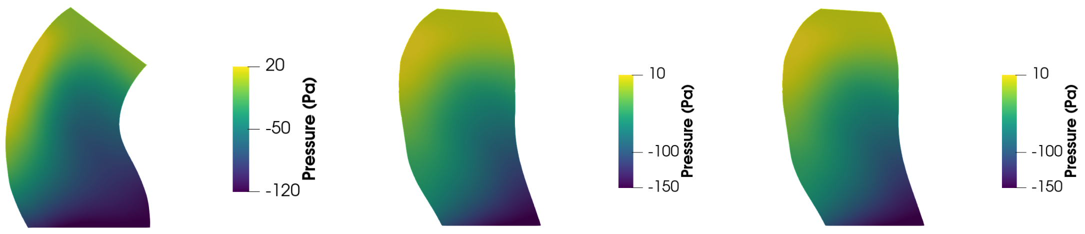

Figure 5.6: B003. Pressure reconstruction in different slices at the same timestep.

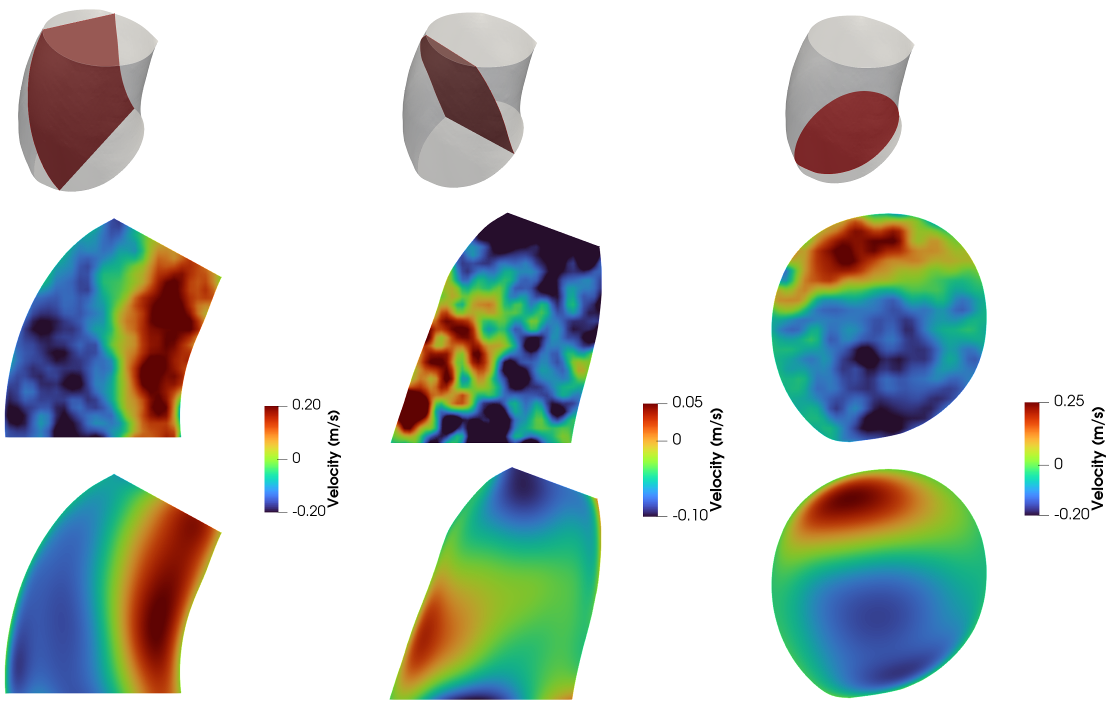

Figure 5.7: B2000. X velocity component colormaps in different slices at the same timestep. On the top there are 4D flow MRI data and, on the bottom, the model prediction.

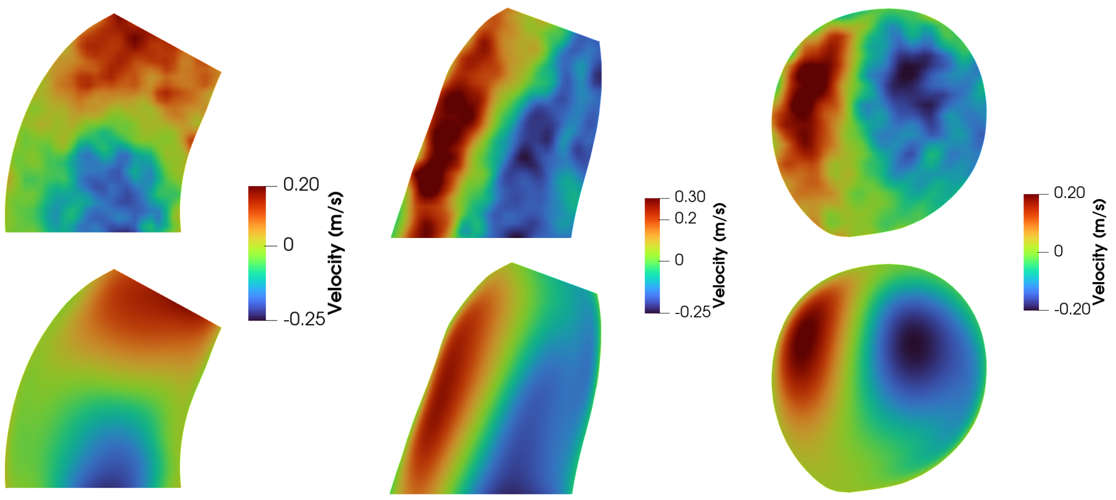

Figure 5.8: B2000. Y velocity component colormaps in different slices at the same timestep. On the top there are 4D flow MRI data and, on the bottom, the model prediction.

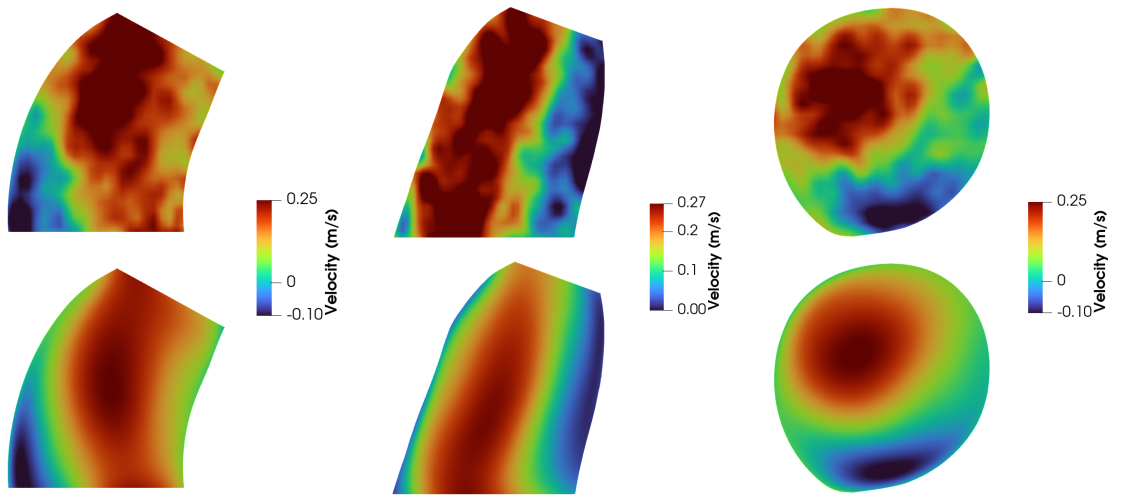

Figure 5.9: B2000. Z velocity component colormaps in different slices at the same timestep. On the top there are 4D flow MRI data and, on the bottom, the model prediction.

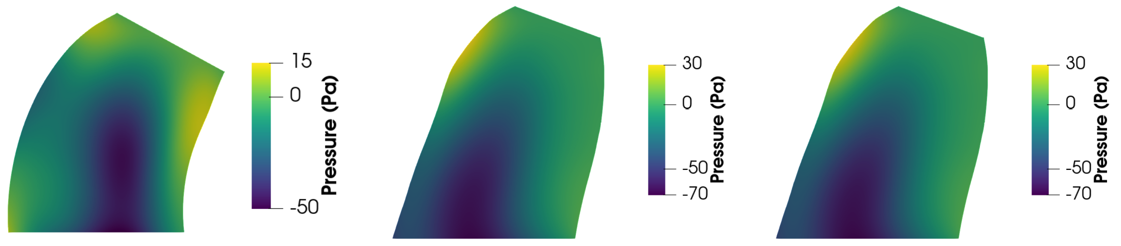

Figure 5.10: B2000. Pressure reconstruction in different slices at the same timestep.

5.3.3. Mass conservation

As a further indicator of the regularization process conducted by the model, it is possible to verify the mass conservation principle in an arbitrary volume $\mathcal{K}$ inside the domain. The outward velocity flux over $\partial\mathcal{K}$ is computed with `Paraview` in two cubes, starting from both the predicted velocity field and the 4D flow measurements. PINNs succeed in enforcing the mass conservation principle, indeed the computed value is approximately two orders of magnitude lower. The fact that it always decreases with the model action highlights the correct enforcing of the governing PDEs. The fluxes are reported in Table 5.3.

PATIENT	Edge length [m]	Cube center [m]	PINN result	4D FLOW result
B2000	0.005	(0.040, 0.150, 0.260)	$2.79 \cdot 10^{-8}$	$2.84 \cdot 10^{-7}$
B2000	0.01	(0.040, 0.140, 0.260)	$2.02 \cdot 10^{-7}$	$3.25 \cdot 10^{-7}$
B003	0.005	(0.030, 0.140, 0.225)	$1.35 \cdot 10^{-7}$	$3.58 \cdot 10^{-6}$
B003	0.01	(0.022, 0.13, 0.240)	$6.59 \cdot 10^{-7}$	$1.09 \cdot 10^{-5}$

Table 5.3: The outward velocity flux over $\partial\mathcal{K}$ for different cubes $\mathcal{K}$.

5.3.4. Wall Shear Stress estimation

The reconstruction of realistic velocity profiles in the near wall region is extremely difficult starting from *in-vivo* data. The profiles are flatter and no help is provided from the measurements. Indeed, the main sources of inaccuracies at the boundaries come from intrinsic limitations in acquisition and segmentation procedures.

The same methods described in the previous section are employed to recover a realistic profile at the wall. They all provide a regular WSS field with a significant improvement with respect to the estimate from raw 4D flow data. Obviously there cannot be a reference field to compare the prediction with, but it is reasonable to conclude that the results are underestimated. All methods identify the same regions with a higher stress and only PAR ZERO recover larger values, as expected from its nature.

As in Section 4.5.2, the parabolic fitting method equipped with $d_n = 7 \cdot 10^{-4}$ m and with the parabola that starts from zero (PAR7 ZERO) is the preferred method. This is because it is more robust with respect to potential inaccuracies present in the near wall region and it could recover high velocity gradients.

The results for a single timestep can be seen in Figures 5.11 and 5.12, where a different colorbar is required to represent the higher WSS values computed through PAR2 ZERO and PAR7 ZERO.

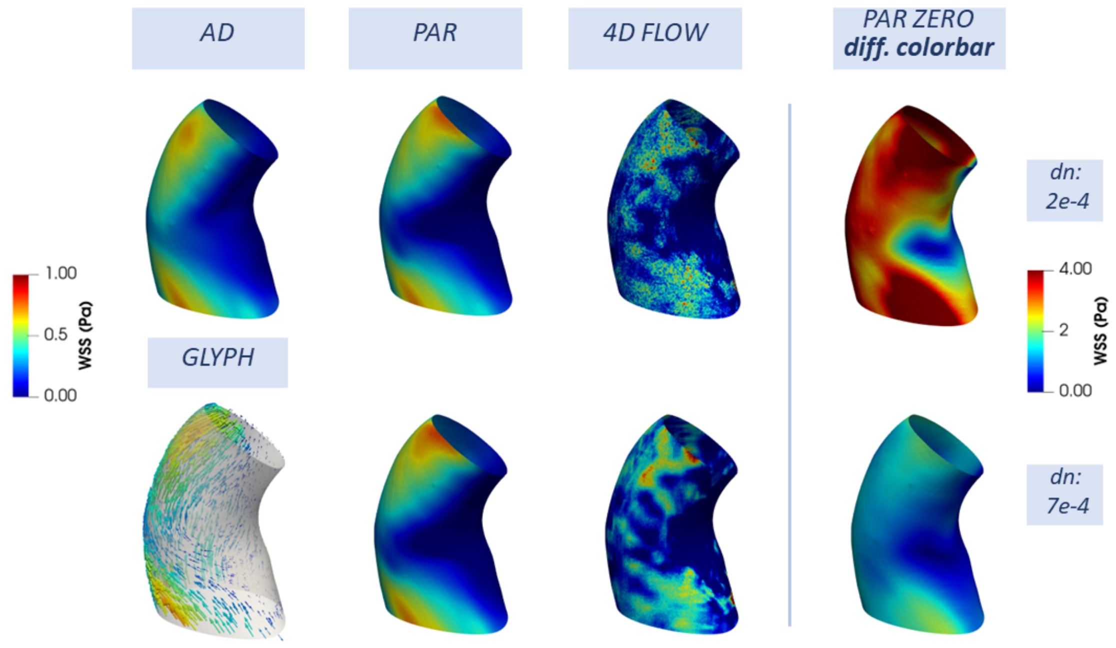

Figure 5.11: WSS computed with different methods for patient B003, and there is a vectorial representation only for the WSS field computed through AD. In the last column, there is a different colorbar since the values are very high.

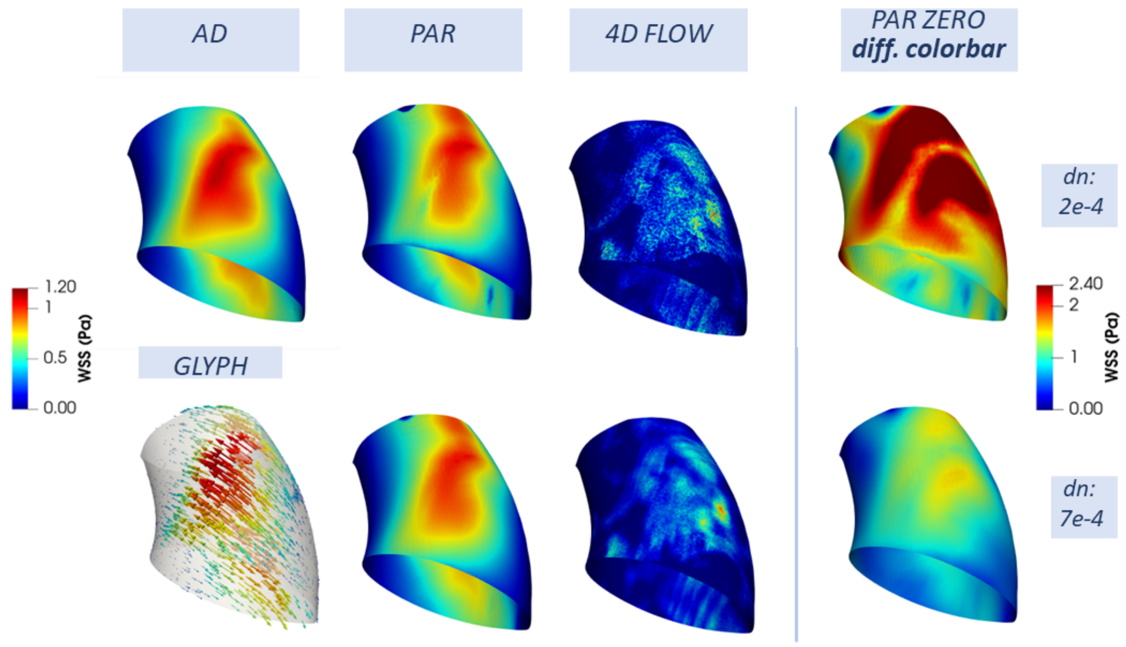

Figure 5.12: WSS computed with different methods for patient B2000, and there is a vectorial representation only for the WSS field computed through AD. In the last column, there is a different colorbar since the values are very high.

6 | Conclusion

In this work physics informed neural networks are employed to improve the quality of 4D flow MRI measurements. Within a deep learning framework, it is possible to combine available data with a modelling knowledge that describes the blood behaviour through physical laws: the measurements provide patient-specific information to guide the training of the MLP, whose output is regularized by neural network interpolation properties together with the physics loss terms. After the training process, the neural network is capable of approximating pressure and velocity fields of a patient in a restricted space-time domain. Within this domain, arbitrary points could be evaluated yielding to a super resolute output. This regular representation constraint to physical laws, enable to estimate crucial biomarkers, such as wall shear stress to comprehend cardiovascular diseases.
To find an optimal training strategy, we analyze a test case starting from a ground-truth solution, as it is possible to evaluate the performances. Thus, synthetic data are generated from a CFD reference simulation to emulate real measurements. In this controlled environment, different strategies to balance loss terms are tested together with the imposition of boundary condition. Relying more on fitting data permits to avoid a flatter output, moreover, enforcing the no-slip condition with an additional term in the loss function improves the performances with respect to the imposition of null velocity with the lifting procedure.
Side tasks, such as wall normal computation, are also faced exploiting deep learning techniques: additional neural networks are used to represent the distance from the wall boundary together with automatic differentiation.
A reference ground-truth for wall shear stress is obtained looking only in a small region of the domain in Chapter 4. This permits to reconstruct accurate velocity gradients in the near wall region, and optimal normal derivatives are easily computed through AD. Starting from this reference, WSS is estimated in the whole domain exploiting different methods that consider various velocity profiles near the boundary. Numerical results show that PINNs suffer from the spectral bias [32], and they cannot correctly recover meaningful velocity profile at the wall. AD cannot be used in challenging cases since the flat and unrealistic profile predicted by it cannot be considered and, thus, there is the need

to force a parabolic profile to recover the typical high velocity gradients.

In Chapter 5, this methodology is then applied on *in-vivo* data recorded on BAV and TAV patients. In particular, BAV patients are characterized by evident alterations of WSS distribution and peak values in the ascending aorta, thus, an accurate WSS estimate could provide an added value in a risk stratification study about aortopathies. WSS is obtained starting from the velocity field and great improvements in velocity and pressure representation, with respect to unprocessed 4D flow data, are obtained through the action of PINNs. In this case, high frequencies with specific flow details are lost, but the patient blood flow behaviour is correctly reconstructed together with the pressure field that was not recorded in the initial 4D flow measurements. Moreover, we highlight the effectiveness of the physical regularization by verifying the mass conservation principle in a small cube embedded in the domain. Finally, a WSS estimate is proposed: the model accuracy is limited since the model fails in predicting precise velocity profiles at the wall boundary but, the imposition of a parabolic profile that starts from a null velocity together with the use of the net to evaluate the innermost points, succeed in proposing a realistic WSS field.

This work highlights the potential of PINNs to denoise and to make super-resolute 4D flow measurements, but the computational resources required make this methodology not yet feasible in real applications, as they require results in a very concise time. The net training is heavy, and, even if the model structure and the weights used in the loss are designed to remain unchanged, the training must be patient-specific. Moreover, it could be improved the capability of the model to capture high frequencies and flow details that are typically lost. The physical knowledge that is enforced in the training favors smoother output fields, as even a constant null prediction is accepted by the physics loss terms.
Further developments could go in the direction of exploiting transfer learning techniques to reuse the information discovered when analyzing a specific patient. Following [28], a possible improvement could be a latent representation of each patient's environment considering both their geometry and their specific flow-details. Then a global neural network could be trained on several patients and it could learn the relations between the outputs and the latent space representations. To conclude, future improvements in the performances and in the training strategies will permit to effectively exploit these techniques in clinical application.

Bibliography

[1] A. Arzani, J.-X. Wang, and R. M. D'Souza. Uncovering near-wall blood flow from sparse data with physics-informed neural networks. *Physics of Fluids*, 33(7), 2021.

[2] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*, 18: 1–43, 2018.

[3] S. Berrone, C. Canuto, M. Pintore, and N. Sukumar. Enforcing dirichlet boundary conditions in physics-informed neural networks and variational physics-informed neural networks. *arXiv preprint arXiv:2210.14795*, 2022.

[4] A. Biswas and V. Shapiro. Approximate distance fields with non-vanishing gradients. *Graphical Models*, 66(3):133–159, 2004.

[5] E. Bollache, A. J. Barker, R. S. Dolan, J. C. Carr, P. van Ooij, R. Ahmadian, A. Powell, J. D. Collins, J. Geiger, and M. Markl. k-t accelerated aortic 4d flow mri in under two minutes: feasibility and impact of resolution, k-space sampling patterns, and respiratory navigator gating on hemodynamic measurements. *Magnetic resonance in medicine*, 79(1):195–207, 2018.

[6] A. Cárdenas-Blanco, C. Tejos, P. Irarrazaval, and I. Cameron. Noise in magnitude magnetic resonance images. *Concepts in Magnetic Resonance Part A: An Educational Journal*, 32(6):409–416, 2008.

[7] Y. Chen, L. Lu, G. E. Karniadakis, and L. Dal Negro. Physics-informed neural networks for inverse problems in nano-optics and metamaterials. *Optics express*, 28 (8):11618–11633, 2020.

[8] P.-H. Chiu, J. C. Wong, C. Ooi, M. H. Dao, and Y.-S. Ong. Can-pinn: A fast physics-informed neural network based on coupled-automatic–numerical differentiation method. *Computer Methods in Applied Mechanics and Engineering*, 395:114909, 2022.

[9] N. E. Cotter. The stone-weierstrass theorem and its application to neural networks. *IEEE transactions on neural networks*, 1(4):290–295, 1990.

[10] F. J. Detmer, D. Lückehe, F. Mut, M. Slawski, S. Hirsch, P. Bijlenga, G. von Voigt, and J. R. Cebral. Comparison of statistical learning approaches for cerebral aneurysm rupture assessment. *International journal of computer assisted radiology and surgery*, 15:141–150, 2020.

[11] Z. Fang. A high-efficient hybrid physics-informed neural networks based on convolutional neural network. *IEEE Transactions on Neural Networks and Learning Systems*, 33(10):5514–5526, 2021.

[12] M. F. Fathi, I. Perez-Raya, A. Baghaie, P. Berg, G. Janiga, A. Arzani, and R. M. D'Souza. Super-resolution and denoising of 4d-flow mri using physics-informed deep neural nets. *Computer Methods and Programs in Biomedicine*, 197:105729, 2020.

[13] E. Ferdian, D. J. Dubowitz, C. A. Mauger, A. Wang, and A. A. Young. Wssnet: aortic wall shear stress estimation using deep learning on 4d flow mri. *Frontiers in Cardiovascular Medicine*, 8:1969, 2022.

[14] H. Gao, L. Sun, and J.-X. Wang. Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. *Journal of Computational Physics*, 428:110079, 2021.

[15] N. Geneva and N. Zabaras. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *Journal of Computational Physics*, 403: 109056, 2020.

[16] A. Hennemuth, O. Friman, C. Schumann, J. Bock, J. Drexl, M. Huellebrand, M. Markl, and H.-O. Peitgen. Fast interactive exploration of 4d mri flow data. In *Medical Imaging 2011: Visualization, Image-Guided Procedures, and Modeling*, volume 7964, pages 110–120. SPIE, 2011.

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

[18] K. Hornik, M. Stinchcombe, and H. White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks*, 3(5):551–560, 1990.

[19] G. Kissas, Y. Yang, E. Hwuang, W. R. Witschey, J. A. Detre, and P. Perdikaris. Machine learning in cardiovascular flows modeling: Predicting arterial blood pres-

sure from non-invasive 4d flow MRI data using physics-informed neural networks. *Computer Methods in Applied Mechanics and Engineering*, 358:112623, jan 2020.

[20] I. E. Lagaris, A. Likas, and D. I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE transactions on neural networks*, 9 (5):987–1000, 1998.

[21] X. Li. Simultaneous approximations of multivariate functions and their derivatives by neural networks with one hidden layer. *Neurocomputing*, 12(4):327–343, 1996.

[22] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[23] A. Logg and G. N. Wells. Dolfin: Automated finite element computing. *ACM Transactions on Mathematical Software (TOMS)*, 37(2):1–28, 2010.

[24] L. Lu, X. Meng, Z. Mao, and G. E. Karniadakis. Deepxde: A deep learning library for solving differential equations. *SIAM review*, 63(1):208–228, 2021.

[25] M. Markl, A. Frydrychowicz, S. Kozerke, M. Hope, and O. Wieben. 4d flow mri. *Journal of Magnetic Resonance Imaging*, 36(5):1015–1036, 2012.

[26] L. McClenny and U. Braga-Neto. Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv preprint arXiv:2009.04544*, 2020.

[27] A. J. Meade Jr and A. A. Fernandez. The numerical solution of linear ordinary differential equations by feedforward neural networks. *Mathematical and Computer Modelling*, 19(12):1–25, 1994.

[28] J. J. Park, P. R. Florence, J. Straub, R. A. Newcombe, and S. Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. 2019.

[29] S. Petersson, P. Dyverfeldt, and T. Ebbers. Assessment of the accuracy of mri wall shear stress estimation using numerical simulations. *Journal of Magnetic Resonance Imaging*, 36(1):128–138, 2012.

[30] F. Piatti, F. Sturla, M. M. Bissell, S. Pirola, M. Lombardi, I. Nesteruk, A. Della Corte, A. C. Redaelli, and E. Votta. 4d flow analysis of bav-related fluid-dynamic alterations: evidences of wall shear stress alterations in absence of clinically-relevant aortic anatomical remodeling. *Frontiers in physiology*, 8:441, 2017.

[31] D. C. Psichogios and L. H. Ungar. A hybrid neural network-first principles approach to process modeling. *AIChE Journal*, 38(10):1499–1511, 1992.

[32] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

[33] 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.

[34] M. Raissi, A. Yazdani, and G. E. Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

[35] F. Regazzoni. nisaba, 2020. URL `https://sci-learning.gitlab.io/nisaba/`.

[36] A. Rich, L. C. Potter, N. Jin, Y. Liu, O. P. Simonetti, and R. Ahmad. A bayesian approach for 4d flow imaging of aortic valve in a single breath-hold. *Magnetic resonance in medicine*, 81(2):811–824, 2019.

[37] S. Saitta, M. Carioni, S. Mukherjee, C.-B. Schönlieb, and A. Redaelli. Implicit neural representations for unsupervised super-resolution and denoising of 4d flow mri. *arXiv preprint arXiv:2302.12835*, 2023.

[38] S. Saitta, L. Maga, C. Armour, E. Votta, D. P. O'Regan, M. Y. Salmasi, T. Athanasiou, J. W. Weinsaft, X. Y. Xu, S. Pirola, et al. Data-driven generation of 4d velocity profiles in the aneurysmal ascending aorta. *Computer Methods and Programs in Biomedicine*, 233:107468, 2023.

[39] H. Samady, P. Eshtehardi, M. C. McDaniel, J. Suo, S. S. Dhawan, C. Maynard, L. H. Timmins, A. A. Quyyumi, and D. P. Giddens. Coronary artery wall shear stress is associated with progression and transformation of atherosclerotic plaque and arterial remodeling in patients with coronary artery disease. *Circulation*, 124(7):779–788, 2011.

[40] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

[41] Z. Stankovic, B. D. Allen, J. Garcia, K. B. Jarvis, and M. Markl. 4d flow imaging with mri. *Cardiovascular Diagnosis and Therapy*, 4(2), 2014.

[42] N. Sukumar and A. Srivastava. Exact imposition of boundary conditions with distance functions in physics-informed deep neural networks. *Computer Methods in Applied Mechanics and Engineering*, 389:114333, 2022.

[43] L. Sun, H. Gao, S. Pan, and J.-X. Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020.

[44] J. Taylor, W. Wang, B. Bala, and T. Bednarz. Optimizing the optimizer for data driven deep neural networks and physics informed neural networks. *arXiv preprint arXiv:2205.07430*, 2022.

[45] V. Vishnevskiy, J. Walheim, and S. Kozerke. Deep variational network for rapid 4d flow mri reconstruction. *Nature Machine Intelligence*, 2(4):228–235, 2020.

[46] P. A. Yushkevich, Y. Gao, and G. Gerig. Itk-snap: An interactive tool for semi-automatic segmentation of multi-modality biomedical images. In *2016 38th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*, pages 3342–3345. IEEE, 2016.

[47] Y. Zhu, N. Zabaras, P.-S. Koutsourelakis, and P. Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.

List of Figures

List of Tables

Acknowledgements

Grazie a mamma e papà per avermi dato la possibilità e per avermi insegnato ad esplorare il mondo. Insieme a tutta la nostra famiglia mi avete insegnato ad aver fiducia e a voler bene alle persone. Provandoci tutti i giorni. Sono orgoglioso di avervi sempre vicino, tutti voi. Siete un esempio, vi voglio bene.
Ringrazio quei professori che in questi anni mi hanno messo alla prova, è servito sbattere continuamente la testa da solo o confrontandomi con gli altri. Avete trasmesso veramente la passione per quello che facciamo.
Ottignies-Louvain-la-Neuve, jamais je ne t'oublierai. Mi hai fatto camminare, mi hai fatto scoprire cosa mi piace fare e, al ritorno, mi hai regalato nuoviamici. Merci beauacouop.
Grazie ai miei amici e a quelle emozioni che ricerchiamo. Vedo sempre di più come stiamo diventando grandi, come abbiamo imparato a non avere paura di prendere decisioni forti, come prendiamo in considerazione anche la via più difficile. Mi avete sempre dato tanta fiducia e tanta felicità in questi anni durante l'università. Bisognava sognare tanto per aspettarselo.
Sometimes science is so difficult it makes me sad, other times volano gli angeli e sappiamo già come finisce.
Ciaos, ci pensa la vita.