

Universal Solution Manifold Networks (USM-Nets): non-intrusive mesh-free surrogate models for problems in variable domains

Francesco Regazzoni

MOX - Department of Mathematics,
Politecnico di Milano, Milan, Italy.
francesco.regazzoni@polimi.it

Stefano Pagani

MOX - Department of Mathematics,
Politecnico di Milano, Milan, Italy.
stefano.pagani@polimi.it

Alfio Quarteroni

MOX-Department of Mathematics, Politecnico di Milano, Milan, Italy.
Institute of Mathematics, EPFL, Lausanne, Switzerland (*Professor Emeritus*)
alfio.quarteroni@polimi.it

We introduce Universal Solution Manifold Network (USM-Net), a novel surrogate model, based on Artificial Neural Networks (ANNs), which applies to differential problems whose solution depends on physical and geometrical parameters. Our method employs a mesh-less architecture, thus overcoming the limitations associated with image segmentation and mesh generation required by traditional discretization methods. Indeed, we encode geometrical variability through scalar landmarks, such as coordinates of points of interest. In biomedical applications, these landmarks can be inexpensively processed from clinical images. Our approach is non-intrusive and modular, as we select a data-driven loss function. The latter can also be modified by considering additional constraints, thus leveraging available physical knowledge. Our approach can also accommodate a universal coordinate system, which supports the USM-Net in learning the correspondence between points belonging to different geometries, boosting prediction accuracy on unobserved geometries. Finally, we present two numerical test cases in computational fluid dynamics involving variable Reynolds numbers as well as computational domains of variable shape. The results show that our method allows for inexpensive but accurate approximations of velocity and pressure, avoiding computationally expensive image segmentation, mesh generation, or re-training for every new instance of physical parameters and shape of the domain.

1 Introduction

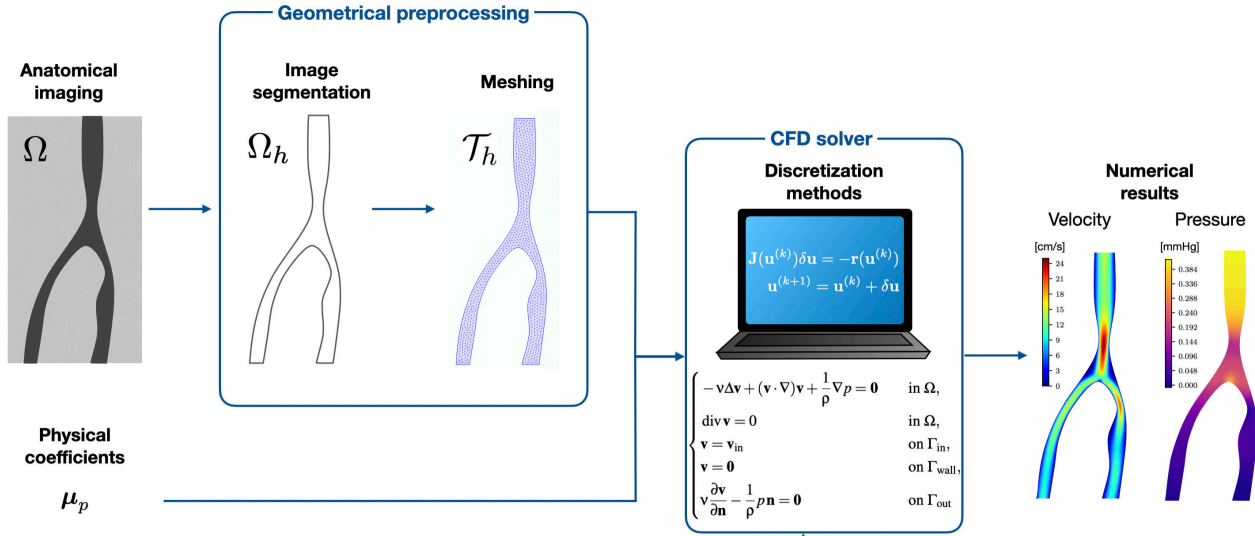
Models and methods in scientific computing and machine learning enable the extraction of relevant knowledge from available data [1]. Data can represent, e.g., physical coefficients, geometrical factors, boundary or initial conditions. This is the case of computational fluid dynamics, a remarkable instance, especially when addressing problems in aero-

dynamics, such as the design of vehicles, and in biomedical engineering, such as patient-specific analysis of blood flow dynamics [2, 3, 4, 5].

The standard approach to modeling and simulation (Fig. 1, top) requires the construction of a computational mesh, a partition of the given geometry in simple elements, e.g. tetrahedral or hexahedral cells. In biomedical applications, the construction of the computational mesh requires a preliminary step of segmentation, i.e. the extraction of the boundaries of the computational domain from medical images, such as those derived from magnetic resonance or computerized tomography. Then, discretization methods (like Finite Elements and Finite Difference methods [6]) assemble on the elements of the mesh a suitable approximation of the operators associated with the partial differential equations (PDEs). Unfortunately, changes in shape require the re-execution of the entire process, necessitating the reallocation of significant computational resources.

For this reason, some computational approaches, like isogeometric analysis (IGA) and shape models, are designed to avoid the regeneration of a new computational mesh when a change of geometry occurs. IGA achieves this thanks to the use of non-uniform rational B-splines (NURBS) that exactly match CAD geometries, usually adopted in an industrial context [7]. Similarly, geometrical shape models describe the possible variations of geometry through a limited number of parameters. Among the most diffused shape models we recall free-form deformation (FFD) [8, 9], radial basis function (RBF) [10], and statistical shape models [11], based e.g. on principal component analysis (PCA) [12]. Firstly conceived in computer graphics, FFD is a technique that surrounds the object with a lattice of control points. Their motion drives the deformation of all object points through a polynomial interpolation. RBFs define a parametrized map describing

Standard modeling and simulation approach



USM-Net approach

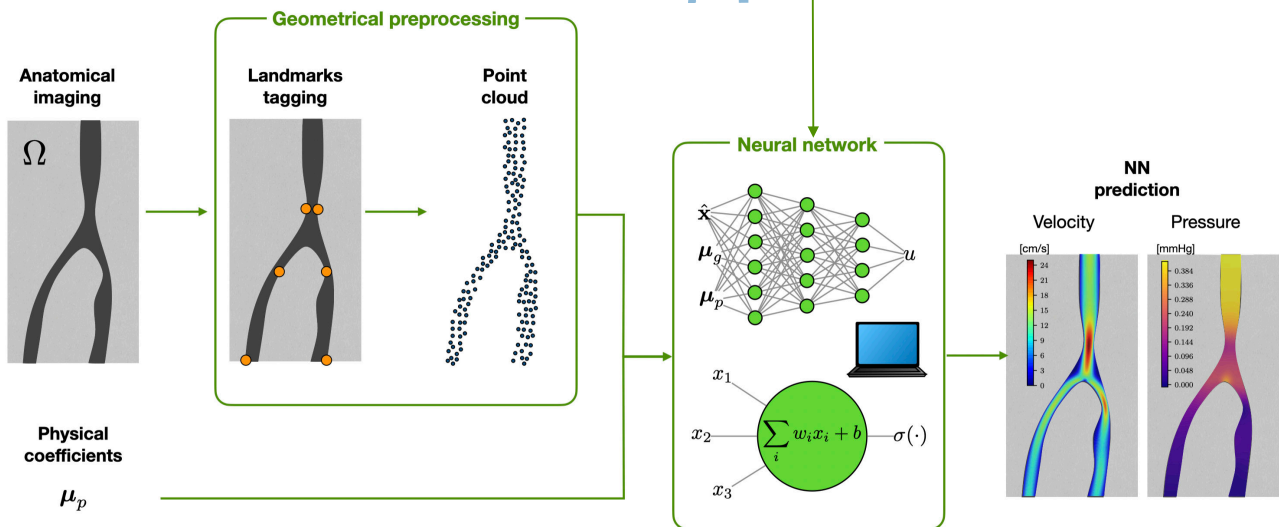


Fig. 1. Comparison between the standard modeling and simulation approach (top) and the USM-Nets approach (bottom) for a use case of clinical interest, namely the prediction of blood flow and pressure within a coronary bifurcation. The former approach requires a geometric preprocessing phase, which consists in segmenting the patient's clinical images to extract a computational domain Ω_h . A partition of the latter into a set of cells (in the figure, into triangles) constitutes the computational mesh \mathcal{T}_h . The numerical solution is then obtained by the discretization of the Navier-Stokes equations on the computational mesh \mathcal{T}_h , solved through suitable computer-based algorithms. Our proposed USM-Net approach lightens both the geometric preprocessing and the solution approximation steps. The first one consists solely of landmarks extraction and (mesh-less) cloud of query points generation. Finally, the solution is obtained by evaluating the neural network as many times as the number of query points. The USM-Net can be trained either from experimentally measured data or from synthetic data generated using the Navier-Stokes solver.

the deformations of the geometry from a small number of selected control points as well. Compared to the FFD approach, the control points position is not constrained on a lattice but user-selected from application to application. Finally, PCA-based models describe a collection of similar geometries through weighted modes describing the principal geometrical variations with respect to a mean shape. These approaches enable the construction of reduced-order models (ROMs) for problems with parameterized geometry, which provide a computationally efficient approximation of the solution for many different choices of the geometrical parameters. In this context, empirical interpolation methods build efficient approximations of differential operators avoiding computationally costly reassembly. Among the main applications of parametrized ROMs [13, 14, 15, 16, 17, 18], we mention those related to computational fluid dynamics using FFD [19, 20], RBF [21, 22, 23] and PCA [24].

Besides introducing a geometrical error, the critical aspect of shape models is that, when deforming a fixed mesh, the elements may encounter such deformations that the discretized problem becomes ill-conditioned. This might considerably limit the ability of the method to explore the geometric variability of the problem. To overcome the limitations of mesh-based approaches, mesh-free and particle methods construct a discretization of the geometry formed solely by a collection of points, relaxing the constraints given by the connectivity [25]. These methods bring numerous advantages in terms of managing geometrical accuracy (even with discontinuities), imposing large deformations, adaptively refining, and code parallelization. However, they generate full-order models (FOMs) with limited computational efficiency.

In this paper, we introduce a new surrogate modeling technique based on artificial neural networks (ANNs), that can learn the solution manifold of a given parametrized PDE. Notably, these surrogate models are not tailored on a given domain, but they account for the influence of geometry on the solution. We leverage the capabilities of approximating arbitrary complex functions with inexpensive output evaluation provided by ANNs, which are indeed *universal* approximators of several families of functions, including continuous functions [26] and Sobolev spaces [27, 28]. Based on these results, the surrogate models we propose are, in principle, able to approximate the solution manifold of a given differential problem with arbitrary accuracy and universally with respect to the variation of domain geometry and physical parameters. For this reason, we name them *Universal Solution Manifold Networks* (USM-Nets). We train these networks using subsamples of solutions snapshots obtained by varying the physical parameters and the domain geometry (either synthetically generated through a FOM or experimentally collected). Therefore, the accuracy of the predictions of a specific USM-Net would directly depend on the richness of the training set used. However, the architecture we propose has the potential to accurately approximate the solution universally with respect to domain and physical parameters, which is not the case for methods that are constrained to a predetermined parameterization of the solution manifold.

The design of a USM-Net avoids complex geometrical preprocessing (comprising segmentation and construction of the computational mesh; see Fig. 1). We encode geometrical variability in a few *geometrical landmarks*, a finite number of scalar indicators, inexpensively processed from the input image. In the most basic case, landmarks are the coordinates of specific reference points where we impose a correspondence between geometries. Landmarks might identify inlets, bifurcations, or other specific structures, and they provide, together with the physical parameters, the input to the ANN.

Our approach is modular, thanks to the possibility of arbitrarily defining the loss function. Besides penalizing the misfit with the available data, during the training of the network we can enforce assumptions of regularity (imposing, e.g., weights penalization), the initial or boundary conditions, or the fulfillment of an equation in a strong (differential) form. The latter would represent a generalization of the so-called physics-informed neural network [29, 30, 31], avoiding new executions of the training process for each geometry variation.

The outline of this paper is as follows. First, in Sec. 2 we present the notation used throughout this paper and the proposed methods. Then, in Sec. 3 we present two test cases and we describe how our proposed methods can be applied to them. In Sec. 4 we present the numerical results and in Sec. 6, finally, we draw our conclusions.

2 Methods

In this section, we first introduce the notation used throughout this paper. Then, we present our proposed USM-Net method.

2.1 Problem setting

We consider a space-dependent physical quantity $\mathbf{u}(\mathbf{x})$, defined on a domain $\Omega \subset \mathbb{R}^d$, where typically $d = 2, 3$. For the sake of generality, we denote $\mathbf{u}(\mathbf{x}) \in \mathbb{R}^k$ as a vector field even though in certain cases it could be a scalar field (that is $k = 1$). For example, $\mathbf{u}(\mathbf{x})$ may correspond to the blood velocity field in a vessel, the displacement field of soft tissue, or the pressure field of air around a body. Very often, the quantity $\mathbf{u}(\mathbf{x})$ depends on a set of physical parameters $\mu_p \in \mathcal{P}$, where $\mathcal{P} \subset \mathbb{R}^{n_p}$ is the parameter space. The parameters μ_p characterize the physical processes that determine \mathbf{u} . For example, the velocity of a fluid depends on its viscosity, while the displacement of biological tissue depends on its stiffness moduli and the applied force. Furthermore, the field $\mathbf{u}(\mathbf{x})$ depends on the domain Ω itself. In many practical applications, we are interested in the family of fields $\mathbf{u}(\mathbf{x})$ obtained by varying the domain Ω in a given set, denoted by \mathcal{G} :

$$\mathcal{G} \subset \{\Omega \subset \mathbb{R}^d, \text{ open and bounded}\}.$$

To stress the dependence of $\mathbf{u}(\mathbf{x})$ on both μ_p and Ω , we will henceforth write $\mathbf{u}(\mathbf{x}; \mu_p, \Omega)$.

Often, the physical processes that determine $\mathbf{u}(\mathbf{x}; \mu_p, \Omega)$ can be described by a mathematical model assuming the form of a boundary-value problem, that is

$$\begin{cases} \mathcal{L}(\mathbf{y}, \mu_p) = \mathbf{0} & \text{for } \mathbf{x} \in \Omega, \\ \mathcal{B}(\mathbf{y}, \mu_p) = \mathbf{0} & \text{for } \mathbf{x} \in \partial\Omega, \\ \mathbf{u} = \mathcal{U}(\mathbf{y}, \mu_p) & \text{for } \mathbf{x} \in \Omega, \end{cases} \quad (1)$$

where \mathbf{y} in the state variable; \mathcal{L} and \mathcal{B} are the operators associated with the differential equation and with the boundary conditions, respectively; \mathcal{U} is the observation operator. Since the state \mathbf{y} is instrumental for obtaining \mathbf{u} , in what follows we will refer to \mathbf{u} and not to \mathbf{y} as the *solution* of the FOM. If the differential problem (1) is well-posed, then, given $\mu_p \in \mathcal{P}$ and $\Omega \in \mathcal{G}$ there exists a unique solution $\mathbf{u}(\cdot; \mu_p, \Omega): \Omega \rightarrow \mathbb{R}^k$. This solution can be numerically approximated through a FOM based e.g. on Finite Differences or Finite Elements [6].

The goal of this paper is to build an emulator that surrogates the solution map $(\mu_p, \Omega) \mapsto \mathbf{u}(\cdot; \mu_p, \Omega)$, that is a model that, for any given value of the geometrical parameters μ_p and any given geometry Ω , provides an approximation of the corresponding solution \mathbf{u} .

2.2 USM-Net

A USM-Net is an ANN-based model, trained on a data-driven basis, that surrogates the solution map $(\mu_p, \Omega) \mapsto \mathbf{u}(\cdot; \mu_p, \Omega)$, without using any FOM. In the most common scenario, a FOM that describes the physical process is available, and it is used to generate the data needed to train the USM-Net. Still, the training of a USM-Net is also possible for physical problems in which the FOM is unknown, provided that sufficient experimental measurements are available. We present two versions of USM-Net:

1. PC-USM-Net, when the solution is represented in terms of the *physical coordinates*, that is $\mathbf{x} \in \Omega$ (see Sec. 2.2.1);
2. UC-USM-Net, with the solution represented now by passing through of a system of *universal coordinates*, that will be defined in Sec. 2.2.2.

2.2.1 PC-USM-Net

The PC-USM-Net architecture is represented in Fig. 2 (top). It consists of an ANN, typically a fully connected ANN (FCNN), whose input is obtained by stacking three vectors:

1. the query point \mathbf{x} , that is the coordinate of the point where the solution $\mathbf{u}(\mathbf{x})$ is sought;
2. the physical parameters μ_p ;
3. a set of geometrical landmarks associated with the domain at hand, that is $\mu_g = P_g(\Omega)$, that typically represent the coordinates of key points of the domain (such as inlets, bifurcation points, etc.) or geometrical measures (such as diameters, thicknesses, etc.). In Sec. 2.3 we will elaborate on possible choices for the function P_g .

The output of the FCNN is an approximation of the solution \mathbf{u} at the query point \mathbf{x} . More precisely, denoting by \mathcal{N} the FCNN and by \mathbf{w} its trainable parameters (weights and biases), we have:

$$\mathbf{u}(\mathbf{x}; \mu_p, \Omega) \simeq \mathcal{N}(\mathbf{x}, \mu_p, P_g(\Omega); \mathbf{w}).$$

Hence, \mathcal{N} features $d + n_p + n_g$ input neurons and k output neurons.

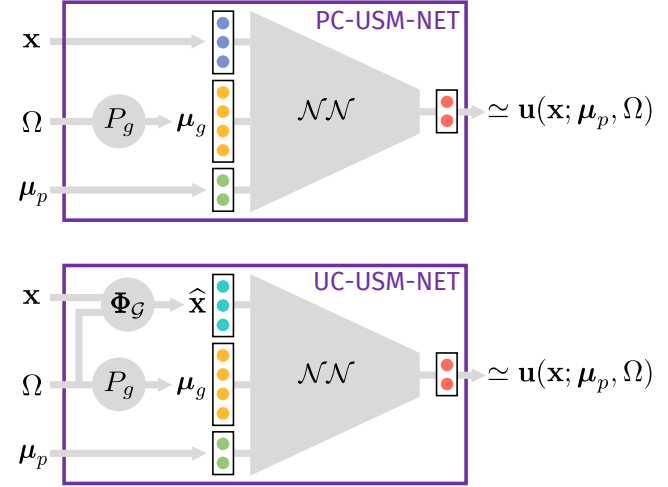


Fig. 2. Architecture of a PC-USM-Net (top) and of a UC-USM-Net (bottom).

2.2.2 UC-USM-Net

As anticipated, a PC-USM-Net has a universality character with respect to domains, i.e. a single network is used to represent solutions in different geometries. However, a given point with physical coordinate \mathbf{x} , given as input to the PC-USM-Net, can play a different *role* for different geometries. For example, a point \mathbf{x} that for one geometry Ω^1 belongs to the boundary of the domain, for another geometry Ω^2 could be internal to the domain, and for yet another Ω^3 could even be external. Therefore, we propose an evolution of PC-USM-Net aimed at capturing more effectively the correspondence between points among geometries.

To achieve this goal, we rely on a *universal coordinates (UC) system* for \mathcal{G} . A UC system is a map Φ_G that, to any domain Ω and to any point $\mathbf{x} \in \Omega$, associates a point $\hat{\mathbf{x}} \in \hat{\Omega}$ belonging to a *reference domain* $\hat{\Omega} \subset \mathbb{R}^d$. More precisely, the reference coordinate is obtained as $\hat{\mathbf{x}} = \Phi_G(\mathbf{x}, \Omega)$. We require that the application $\Phi_G(\cdot, \Omega): \Omega \rightarrow \hat{\Omega}$ be a continuous bijection for any $\Omega \in \mathcal{G}$. Hence, the UC system Φ_G defines a coordinate transformation that maps each domain $\Omega \in \mathcal{G}$ into the reference one $\hat{\Omega}$. In Sec. 3 we will present two concrete examples of UC systems.

Whenever a UC system is available, it can augment a PC-USM-Net. The enhanced version of a PC-USM-Net,

called UC-USM-Net, is obtained by giving as input to \mathcal{N} the reference coordinate $\widehat{\mathbf{x}} = \Phi_G(\mathbf{x}, \Omega)$ instead of the physical one $\mathbf{x} \in \Omega$. More precisely, the surrogate model is defined as

$$\mathbf{u}(\mathbf{x}; \mu_p, \Omega) \simeq \mathcal{N}(\mathcal{N}(\Phi_G(\mathbf{x}, \Omega), \mu_p, P_g(\Omega); \mathbf{w})). \quad (2)$$

The resulting architecture is represented in Fig. 2 (bottom).

We remark that UC-USM-Net is a generalization of PC-USM-Net, as the latter can be obtained from the former by setting Φ_G equal to the identity function, that is by setting $\widehat{\mathbf{x}} = \mathbf{x}$. For this reason, from now on we will consider without loss of generality the surrogate model of Eq. (2).

As we will show in the results section, UC-USM-Nets allow to improve the generalization accuracy of PC-USM-Nets, that is the accuracy of predictions for physical parameters and geometries not included in the training set, by providing geometrical prior knowledge during training. Moreover, we will show that, besides helping the ANN to link together points of different geometries, a UC system might provide details of the geometry that are not captured by the landmarks.

We remark that, in practical applications, both the injectivity and the surjectivity requirements of the map $\Phi_G(\cdot, \Omega): \Omega \rightarrow \widehat{\Omega}$ can be relaxed.

2.3 Geometrical landmarks

In order to build a surrogate model that learns an approximation of the solution map, we introduce a low-dimensional description of the geometry. In particular, we construct an operator $P_g: \mathcal{G} \mapsto \mathbb{R}^{n_g}$ that, to any given computational domain $\Omega \in \mathcal{G}$, associates a finite number (say n_g) of geometrical landmarks $\mu_g = P_g(\Omega) \in \mathbb{R}^{n_g}$, which are suitable to provide a compact description of Ω . Depending on the structure of \mathcal{G} , different strategies can be followed to define the operator P_g .

1. In case an explicit parametrization of the elements of the space \mathcal{G} is available, we define P_g in such a way that the landmarks $\mu_g = P_g(\Omega)$ coincide with the geometrical parameters themselves. An example is provided in Test Case 1.
2. If such a parameterization is not available (as it is in many cases of practical interest), a straightforward choice is to take the coordinates of key points in the domain as landmarks. An example is provided in Test Case 2.
3. Other more sophisticated techniques can be used to obtain a low-dimensional description of the computational domains. For example, the geometrical landmarks can be defined as the first, more relevant, coefficients associated with a POD analysis of a finite subset of \mathcal{G} (shape model). Entering into the details of this or other techniques is beyond the scope of this paper. The method proposed in this paper is indeed general, as it is built on top of the different techniques that can be used to define the map.

In general, our method does not require the operator P_g to be invertible. As a matter of fact, P_g is invertible only when an explicit parametrization of the space \mathcal{G} is available. In fact, we allow for the case when two different geometries $\Omega^1 \neq \Omega^2$, both belonging to \mathcal{G} , are associated to identical landmarks (i.e. $P_g(\Omega^1) = P_g(\Omega^2)$). Since the geometrical landmarks characterize the variability of the geometry, a good design of P_g requires that the condition $P_g(\Omega^1) = P_g(\Omega^2)$ implies that Ω^1 and Ω^2 are *minimally* different.

2.4 Training a USM-Net

To train a USM-Net (that is, either a PC-USM-Net or a UC-USM-Net), we require the output of problem (1) for several pairs $(\mu_p, \Omega) \in \mathcal{P} \times \mathcal{G}$. These solutions, called *snapshots*, are typically obtained through the FOM, a high-fidelity numerical solver of Eq. (1), based e.g. on Finite Elements of Finite Differences [6]. Yet, as our method is non-intrusive and does not require any knowledge of equation (1), snapshots can also be derived from a black-box solver, or even from experimental measurements.

We consider a collection of N_{sn} snapshots, associated with $\mu_p^i \in \mathcal{P}$ and $\Omega^i \in \mathcal{G}$, for $i = 1, \dots, N_{sn}$. For any snapshot, then, we consider a number of observations in a set of points belonging to Ω^i . In case of high variability of the geometries in \mathcal{G} , the resolution of the FOM typically requires the generation of different meshes, without a one-to-one correspondence of the nodes. Therefore, to guarantee generality, we consider the case where each snapshot has a potentially different number of observation points. Specifically, we denote by $\{\mathbf{x}_j^i, j = 1, \dots, N_{pt}^i\}$ the set of observation points associated with the i -th snapshot. In conclusion, the training dataset consists of the following set

$$\{\mu_p^i, \Omega^i, \{\mathbf{u}(\mathbf{x}_j^i; \mu_p^i, \Omega^i)\}_{j=1}^{N_{pt}^i}\}_{i=1}^{N_{sn}}.$$

Training the USM-Net requires solving the following minimization problem

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \mathcal{J}(\mathbf{w}).$$

The loss function \mathcal{J} is given by the misfit between snapshot data and predictions, plus (optionally) a regularization term \mathcal{R} :

$$\mathcal{J}(\mathbf{w}) = \frac{1}{N_{sn}} \sum_{i=1}^{N_{sn}} \frac{1}{N_{pt}^i} \sum_{j=1}^{N_{pt}^i} d(\mathbf{u}_j^i, \tilde{\mathbf{u}}_j^i) + \mathcal{R}(\mathbf{w}), \quad (3)$$

having defined

$$\begin{aligned} \mathbf{u}_j^i &= \mathbf{u}(\mathbf{x}_j^i; \mu_p^i, \Omega^i), \\ \tilde{\mathbf{u}}_j^i &= \mathcal{N}(\mathcal{N}(\Phi_G(\mathbf{x}_j^i, \Omega^i), \mu_p^i, P_g(\Omega^i); \mathbf{w})) \end{aligned}$$

and where $d(\cdot, \cdot)$ is a suitable discrepancy metric (typically, $d(\mathbf{u}, \tilde{\mathbf{u}}) = \|\mathbf{u} - \tilde{\mathbf{u}}\|^2$). Standard techniques, such as Tikhonov

or LASSO regularization, can be used for the regularization term \mathcal{R} . Additionally, \mathcal{R} can be augmented by suitable terms informing the USM-Net of physical knowledge available on the solution (see also Sec. 2.5). An example in this sense will be shown in Sec. 4.

2.5 Grey-box USM-Net

So far, we have presented USM-Nets as fully non-intrusive (black-box) surrogate modeling techniques. Still, physical knowledge can be optionally embedded into their construction. Indeed, the training process can be augmented by informing the network either of physical constraints (such as conservation principles, symmetry properties, or the positivity of the solution) or of the differential equations and boundary conditions that characterize the problem. We distinguish between *weak imposition* and *strong imposition* of the physical knowledge.

Weak imposition. Prior knowledge on the solution map can be enforced through the regularization term \mathcal{R} of the loss function of (3). \mathcal{R} can include the norm of the residual of the FOM equations and boundary conditions evaluated in a collection of collocation points, as done in the training of Physics Informed Neural Networks [29]. Similarly, other physical constraints can be rephrased in terms of minimization of a regularization term \mathcal{R} . Thanks to Automatic Differentiation, the inclusion of differential operators in the term \mathcal{R} does not introduce severe implementation efforts, even if it will slow down the training process. Therefore, the user should wisely balance the advantages and disadvantages of introducing such a term. An example of the weak imposition of the boundary conditions is presented in Test Case 1 (Sec. 3.1).

Strong imposition. Alternatively, we can enforce physical constraints by defining an architecture $\mathcal{N}\mathcal{N}$ that satisfies them by construction. We now give a brief list of examples:

1. Non-negativity of the solution can be enforced by introducing after the FCNN a further layer that applies an operator with nonnegative output, such as $(\cdot)^2$ or $|\cdot|$. In other terms, we perform a composition between the FCNN and the nonnegative operator.
2. Symmetry w.r.t. a given input coordinate can be enforced, e.g., by introducing an input layer that preprocesses the corresponding input through an even function, such as $|\cdot|$. As in the previous point, this corresponds to performing a composition between the even function and the FCNN.
3. Dirichlet boundary conditions on a portion of the boundary (i.e. $\mathbf{u}(\mathbf{x}) = \mathbf{u}_D$ on $\Gamma_D \subset \partial\Omega$) can be strongly enforced by introducing a multiplicative layer after the FCNN that multiplies the solution by a mask function $\Phi_{BC}(\mathbf{x}, \Omega)$, such that $\Phi_{BC} = 0$ on Γ_D and $\Phi_{BC} \neq 0$ elsewhere, and sums the datum \mathbf{u}_D .
4. Solenoidality of the solution ($\nabla \cdot \mathbf{u} = 0$), a common requirement in fluid dynamics, can be enforced by interpreting the FCNN output as the flow field potential and

introducing an output layer that returns its curl. An example of this technique is described in Sec. 3.1.

2.6 Notes about implementation

From the implementation point of view, a few cautions are needed to make the training of USM-Nets computationally light. First of all, despite the application of the P_g map in Fig. 2 is indicated as being an integral part of the USM-Net, the landmarks $\mu_g^i = P_g(\Omega^i)$ can be pre-calculated before the training. Similarly, in the case a UC system is employed, the coordinate transformation $\widehat{\mathbf{x}}_j^i = \Phi_{\mathcal{G}}(\mathbf{x}_j^i, \Omega^i)$ for any i and j can be performed offline, at a stage prior to training. In this manner, we set up an augmented dataset consisting of:

$$\begin{array}{ccc} (\widehat{\mathbf{x}}_1^1, & \mu_g^1, & \mu_p^1), & \mathbf{u}_1^1 \\ (\widehat{\mathbf{x}}_2^1, & \mu_g^1, & \mu_p^1), & \mathbf{u}_2^1 \\ \vdots & \vdots & \vdots & \vdots \\ (\widehat{\mathbf{x}}_{N_{pt}}^1, & \mu_g^1, & \mu_p^1), & \mathbf{u}_{N_{pt}}^1 \\ (\widehat{\mathbf{x}}_1^2, & \mu_g^2, & \mu_p^2), & \mathbf{u}_1^2 \\ \vdots & \vdots & \vdots & \vdots \end{array}$$

and $\mathcal{N}\mathcal{N}$ is trained to fit the map from the first three columns to the last one.

Once the $\mathcal{N}\mathcal{N}$ is trained, it can be used to approximate the solution for unseen parameters and/or geometries. This is the *online* stage, which consists of the following steps:

1. Receive Ω and μ_p ,
2. Compute $\mu_g = P_g(\Omega)$.
3. For any \mathbf{x}_j for which the solution is needed, compute $\widehat{\mathbf{x}}_j = \Phi_{\mathcal{G}}(\mathbf{x}_j, \Omega)$.
4. Evaluate $\mathbf{u}(\mathbf{x}_j; \mu_p, \Omega) \simeq \mathcal{N}\mathcal{N}(\widehat{\mathbf{x}}_j, \mu_p, \mu_g; \mathbf{w})$. Typically, this operation can be vectorized to further increase the velocity of execution.

We recall that both P_g and (if used) $\Phi_{\mathcal{G}}$ are defined case-by-case, depending on the application.

3 Test cases

In this section, we present two test cases and provide details on the implementation choices we followed to apply the methods presented in Sec. 2.

3.1 Test Case 1: lid-driven cavity

Test Case 1 is based on the well-known stationary lid-driven cavity problem (see, e.g., [32]), for which we consider an extension with variable geometry. The challenge of this test case is to capture the different vortex topologies formed for different Reynolds numbers and different aspect ratios of the geometry.

We consider a rectangular domain $\Omega_H = (0, 1) \times (0, H)$, with $H > 0$, and the following PDE (Navier-Stokes equations), where $\Gamma_H^D = \{(x, y)^T \in \Omega_H \text{ s.t. } y = H\}$ denotes the

top edge of the domain and \mathbb{Re} the Reynolds number:

$$\begin{cases} -\frac{1}{\mathbb{Re}}\Delta\mathbf{v} + (\mathbf{v}\cdot\nabla)\mathbf{v} + \nabla p = \mathbf{0} & \text{in } \Omega_H, \\ \nabla\cdot\mathbf{v} = 0 & \text{in } \Omega_H, \\ \mathbf{v} = (1,0)^T & \text{on } \Gamma_H^D, \\ \mathbf{v} = \mathbf{0} & \text{on } \partial\Omega_H \setminus \Gamma_H^D. \end{cases} \quad (4)$$

The unknowns on the problem are the fluid velocity \mathbf{v} and pressure p . The goal of Test Case 1 is to build an emulator that approximates the fluid velocity \mathbf{v} , given the geometry Ω_H and the Reynolds number. More precisely, we consider geometries with height H in the interval $[0.5, 2]$:

$$\mathcal{G} = \{\Omega_H = (0, 1) \times (0, H), \text{ for } 0.5 \leq H \leq 2\}.$$

The physical parameter consists of $\mu_p = \mathbb{Re}$ and ranges in the interval $\mathcal{P} = [10^2, 10^4]$.

3.1.1 Training data generation

To generate training data, we consider a Taylor-Hood Finite Element approximation of problem (4). We employ structured triangular computational meshes with a uniform space resolution of $h = 10^{-2}$. We remark that, as a consequence, Finite Element approximations associated with different domains of \mathcal{G} might feature different numbers of degrees of freedom. To tackle Newton convergence issues for large \mathbb{Re} , we equip the solver with an adaptive continuation ramp with respect to the Dirichlet datum.

To explore the set $\mathcal{G} \times \mathcal{P}$, we employ a Monte Carlo approach by independently sampling from a uniform distribution for H and a log-uniform distribution for \mathbb{Re} . After each FOM resolution, we export the velocity $\mathbf{v}(\mathbf{x})$ at a set of points randomly selected within the domain Ω_H .

3.1.2 Geometrical landmarks

Test Case 1 has an explicit parametrization of the domains in the set \mathcal{G} , the height H being the parameter. Henceforth, we define $\mu_g = H$ as the unique geometrical landmark, by setting $P_g(\Omega_H) := H$.

3.1.3 UC system

A straightforward (and also effective) UC system for Test Case 1 consists in mapping each domain $\Omega_H \in \mathcal{G}$ into the unit square $\hat{\Omega} := (0, 1)^2$, through the transformation:

$$\hat{x} = x, \quad \hat{y} = y/H.$$

More precisely,

$$\Phi_{\mathcal{G}}\left(\begin{pmatrix} x \\ y \end{pmatrix}, \Omega_H\right) := \begin{pmatrix} x \\ y/H \end{pmatrix}.$$

3.1.4 USM-Net architecture

We consider two different ANN architectures to build USM-Nets for Test Case 1 (see Fig. 3).

Velocity-field architecture The first architecture for $\mathcal{N}\mathcal{N}$ relies on a FCNN mapping \mathbf{x} (or $\hat{\mathbf{x}}$), μ_p and μ_g into an approximation of $\mathbf{v}(\mathbf{x}; \mathbb{Re}, \Omega_H)$. To ease the FCNN training, we normalize both input and output data by mapping them in the interval $[-1, 1]$, and we preprocess the Reynolds number through a log transformation. In conclusion, the FCNN features 4 input neurons and 2 output neurons.

Potential-field architecture As an alternative, we build a FCNN with a single output neuron, interpreted as the fluid flow potential $\psi(\mathbf{x}; \mathbb{Re}, \Omega_H)$, and we subsequently compute the approximation of the velocity field as:

$$\mathbf{v}(\mathbf{x}; \mathbb{Re}, \Omega_H) = \begin{pmatrix} +\frac{\partial}{\partial y}\psi(\mathbf{x}; \mathbb{Re}, \Omega_H) \\ -\frac{\partial}{\partial x}\psi(\mathbf{x}; \mathbb{Re}, \Omega_H) \end{pmatrix} \quad (5)$$

These operations are performed through Automatic Differentiation (AD) of the FCNN output. We remark that we do not need a FOM-based potential ψ for training data. The training is done directly with respect to the velocity data. The operations of (5) represent indeed the last layer of the architecture $\mathcal{N}\mathcal{N}$. Input and outputs are normalized as for the velocity-field architecture.

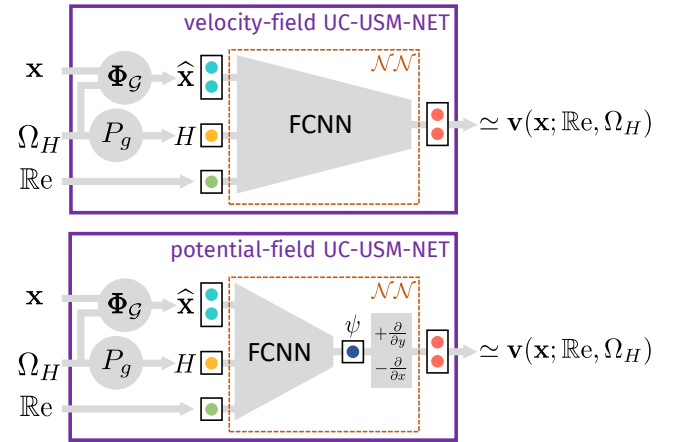


Fig. 3. Test Case 1: comparison of velocity-field and potential-field architectures.

3.1.5 Loss function

Since the goal of Test Case 1 is to reconstruct the velocity field with a focus on the vortex structure of the solution, we employ a discrepancy metric d that emphasizes the role of flow direction at each point in the domain, including those

with low flow intensity. Specifically, we define

$$d(\mathbf{u}, \tilde{\mathbf{u}}) = \|\mathbf{u} - \tilde{\mathbf{u}}\|^2 + \left\| \frac{\mathbf{u}}{\varepsilon + \|\mathbf{u}\|} - \frac{\tilde{\mathbf{u}}}{\varepsilon + \|\tilde{\mathbf{u}}\|} \right\|^2$$

where $\varepsilon = 10^{-4}$ is a small constant to avoid singularities. The second term drives the USM-Net to accurately match the direction of the velocity. Without this term, indeed, the flow direction would not be captured well in the regions with low flow magnitude, due to the low contribution in the first term of the loss.

Moreover, we augment the loss function with the following physics-based regularization term, aimed at enforcing the satisfaction of the Dirichlet boundary conditions:

$$\mathcal{R}(\mathbf{w}) = \frac{1}{N_{BC} N_{pt}^{BC}} \sum_{i=1}^{N_{BC}} \sum_{j=1}^{N_{pt}^{BC}} \|\mathbf{u}_{BC,j}^i - \mathbf{v}_{BC}(\mathbf{x}_{BC,j}^i)\|^2,$$

$$\mathbf{u}_{BC,j}^i = \mathcal{N}(\mathcal{N}(\mathbf{x}_{BC,j}^i, \mu_p^{BC,i}, \mu_g^{BC,i}; \mathbf{w}))$$

where $\mu_p^{BC,i} \in [10^2, 10^4]$ and $\mu_g^{BC,i} \in [0.5, 2]$ are sample points, $\mathbf{x}_{BC,j}^i$ is a set of points belonging to the boundary, and \mathbf{v}_{BC} is the Dirichlet datum (see (4)).

3.2 Test Case 2: coronary bifurcation

As a second test case, we consider the problem of predicting blood flow and pressure field within a coronary bifurcation in the presence of stenosis.

More precisely, we consider a computational domain Ω , corresponding to the 2D representation of a section of a coronary artery with a bifurcation. We denote by Γ_{in} the portion of the boundary corresponding to the inlet, by Γ_{out} the two outlets and by $\Gamma_{wall} = \Gamma_{top} \cup \Gamma_{bottom} \cup \Gamma_{front}$ the vessel wall. In this test case, we will consider many different computational domains, each representing a coronary bifurcation in a different virtual patient. An example domain is represented in Fig. 4.

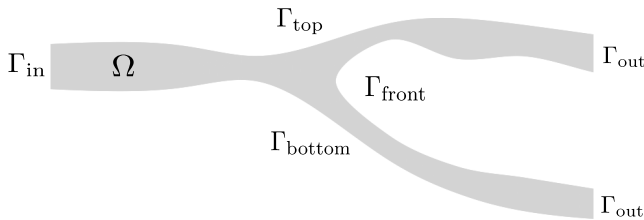


Fig. 4. Test Case 2: example of computational domain and corresponding boundaries.

We consider the following stationary Navier-Stokes model, describing the steady-state fluid flow in the coronary

bifurcation:

$$\begin{cases} -\nu \Delta \mathbf{v} + (\mathbf{v} \cdot \nabla) \mathbf{v} + \frac{1}{\rho} \nabla p = \mathbf{0} & \text{in } \Omega, \\ \text{div } \mathbf{v} = 0 & \text{in } \Omega, \\ \mathbf{v} = \mathbf{v}_{in} & \text{on } \Gamma_{in}, \\ \mathbf{v} = \mathbf{0} & \text{on } \Gamma_{wall}, \\ \mathbf{v} \frac{\partial \mathbf{v}}{\partial \mathbf{n}} - \frac{1}{\rho} p \mathbf{n} = \mathbf{0} & \text{on } \Gamma_{out}, \end{cases} \quad (6)$$

where $\nu = 4.72 \text{ mL}^2 \text{ s}^{-1}$ is the kinematic viscosity of blood and $\rho = 1060 \text{ kg m}^{-3}$ its density. At the inlet, we consider a parabolic profile with a maximum velocity equal to 14 cm s^{-1} . In Fig. 6, we show the numerical solution of (6) in the example computational domain of Fig. 4.

3.2.1 Geometrical variability and landmarks

The aim of Test Case 2 is to build a reduced model predicting the pressure and velocity fields in an arbitrary domain representing a coronary bifurcation. We synthetically generate a large number of different computational domains corresponding to many virtual patients. To do this, we use splines obtained by randomly varying their parameters in suitable intervals, defined to reflect the realistic variability observed among patients [33]. Notice that the geometries thus obtained may present stenoses, of a more or less acute degree, located upstream of the bifurcation or in the two branches downstream of it. A subsample of the geometries obtained following this procedure is displayed in Fig. 7.

Due to the lack of an explicit parameterization of these geometries (a common issue when dealing with domains from real patients), we need to define geometrical landmarks to characterize each geometry. To this end, we use an operatively light procedure that can also be easily adopted in a clinical context. Specifically, we define landmarks as the coordinates y of the vessel wall corresponding to a set of predefined coordinates x . These coordinates contain information regarding the lumen thickness at various levels and the possible presence of stenosis. Note that, in clinical practice, these landmarks can be easily derived directly from medical imaging, without the need to construct a computational mesh. In this test case, we will consider two sets of landmarks containing, respectively, 26 and 6 coordinates (see Fig. 5). Clearly, the first set is much more informative than the second one. The aim is to test the robustness of the proposed methods in the case where the landmarks provide a poor description of the geometry, and are not able to exhaustively capture its variability.

3.2.2 UC system

Differently than in Test Case 1, where, thanks to the simplicity of the domains of the \mathcal{G} space, it was possible to explicitly define a UC system, in Test Case 2 the construction of a UC system is not an immediate task. We propose to rely on two Laplacian-based fields, which define the inlet-outlet and

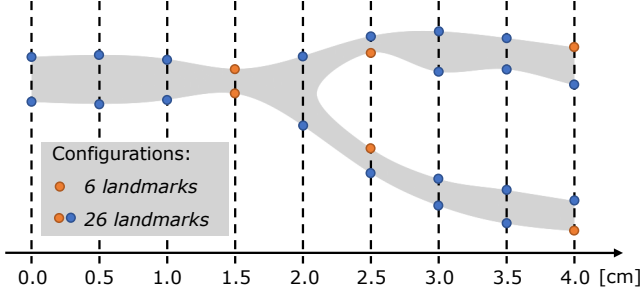


Fig. 5. Test Case 2: geometrical landmarks μ_g .

top-bottom directions, respectively. More precisely, given a geometry $\Omega \in \mathcal{G}$, we define the fields $\psi_{LR}: \Omega \rightarrow \hat{\Omega}$ and $\psi_{TD}: \Omega \rightarrow \hat{\Omega}$ as the solutions of the following differential problems:

$$\begin{cases} -\Delta\psi_{LR} = 0 & \text{in } \Omega, \\ \psi_{LR} = 0 & \text{on } \Gamma_{in}, \\ \psi_{LR} = 1 & \text{on } \Gamma_{out} \cup \Gamma_{front}, \\ \psi_{LR} = \frac{x - x_{min}}{x_{max} - x_{min}} & \text{on } \Gamma_{top} \cup \Gamma_{bottom}, \end{cases} \quad (7)$$

$$\begin{cases} -\Delta\psi_{TD} = 0 & \text{in } \Omega, \\ \psi_{TD} = +\alpha + (1 - \alpha) \frac{x - x_{min}}{x_{max} - x_{min}} & \text{on } \Gamma_{top}, \\ \psi_{TD} = -\alpha - (1 - \alpha) \frac{x - x_{min}}{x_{max} - x_{min}} & \text{on } \Gamma_{bottom}, \\ \frac{\partial\psi_{TD}}{\partial\mathbf{n}} = 0 & \text{on } \Gamma_{in} \cup \Gamma_{out} \cup \Gamma_{front}. \end{cases} \quad (8)$$

In Fig. 8 we show the fields ψ_{LR} and ψ_{TD} obtained for the domain of Fig. 4. The field ψ_{LR} bridges the inlet region (i.e. Γ_{in} , where $\psi_{LR} = 0$) with the frontal region of the domain (i.e. $\Gamma_{out} \cup \Gamma_{front}$, where $\psi_{LR} = 1$). Conversely, ψ_{TD} defines the proximity of each lumen point to the upper wall relative to the lower wall. Setting a constant $\alpha < 1$ allows better differentiation of the ψ_{TD} field within each branch downstream of the bifurcation. Specifically, we set $\alpha = 0.1$.

The UC system is thus defined as:

$$\hat{\mathbf{x}} = \Phi_{\mathcal{G}}(\mathbf{x}, \Omega) := \begin{pmatrix} \psi_{LR}(\mathbf{x}; \Omega) \\ \psi_{TD}(\mathbf{x}; \Omega) \end{pmatrix}.$$

In Fig. 9 we show the reference domain $\hat{\Omega}$ and the mutual correspondences between the boundary of the physical and reference domains.

3.2.3 Training data generation

To generate training data, we employ the Finite Element solver described for Test Case 1 (see Sec. 3.1). For space discretization, we consider triangular computational meshes

with a space resolution of nearly $h = 0.2$ mm. The UC coordinates are obtained by solving the differential problems (7) and (8) with $P1$ Finite Elements on the same computational mesh.

3.2.4 USM-Net architecture

In Test Case 2, we employ a FCNN connecting \mathbf{x} or $\hat{\mathbf{x}}$ (depending on whether a PC-USM-Net or a UC-USM-Net is used), μ_p and μ_g into an approximation of $\mathbf{u}(\mathbf{x}; \Omega_H)$ where the solution $\mathbf{u} = (\mathbf{v}, p)$ is given by the pair velocity-pressure. Similarly to Test Case 2, we add a normalization layer before the input and after the output layers of the FCNN, to constrain each input and each output in the interval $[-1, 1]$.

3.2.5 Loss function

We employ a purely black-box loss function, as defined in (3), with a quadratic discrepancy metric

$$d(\mathbf{u}, \tilde{\mathbf{u}}) = \|\mathbf{u} - \tilde{\mathbf{u}}\|^2$$

and without any regularization terms.

4 Results

In this section, we present the results obtained by applying the methods presented in Sec. 2 to the two test cases of Sec. 3. These results have been obtained using TensorFlow [34] and the optimizers of SciPy [35].

4.1 Test Case 1: lid-driven cavity

We construct a training set of $N_{sn} = 400$ numerical simulations obtained by randomly sampling the height H and the physical parameter $\mu_p = \mathbb{R}$. Some examples of streamlines resulting from numerical solutions by varying the parameters are reported in Fig. 10. For each geometry, we subsample the solution in $N_{pr} = 360$ random points.

By varying the dimensions of this dataset, we train several USM-Nets, formed by FCNNs made of 3 inner layers, respectively consisting of 30, 20, and 10 neurons. We consider four configurations:

1. velocity-field PC-USM-Net: receiving as inputs the two spatial coordinates, the physical and the geometrical parameters, and producing as outputs the two velocity components;
2. velocity-field UC-USM-Net: receiving as input the two universal coordinates, the physical and the geometrical parameters, and producing as outputs the two velocity components;
3. potential-field PC-USM-Net: receiving as inputs the two spatial coordinates, the physical and the geometrical parameters, and producing as output the two velocity components computed from the fluid flow potential;
4. potential-field UC-USM-Net: receiving as inputs the two universal coordinates, the physical and the geometrical parameters, and producing as output the two velocity components computed from the fluid flow potential.

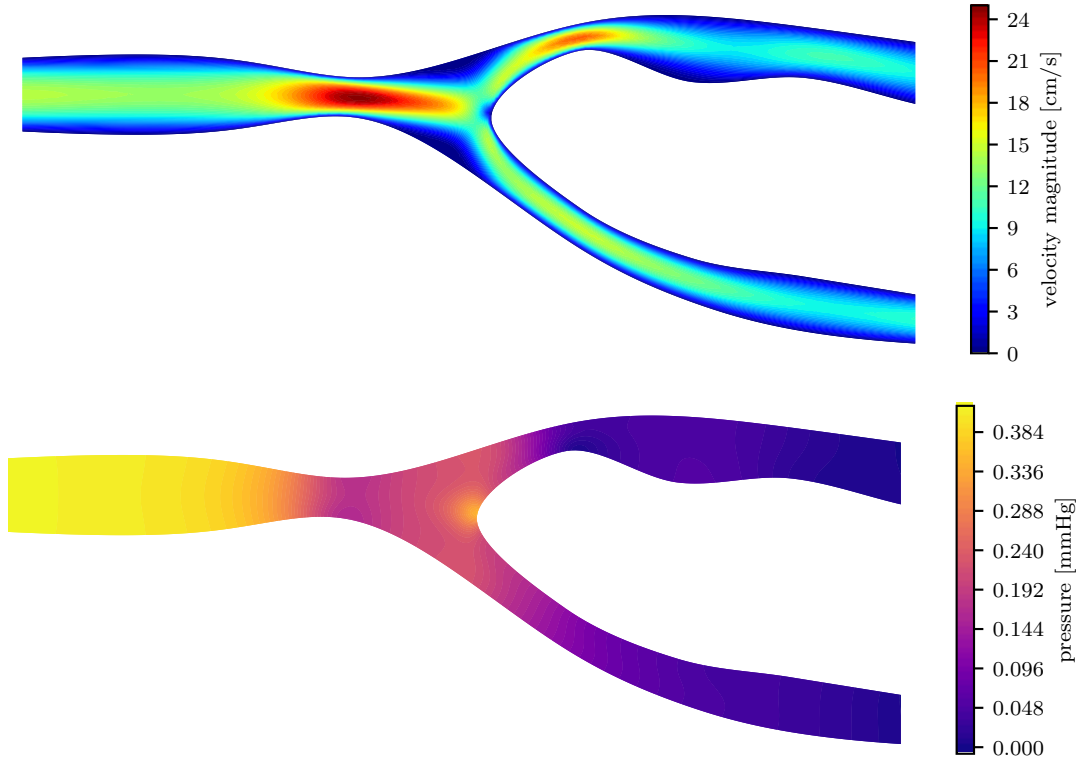


Fig. 6. Test Case 2: numerical solution of (6) on the computational domain of Fig. 4. Top: velocity field; bottom: pressure field.

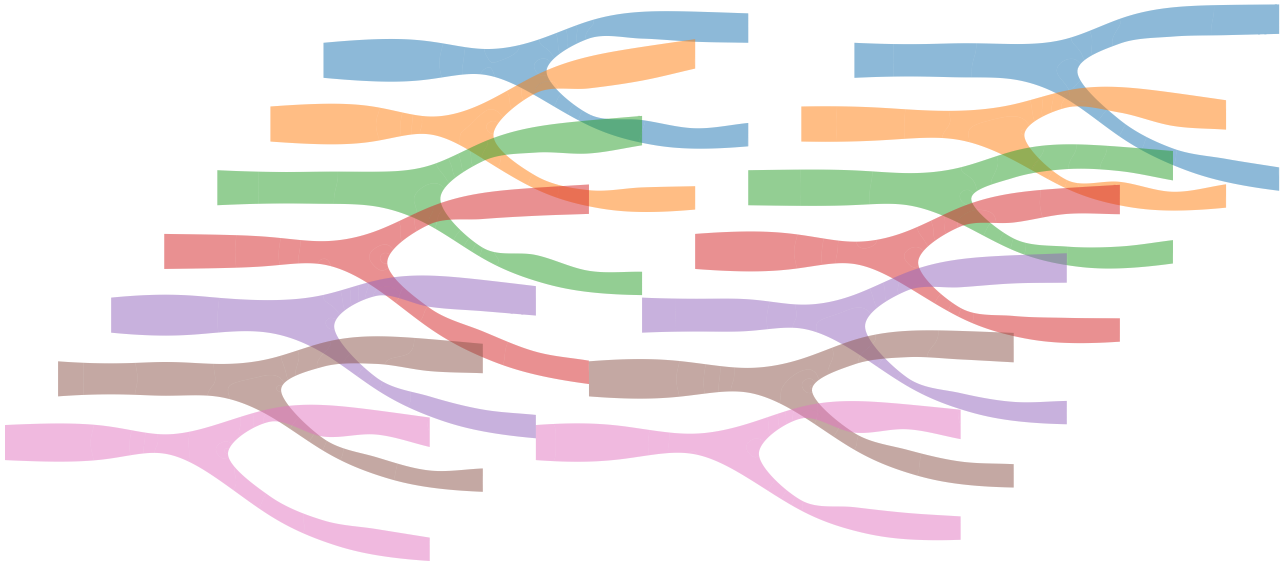


Fig. 7. Test Case 2: representation of some of the geometries $\Omega \in \mathcal{G}$ included in the training dataset.

For each configuration, we perform 500 epochs of the ADAM optimizers [36] followed by 20000 epochs of the BFGS method [37] to ensure convergence of the optimizer. For the case of a training set composed of $N_{sn} = 100$ numerical simulations, we post-process the velocity field to display streamlines. In Fig. 11 we report the streamlines resulting from the different ANN configurations for three test cases extracted from the 40 numerical simulations that formed the

test set: they represent the best, the average, and the worst-case scenarios, respectively.

Training the ANN with a loss function composed only of the data misfit term results insufficient for an accurate reconstruction of the streamlines, especially in low-velocity areas. In Figs. 12 and 13, we show the effect of the loss function components described in Section 3.1 on the streamlines reconstruction.

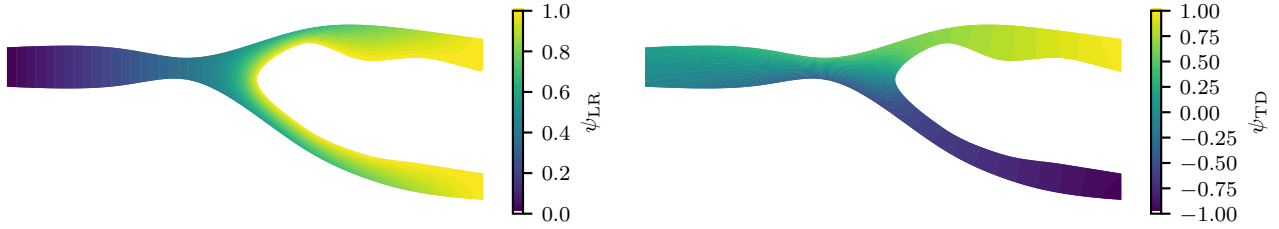


Fig. 8. Test Case 2: fields ψ_{LR} and ψ_{TD} associated with the domain of Fig. 8.

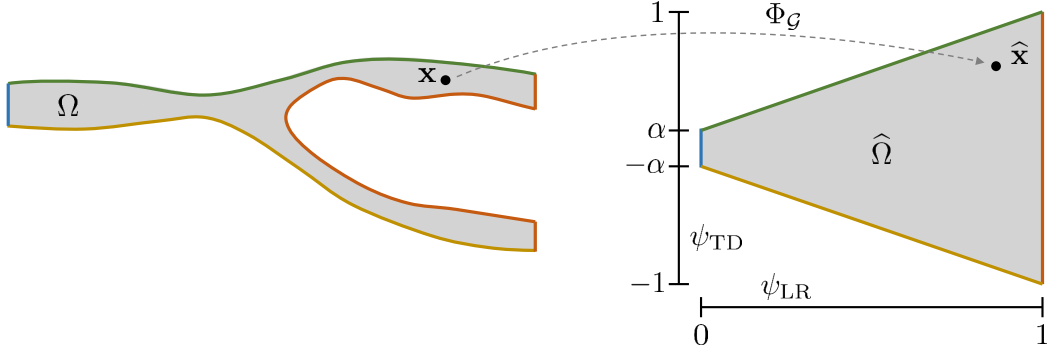


Fig. 9. Test Case 2: representation of the UC system Φ_G . Left: physical domain Ω ; right: reference domain $\hat{\Omega}$.

To assess the generalization error on the test set resulting from different training configurations, we repeat the training considering only a subset of $[25, 50, 100, 200, 400]$ numerical simulations as training set. The test set comprises 40 numerical simulations sampled in 10000 points. We repeat the training with 10 different random initializations of the ANN weights and biases. The average root mean squared error (RMSE) of the velocity magnitude and direction are reported in Fig. 14, together with bands indicating the maximum and the minimum values for the different training set dimensions and ANN configurations.

4.2 Test Case 2: coronary bifurcation

We consider a training set consisting of $N_{sn} = 500$ different geometries. For each geometry, we take the solution in $N_{pt} = 1000$ randomly generated points. Then, for both the landmark configurations (26 landmarks and 6 landmarks), we train a PC-USM-Net and a UC-USM-Net. We consider a FCNN with 4 inner layers, respectively consisting of 20, 15, 10, and 5 neurons. This architecture has been tuned in order to minimize the validation error on a set of 100 geometries not included in the training dataset. To train the FCNN weights and biases, we run 200 iterations of the Adam optimizer [36] and, subsequently, 5000 iterations of the BFGS algorithm [37]. We perform 10 different training runs for each configuration, starting from different random initializations of the FCNN parameters. Each training run lasts about 45 minutes on a laptop equipped with Intel Core i7-1165G7 CPU (2.80 GHz).

In Fig. 15, we show boxplots of the errors associated with a testing dataset of 100 geometries, neither in the training nor in the validation dataset. As expected, USM-Nets

that are provided with 26 landmarks generate more accurate predictions than those that are aware of only 6 landmarks. However, we notice that USM-Nets based on only 6 landmarks still have a noticeable accuracy (relative RMSE error of about 3% on both the velocity and pressure). This figure is compatible with the levels of precision typically required in clinical practice.

Furthermore, the boxplots show that the use of UC can significantly enhance the performance of the USM-Net. The improvement is all the more evident in the case of the velocity field, compared to the pressure field.

In order to highlight the role that a UC system has in improving the generalization accuracy of USM-Nets, we consider the pair of domains in the testing set that are characterized by the most similar landmarks. More precisely, these domains, which we will call Ω^1 and Ω^2 , are such that $\|P_g(\Omega^1) - P_g(\Omega^2)\| < 10^{-4}$. Since landmarks characterize the domain at some control points, two domains with very similar landmarks may differ significantly away from the control points. This is what happens for the two domains considered, in particular in correspondence of the upper out-flow track (see Fig. 16, left). On the right side of Fig. 16 we show a detail of the velocity field obtained for these two domains with a PC-USM-Net and a UC-USM-Net, in comparison with the reference solution obtained by means of the FOM. We recall that the domain Ω affects the PC-USM-Net result only through the landmarks μ_g . Therefore, the PC-USM-Net will provide the same solution for two geometries with identical landmarks. As shown by Fig. 16, this entails that the PC-USM-Net is not very effective in capturing the solution near the edge, where the solution is heavily affected by the geometric details of the domain not captured by the

Training snapshots

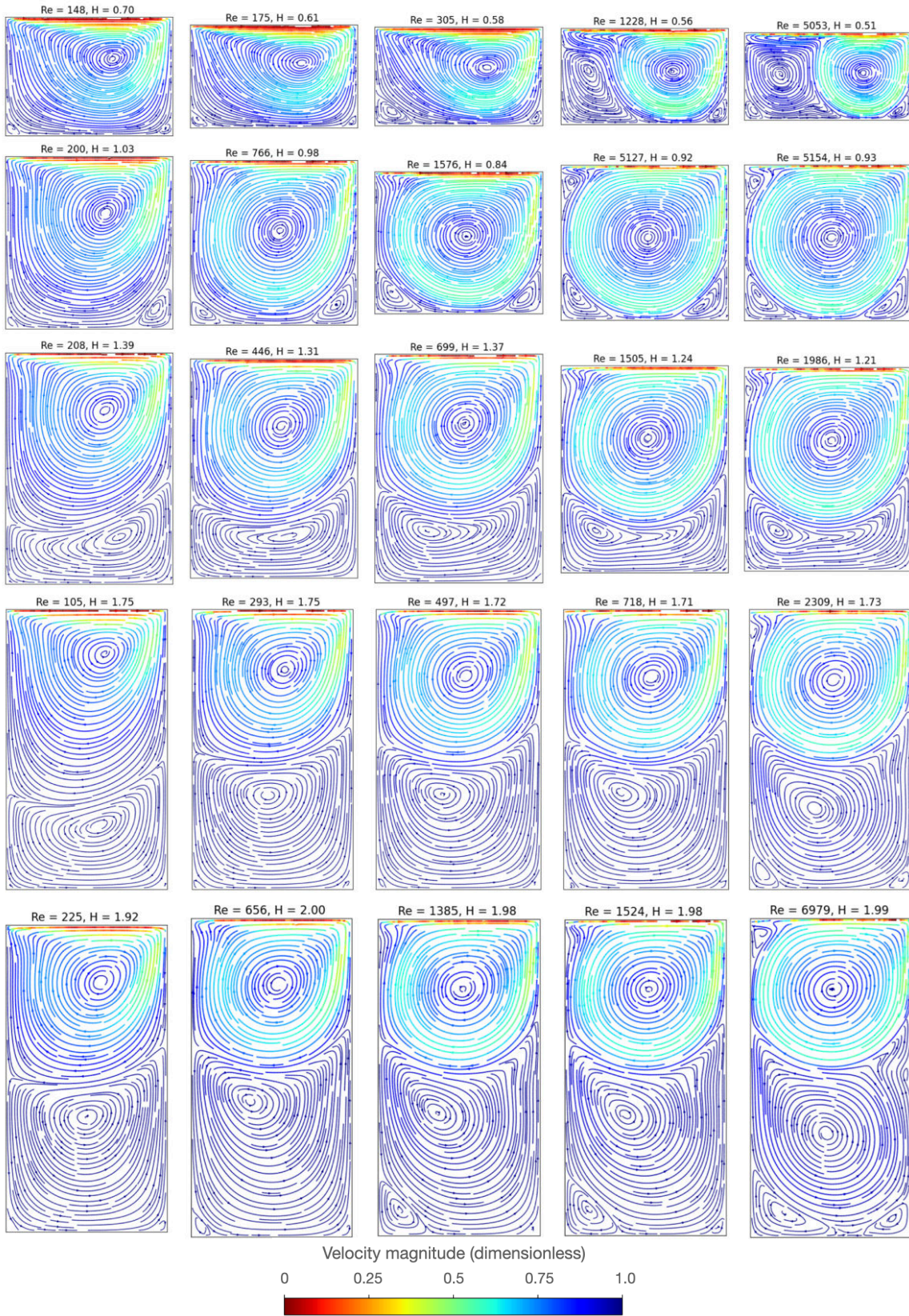


Fig. 10. Test Case 1: comparison of some numerical solutions that constitute the training set. The dataset is generated by approximating the FOM (4) for random samples of the values of the physical and geometrical parameters.

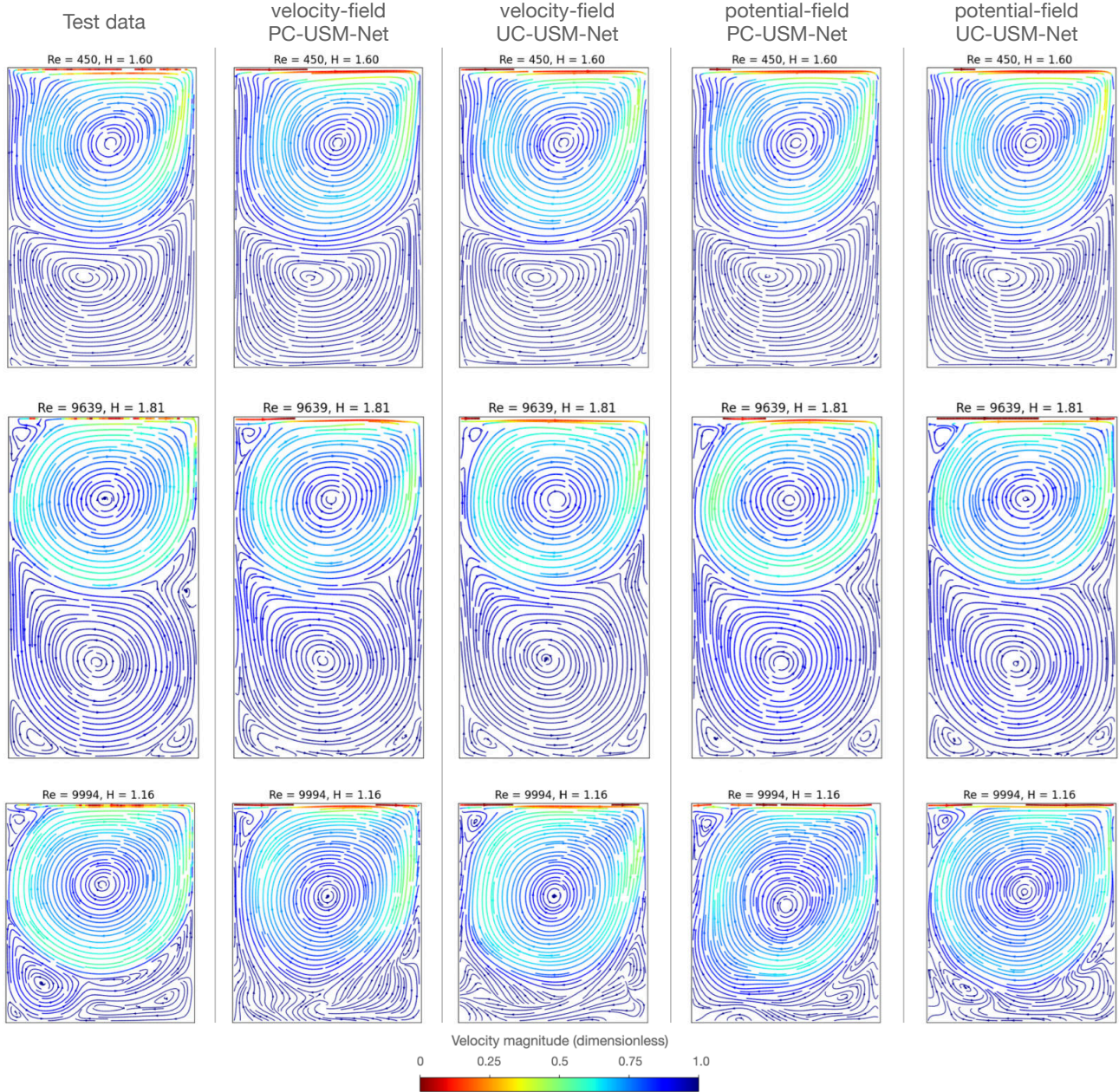


Fig. 11. Test Case 1: comparison of ANN streamlines reconstruction on three numerical solutions from the test set. Test cases are selected to display the range of ANN reconstruction errors in the test set: from minor (best case scenario: first row) to significant (worst case scenario: last row).

landmarks. The use of a UC system is helpful in this regard by defining a model that receives as input, not the physical coordinates, but the reference ones. These coordinates directly encode details of the geometry not captured by the landmarks. In particular, the UC system makes the points belonging to the boundary of the various domains correspond to each other. In this way, UC-USM-Nets are more effective than PC-USM-Nets in capturing the velocity field close to the boundary.

In Figs. 17 and 18 we show the velocity and pressure fields predicted by one of the trained UC-USM-Nets on a subset of the test dataset.

5 Discussion

We have introduced USM-Nets, a deep learning class of surrogate models capable of learning the solution manifold of a PDE universally with respect to physical parameters and geometry.

The ability of a surrogate model to capture the geometrical variability of the solution of a differential problem is a feature of great interest. Indeed, many applications require considering the solution of a physical problem in different domains. Biomedicine offers several examples in this regard since each patient presents a different geometry, and in many cases (as in hemodynamics) the geometry itself is the prin-

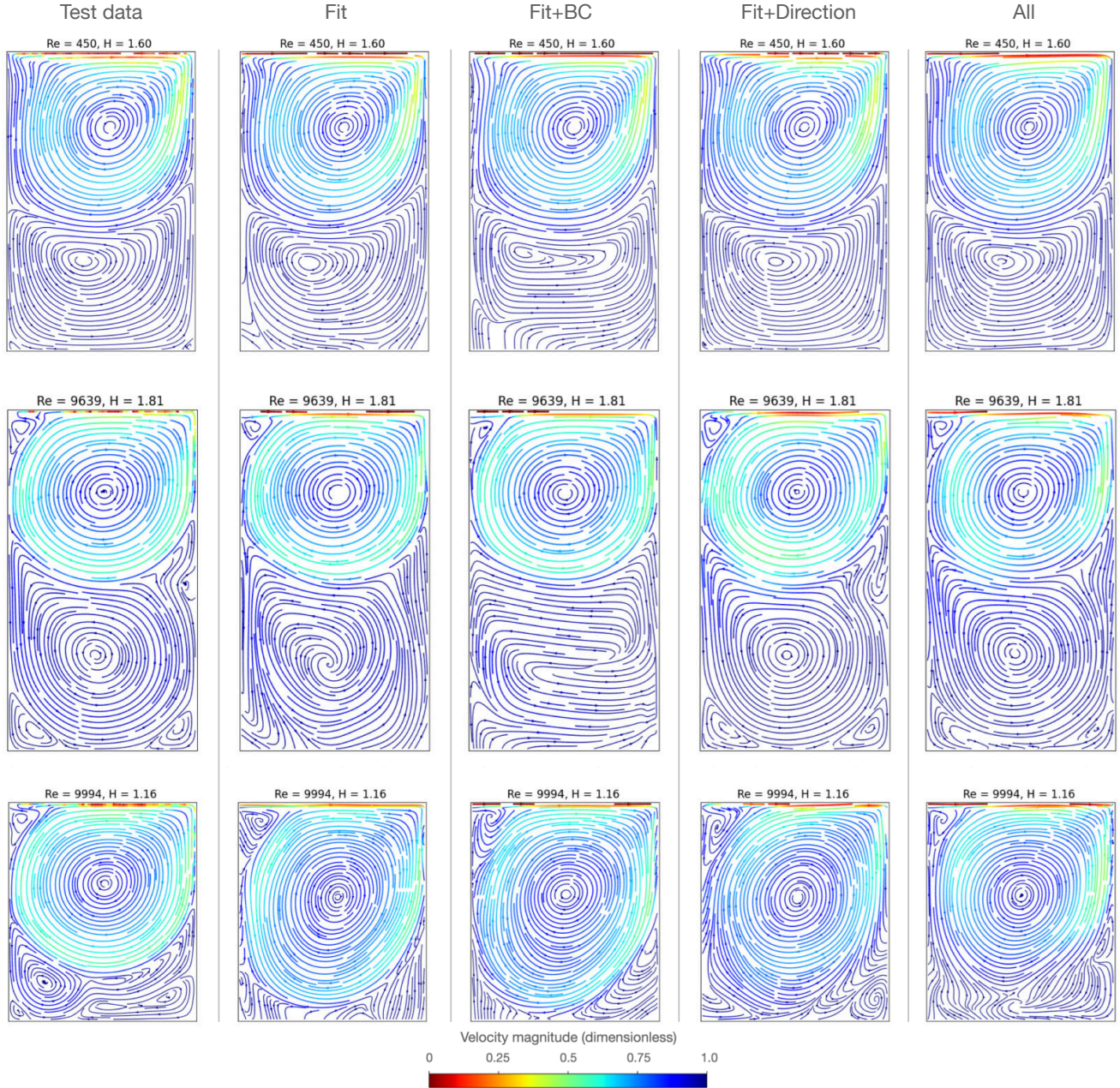


Fig. 12. Test Case 1: comparison of velocity-field PC-USM-Net streamlines reconstruction on three numerical solutions from the test set on varying the definition of the loss function. We consider a loss function composed of the data misfit term (second column), combined with boundary conditions (third column) or with the direction regularization (fourth column). Test cases are selected to display the range of ANN reconstruction errors in the test set: from minor (best case scenario: first row) to significant (worst case scenario: last row).

cial determinant of the solution. Examples are given by the blood flow in an aneurysm, in a stenotic artery, or through an artificial valve.

Nevertheless, most of the reduced-order/surrogate models available in the literature consider a fixed domain accounting only for the variability of physical parameters [38, 39, 40, 41, 42]. Few models rely on parametrized shape models that guarantee correspondence between points coming from different shapes, enabling the construction of projection-based models. As a matter of fact, representing a solution manifold in variable geometries is an arduous task.

There are two main difficulties in this regard: (1) how to encode the properties of the geometry at hand and (2) how to construct a discrete representation of the solution that is universal with respect to the shape of the domain.

Concerning point (1), USM-Nets only require the definition of a finite set of scalar quantities, called geometrical landmarks, that characterize the salient properties of the geometry at hand. Landmarks make USM-Nets an extremely flexible technique that can address a wide range of real-world applications. There are different approaches to landmark definition, such as the one based on the statisti-

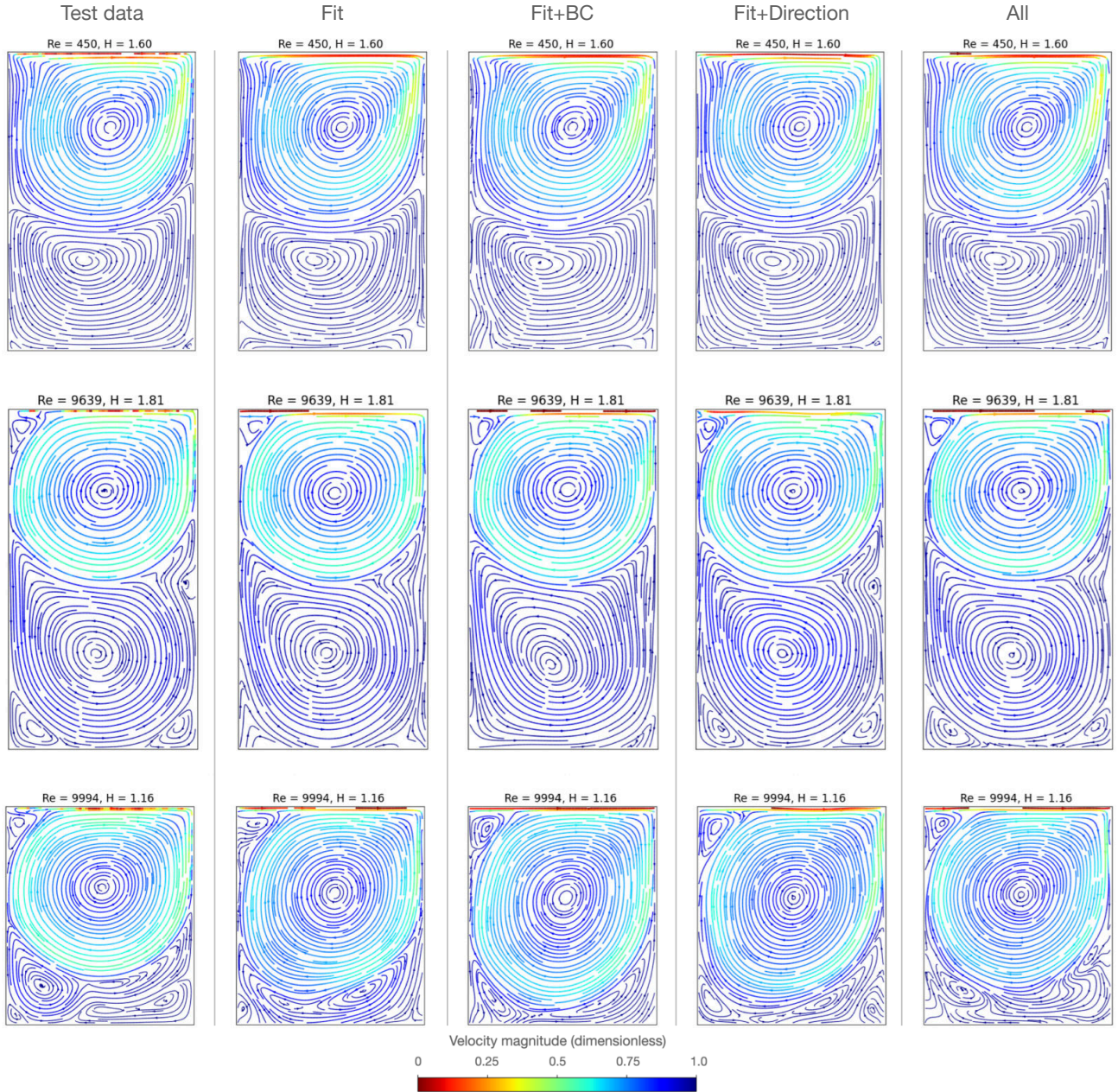


Fig. 13. Test Case 1: comparison of potential-field UC-USM-Net streamlines reconstruction on three numerical solutions from the test set on varying the definition of the loss function. We consider a loss function composed of the data misfit term (second column), combined with boundary conditions (third column) or with the direction regularization (fourth column). Test cases are selected in order to display the range of ANN reconstruction errors in the test set: from minor (best case scenario: first row) to significant (worst case scenario: last row).

cal analysis of sampled geometries, like the first coefficients of proper orthogonal decomposition (POD) or the positions of control points. However, approaches combining POD and ANN [43,44,45,46] might present difficulties in the database construction and limited generalization properties imposed by both the shape model, encoding the correspondence between points belonging to different geometries, and the truncation of the expansion. Landmarks could be simply the coordinates of some points that characterize the geometry at hand. This case, specifically, is well suited for clinical applications. Landmarks, such as the coordinates of a bifurcation,

the position of an inlet, diameters, or areas can be processed directly from medical images without the need for segmentation and the generation of computational grids. The great flexibility of USM-Nets lies in the fact that no structural requirements are imposed on the definition of landmarks.

Concerning point (2), a key feature of USM-Nets is their mesh-less nature, which frees them from a predetermined triangulation of the domain, overcoming the technical difficulties related to mesh element deformations. The mesh-less nature of USM-Nets is achieved by their architectural design. Unlike many existing surrogate modeling methods, that pro-

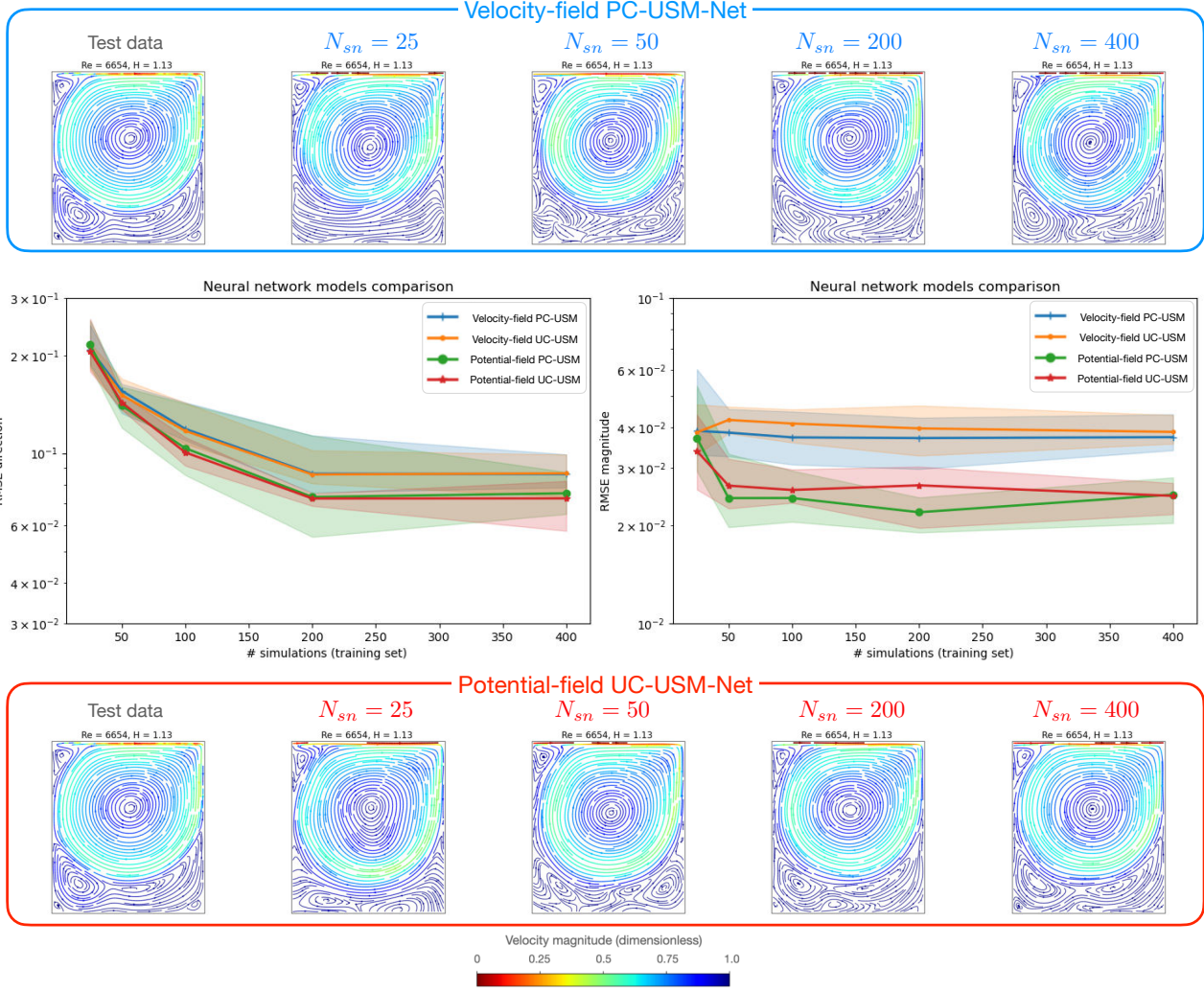


Fig. 14. Test Case 1: RMSE on the velocity magnitude and direction on the test set, made of 40 numerical simulations sampled in 10000 points. We compare the four different configurations of ANN on varying the dimension of the training set. The error in the direction is significantly larger than the error in the magnitude and is inversely proportional to the number of simulations of the training set.

vide a map from the problem parameters to a set of degrees of freedom associated with a preconstructed parametrization of the solution trial manifold, the output of USM-Nets is the solution itself evaluated in a query point. Indeed, by fixing a given parameter vector μ_p and a given geometry Ω , the USM-Net is a function from \mathbb{R}^d to \mathbb{R}^k , that is an approximation of the solution $\mathbf{u}(\cdot; \mu_p, \Omega)$. Hence, instead of passing through a parametrization of the approximate solution, we make the ANN *coincide* with the approximate solution. A further advantage of this architectural design is that USM-Nets encode by construction the spatial correlation (that is, with respect to the input \mathbf{x}) and do not need to learn it, thus achieving elevated accuracy levels even with lightweight NNs.

We have then presented an enhanced version of PC-USM-Nets, called UC-USM-Nets, based on a universal coordinate system. Even if it is not straightforward in all practical cases to define a UC system (such as when the domain may vary in topology), the use of a UC system can

improve the generalization accuracy of PC-USM-Nets, as shown by the numerical results. A UC system acts at two levels. Firstly, it allows us to partially compensate for the possible non-exhaustiveness of the geometrical landmarks in describing the geometry (see also point (1) of the discussion). In Test Case 2, for example, in the setting with only six landmarks, we can have two geometries Ω^1 and Ω^2 that differ from each other, even though they have the same landmarks ($P_g(\Omega^1) = P_g(\Omega^2)$). In boundary areas far from the landmarks, the PC-USM-Net might fail in satisfying the no-slip condition, while the UC-USM-Net, on the other hand, allows the solution to be more accurate because the UC system *informs* the model of the position of the boundary in the geometry at hand. Furthermore, regarding point (2), a UC system provides a more effective representation of the solution manifold. In fact, in this case, the FCNN does not receive as input the coordinates $\mathbf{x} \in \Omega$, but rather $\hat{\mathbf{x}} \in \hat{\Omega}$, which are more informative of the *role* that each point plays within the specific domain.

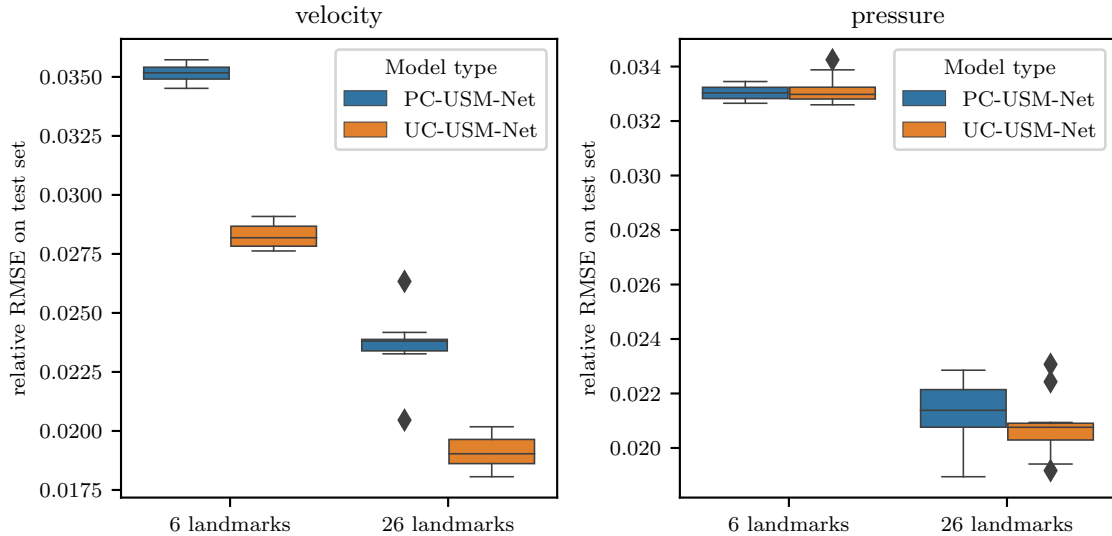


Fig. 15. Test Case 2: boxplots of the errors on the test dataset obtained with 26 landmarks and 6 landmarks and with the PC-USM-Net and the UC-USM-Net architecture. The boxplots refer to 10 training runs obtained starting from different random initializations of the ANN weights and biases. Left: error on the velocity field; right: error on the pressure field.

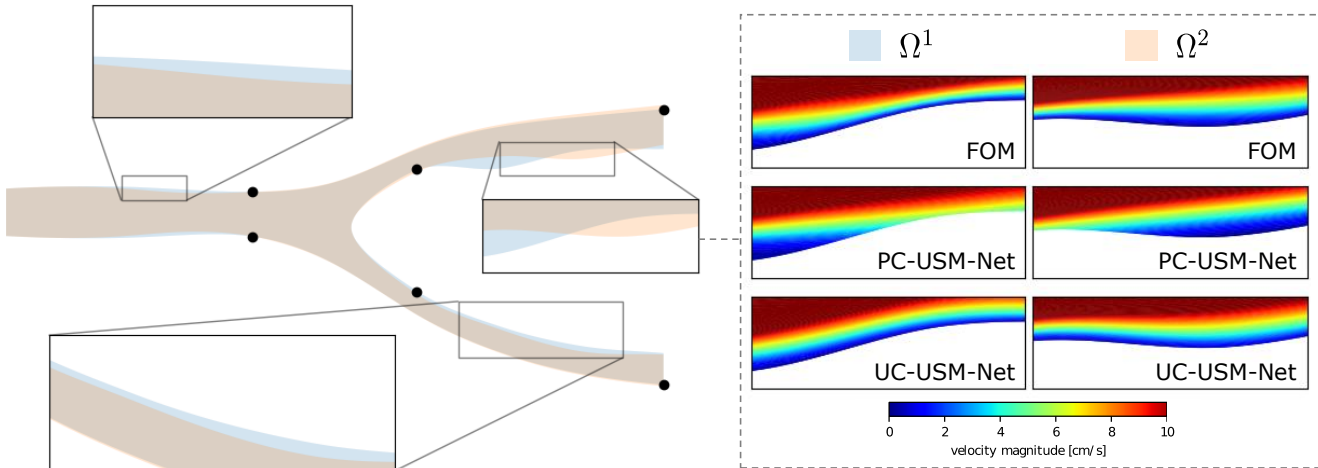


Fig. 16. Test case 2: two domains (Ω^1 and Ω^2) belonging to the testing set that feature almost identical landmarks, that is $P_g(\Omega^1) \simeq P_g(\Omega^2)$ (left). On the right, a detail of the velocity field is compared among FOM solution, PC-USM-Net, and UC-USM-Net surrogates. The color map is intentionally flattened towards low values to highlight velocity variations near the edge.

6 Conclusions

We have proposed a novel technique (USM-Nets), based on ANNs, to build data-driven surrogate models that approximate the solution of differential equations while accounting for the dependence on both scalar physical parameters and the domain geometry. Our method is non-intrusive as it does not require the knowledge of the FOM equations, but rather it is trained with samples of precomputed solution snapshots obtained for different parameters and geometries. It is also meshless since the USM-Net learns the map from point coordinates to the solution. To characterize the geometrical features of the domain at hand, we consider a set of geometrical landmarks defined by the user. Our method is highly flexible, as it does not pose specific requirements on the definition of these landmarks, making it suitable for practical applications

and significantly easing its technological or clinical translation.

We have then presented an enhanced version of our surrogate modeling method, based on a UC system employed to pre-process the physical coordinates. As shown by our numerical results, using this UC system enhances the generalization accuracy in some cases.

We have finally presented two test cases in fluid dynamics. The first is a lid-driven cavity problem with variable geometry and variable Reynolds number; the second one consists in predicting the steady-state pressure and velocity field within a coronary bifurcation, given the patient geometry. In both test cases, despite the noticeable variability of the physical and/or geometrical parameters, USM-Nets were able to approximate the solution within an approximation of the or-

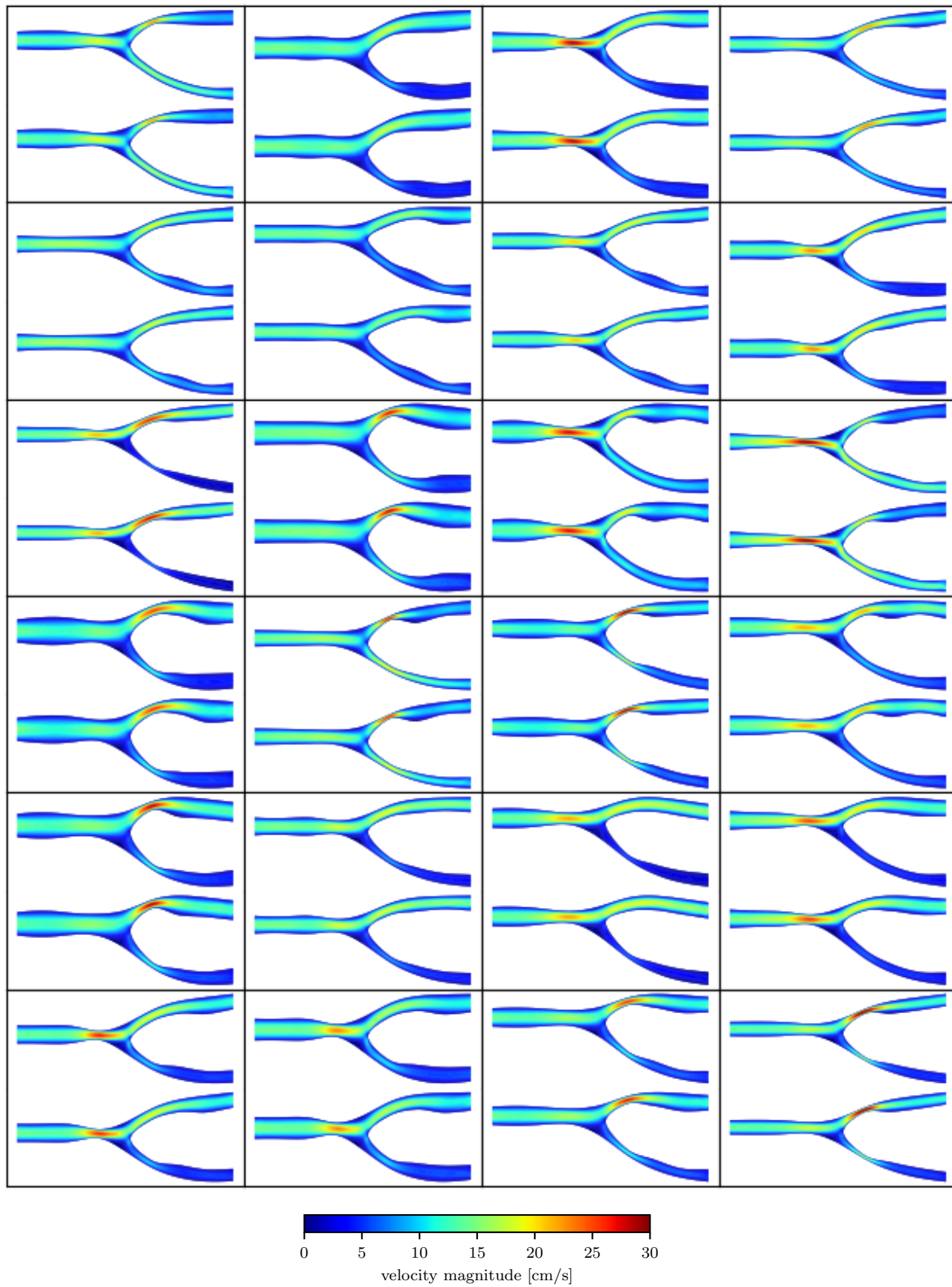


Fig. 17. Test Case 2: comparison of the velocity magnitude field obtained with the FOM (top figure within each box) and with the UC-USM-Net (bottom figure within each box) in a subset of the test set.

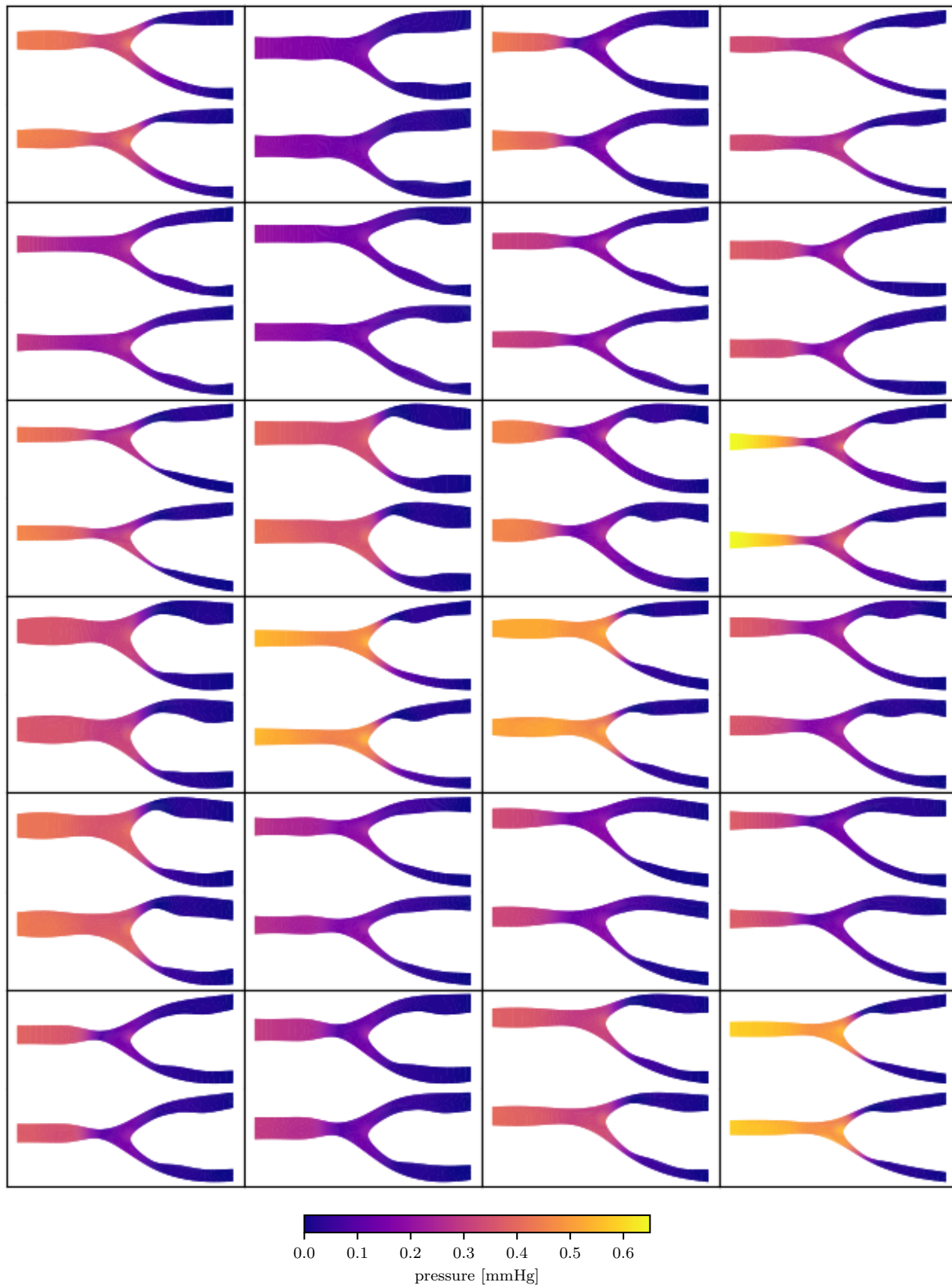


Fig. 18. Test Case 2: comparison of the pressure field obtained with the FOM (top figure within each box) and with the UC-USM-Net (bottom figure within each box) in a subset of the test set.

der of 10^{-2} , being trained, and a few hundreds of solution snapshots.

References

- [1] Alber, M., Buganza Tepole, A., Cannon, W. R., De, S., Dura-Bernal, S., Garikipati, K., Karniadakis, G., Lytton, W. W., Perdikaris, P., Petzold, L., et al., 2019. “Integrating machine learning and multiscale modeling—perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences”. *NPJ digital medicine*, **2**(1), pp. 1–11.
- [2] Anderson, J. D., and Wendt, J., 1995. *Computational fluid dynamics*, Vol. 206. Springer.
- [3] Parolini, N., and Quarteroni, A., 2005. “Mathematical models and numerical simulations for the america’s cup”. *Computer Methods in Applied Mechanics and Engineering*, **194**(9-11), pp. 1001–1026.
- [4] Formaggia, L., Quarteroni, A., and Veneziani, A., 2010. *Cardiovascular Mathematics: Modeling and simulation of the circulatory system*, Vol. 1. Springer Science & Business Media.
- [5] Brunton, S. L., Noack, B. R., and Koumoutsakos, P., 2020. “Machine learning for fluid mechanics”. *Annual Review of Fluid Mechanics*, **52**, pp. 477–508.
- [6] Quarteroni, A., 2017. *Numerical Models for Differential Problems*, 3rd ed. Springer.
- [7] Hughes, T. J., Cottrell, J. A., and Bazilevs, Y., 2005. “Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement”. *Computer methods in applied mechanics and engineering*, **194**(39-41), pp. 4135–4195.
- [8] Sederberg, T. W., and Parry, S. R., 1986. “Free-form deformation of solid geometric models”. In Proceedings of the 13th annual conference on Computer graphics and interactive techniques, pp. 151–160.
- [9] Lamousin, H. J., and Waggenspack, N., 1994. “Nurbs-based free-form deformations”. *IEEE Computer Graphics and Applications*, **14**(6), pp. 59–65.
- [10] Buhmann, M. D., 2000. “Radial basis functions”. *Acta numerica*, **9**, pp. 1–38.
- [11] Heimann, T., and Meinzer, H.-P., 2009. “Statistical shape models for 3d medical image segmentation: a review”. *Medical image analysis*, **13**(4), pp. 543–563.
- [12] Jolliffe, I. T., and Cadima, J., 2016. “Principal component analysis: a review and recent developments”. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, **374**(2065), p. 20150202.
- [13] Quarteroni, A., Rozza, G., and Manzoni, A., 2011. “Certified reduced basis approximation for parametrized partial differential equations and applications”. *Journal of Mathematics in Industry*, **1**(1), pp. 1–49.
- [14] Carlberg, K., Bou-Mosleh, C., and Farhat, C., 2011. “Efficient non-linear model reduction via a least-squares petrov–galerkin projection and compressive tensor approximations”. *International Journal for numerical methods in engineering*, **86**(2), pp. 155–181.
- [15] Quarteroni, A., Manzoni, A., and Negri, F., 2015. *Reduced basis methods for partial differential equations: an introduction*, Vol. 92. Springer.
- [16] Hesthaven, J. S., Rozza, G., Stamm, B., et al., 2016. *Certified reduced basis methods for parametrized partial differential equations*, Vol. 590. Springer.
- [17] Peherstorfer, B., and Willcox, K., 2016. “Data-driven operator inference for nonintrusive projection-based model reduction”. *Computer Methods in Applied Mechanics and Engineering*, **306**, pp. 196–215.
- [18] Taira, K., Brunton, S. L., Dawson, S. T., Rowley, C. W., Colonius, T., McKeon, B. J., Schmidt, O. T., Gordeyev, S., Theofilis, V., and Ukeiley, L. S., 2017. “Modal analysis of fluid flows: An overview”. *Aiaa Journal*, **55**(12), pp. 4013–4041.
- [19] Samareh, J., 2004. “Aerodynamic shape optimization based on free-form deformation”. In 10th AIAA/ISSMO multidisciplinary analysis and optimization conference, p. 4630.
- [20] Lassila, T., and Rozza, G., 2010. “Parametric free-form shape design with pde models and reduced basis method”. *Computer Methods in Applied Mechanics and Engineering*, **199**(23-24), pp. 1583–1592.
- [21] Morris, A., Allen, C., and Rendall, T., 2008. “Cfd-based optimization of aerofoils using radial basis functions for domain element parameterization and mesh deformation”. *International journal for numerical methods in fluids*, **58**(8), pp. 827–860.
- [22] Rendall, T. C., and Allen, C. B., 2008. “Unified fluid–structure interpolation and mesh motion using radial basis functions”. *International journal for numerical methods in engineering*, **74**(10), pp. 1519–1559.
- [23] Manzoni, A., Quarteroni, A., and Rozza, G., 2012. “Model reduction techniques for fast blood flow simulation in parametrized geometries”. *International journal for numerical methods in biomedical engineering*, **28**(6-7), pp. 604–625.
- [24] Sangalli, L. M., Secchi, P., Vantini, S., and Veneziani, A., 2009. “A case study in exploratory functional data analysis: geometrical features of the internal carotid artery”. *Journal of the American Statistical Association*, **104**(485), pp. 37–48.
- [25] Li, S., and Liu, W. K., 2002. “Meshfree and particle methods and their applications”. *Appl. Mech. Rev.*, **55**(1), pp. 1–34.
- [26] Cybenko, G., 1989. “Approximation by superpositions of a sigmoidal function”. *Mathematics of control, signals and systems*, **2**(4), pp. 303–314.
- [27] Mhaskar, H. N., 1996. “Neural Networks for Optimal Approximation of Smooth and Analytic Functions”. *Neural Computation*, **8**(1), Jan., pp. 164–177.
- [28] Montanelli, H., and Du, Q., 2019. “New error bounds for deep relu networks using sparse grids”. *SIAM Journal on Mathematics of Data Science*, **1**(1), pp. 78–92.
- [29] Raissi, M., Perdikaris, P., and Karniadakis, G. E., 2019. “Physics-informed neural networks: A deep learn-

- ing framework for solving forward and inverse problems involving nonlinear partial differential equations”. *Journal of Computational physics*, **378**, pp. 686–707.
- [30] Raissi, M., Yazdani, A., and Karniadakis, G. E., 2020. “Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations”. *Science*, **367**(6481), pp. 1026–1030.
- [31] Regazzoni, F., Pagani, S., Alessandro, C., Alessandro, L., and Quarteroni, A. M., 2021. “A physics-informed multi-fidelity approach for the estimation of differential equations parameters in low-data or large-noise regimes”. *Atti Accad. Naz. Lincei Cl. Sci. Fis. Mat. Natur.*, **32**(3), pp. 437–470.
- [32] Botella, O., and Peyret, R., 1998. “Benchmark spectral results on the lid-driven cavity flow”. *Computers & Fluids*, **27**(4), pp. 421–433.
- [33] Chiastra, C., Iannaccone, F., Grundeken, M. J., Gijssen, F. J., Segers, P., De Beule, M., Serruys, P. W., Wykrzykowska, J. J., van der Steen, A. F., and Wentzel, J. J., 2016. “Coronary fractional flow reserve measurements of a stenosed side branch: a computational study investigating the influence of the bifurcation angle”. *Biomedical engineering online*, **15**(1), pp. 1–16.
- [34] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.
- [35] Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors, 2020. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. *Nature Methods*, **17**, pp. 261–272.
- [36] Kingma, D. P., and Ba, J., 2014. “Adam: A method for stochastic optimization”. *arXiv preprint arXiv:1412.6980*.
- [37] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y., 2016. *Deep learning*, Vol. 1. MIT press Cambridge.
- [38] Benner, P., Mehrmann, V., and Sorensen, D. C., 2005. *Dimension reduction of large-scale systems*, Vol. 45. Springer.
- [39] Antoulas, A. C., Sorensen, D. C., and Gugercin, S., 2000. A survey of model reduction methods for large-scale systems. Tech. rep.
- [40] Lassila, T., Manzoni, A., Quarteroni, A., and Rozza, G., 2014. “Model order reduction in fluid dynamics: challenges and perspectives”. *Reduced Order Methods for modeling and computational reduction*, pp. 235–273.
- [41] Benner, P., Gugercin, S., and Willcox, K., 2015. “A survey of projection-based model reduction methods for parametric dynamical systems”. *SIAM review*, **57**(4), pp. 483–531.
- [42] Peherstorfer, B., and Willcox, K., 2015. “Dynamic data-driven reduced-order models”. *Computer Methods in Applied Mechanics and Engineering*, **291**, pp. 21–41.
- [43] Hesthaven, J. S., and Ubbiali, S., 2018. “Non-intrusive reduced order modeling of nonlinear problems using neural networks”. *Journal of Computational Physics*, **363**, pp. 55–78.
- [44] Carlberg, K. T., Jameson, A., Kochenderfer, M. J., Morton, J., Peng, L., and Witherden, F. D., 2019. “Recovering missing cfd data for high-order discretizations using deep neural networks and dynamics learning”. *Journal of Computational Physics*, **395**, pp. 105–124.
- [45] Dal Santo, N., Deparis, S., and Pegolotti, L., 2020. “Data driven approximation of parametrized pdes by reduced basis and neural networks”. *Journal of Computational Physics*, **416**, p. 109550.
- [46] O’Leary-Roseberry, T., Du, X., Chaudhuri, A., Martins, J. R., Willcox, K., and Ghattas, O., 2021. “Adaptive projected residual networks for learning parametric maps from sparse data”. *arXiv preprint arXiv:2112.07096*.