MOX-Report No. 71/2024

Shape-informed surrogate models based on signed distance function domain encoding

Zhang, L.; Pagani, S.; Zhang, J.; Regazzoni, F.

Shape-informed surrogate models based on signed distance function domain encoding

Linying Zhang[1,2], Stefano Pagani[1], Jun Zhang[2], and Francesco Regazzoni [*1]

[1]MOX, Laboratory of Modeling and Scientific Computing, Dipartimento di Matematica, Politecnico di Milano, Milano, Italy
[2]School of Aeronautic Science and Engineering, Beihang University, Beijing, China

Abstract

We propose a non-intrusive method to build surrogate models that approximate the solution of parameterized partial differential equations (PDEs), capable of taking into account the dependence of the solution on the shape of the computational domain. Our approach is based on the combination of two neural networks (NNs). The first NN, conditioned on a latent code, provides an implicit representation of geometry variability through signed distance functions. This automated shape encoding technique generates compact, low-dimensional representations of geometries within a latent space, without requiring the explicit construction of an encoder. The second NN reconstructs the output physical fields independently for each spatial point, thus avoiding the computational burden typically associated with high-dimensional discretizations like computational meshes. Furthermore, we show that accuracy in geometrical characterization can be further enhanced by employing Fourier feature mapping as input feature of the NN. The meshless nature of the proposed method, combined with the dimensionality reduction achieved through automatic feature extraction in latent space, makes it highly flexible and computationally efficient. This strategy eliminates the need for manual intervention in extracting geometric parameters, and can even be applied in cases where geometries undergo changes in their topology. Numerical tests in the field of fluid dynamics and solid mechanics demonstrate the effectiveness of the proposed method in accurately predict the solution of PDEs in domains of arbitrary shape. Remarkably, the results show that it achieves accuracy comparable to the best-case scenarios where an explicit parametrization of the computational domain is available.

Keywords: Scientific Machine Learning, Surrogate Modeling, Shape Models, Neural Networks, Signed Distance Functions.

1 Introduction

Reduced order models (ROMs) provide efficient and reliable approximations of high-fidelity models, known as full-order models (FOMs), which are often

---

[*]*Corresponding author* (`francesco.regazzoni@polimi.it`)

based on methods like the Finite Element Method (FEM) or the Finite Volume Method (FVM). ROMs are designed to offer a significantly reduced computational cost compared to the corresponding FOMs. ROMs can be classified as either intrusive or non-intrusive. Intrusive ROMs require explicit knowledge of the FOM, such as in projection-based methods [33, 25, 9]. Non-intrusive ROMs, also referred to as surrogate models or emulators, are constructed from a set of input-output pairs generated by the FOM, either without knowledge of the underlying equations or with only partial understanding of the physics involved [23, 26, 6, 35].

In recent years, there has been a notable rise in methods for constructing surrogate models using machine learning tools such as neural networks (NNs), Gaussian processes, and support vector machines [4, 14, 15, 18, 20, 28]. Despite this progress, most existing methods, whether intrusive or non-intrusive, are tailored to a specific geometry and require re-training when the domain shape changes. This limitation is particularly challenging in various applications, including the biomedical field, where the solutions of mathematical models are highly dependent on the shape of the computational domain, which can vary significantly from patient to patient.

Efforts to handle changes in the shape of domains flexibly have leveraged geometrical shape models, which represent different geometries by varying a limited number of parameters. Some of the most commonly used techniques include: free-form deformations [32, 12], where points in the domain are deformed via interpolation of displacements on a lattice grid; radial-basis functions [5], which define a map parameterized by the displacement of a few control points; statistical shape models [8], such as those based on principal component analysis (PCA) [11], which describe the variation of geometries as a weighted sum of modes of variation with respect to an average shape, possibly combined with Large-Deformation Diffeomorphic Metric Mapping [1]; isogeometric analysis (IGA) which, similarly, handles changes in geometry without the need to regenerate a new mesh, by using non-uniform rational B-splines (NURBS) [10].

In the context of model order reduction, shape models have been employed to construct geometric parameterizations that can be explicitly handled by the ROMs in the same way as physical parameters are treated [13, 17, 31]. However, these methods often depend on deforming a predefined mesh, which restricts the range of geometries that can be effectively handled to guarantee sufficient mesh quality. Additionally, in practical scenarios, statistical shape models like PCA may need a large number of modes to accurately capture geometric variability [30]. Moreover, even when shape models significantly reduce the dimensionality of geometric variability, they often fall short in effectively extracting useful features for predicting the solution of differential models defined on the geometry considered (see, e.g., [29]).

This complexity has motivated researchers to shift to geometrically informed deep learning models for reduced-order modeling in variable shape. One architecture designed to manage changes in mesh topology is the Graph Neural Network (GNN) [36], which has been employed to train surrogate models that can generalize to unseen geometries [22]. PointNets [24], namely NNs that can handle point-cloud representations of geometries, thanks on invariant operations with respect to point ordering and on local-global aggregation, provide another tool to inform surrogate models with geometrical features [19]. Convolution-deconvolution operations have been applied to perform an image-to-image re-

gression [7], which however suffers from inaccuracy near the boundary due to the lack of very fine pixels in the geometry images. A recently proposed method addressing limited resolution challenges is USM-Net [27], namely a surrogate modeling technique which applies to differential problems whose solution depends on both physical and geometrical parameters. A USM-Net is based on two main components. The first component is a shape-encoder, which is a map that associates to each geometry a (low-dimensional) vector of scalars, called *shape codes* or *geometrical features*, encoding the main properties of the considered geometry. The second component is a coordinate-based NN (that is a NN taking space coordinates as inputs, also known as neural field [37]), conditioned on both the physical parameters and the geometrical features, that predicts the output of the PDE in each point of the domain. Thanks to their meshless nature, USM-Nets allow to flexibly handle changes of shape of the domain, while relying on lightweight architectures that yield very good generalization properties (see [28] for an investigation on the effectiveness of coordinate-based NNs to approximate the solution of PDEs). Optionally, USM-Nets can be supplemented with a universal coordinate system, consisting in a map of all the different geometries onto a reference one, thus enhancing the surrogate model accuracy [27].

In [27], two shape-encoding techniques were considered. The first technique, applicable when an exact parametrization of the geometries is available, simply consists in employing the parameters used to construct the geometry as shape codes. This approach, despite being highly appealing in contexts like industrial design, is not practical in fields such as geosciences and biomedicine, that is when the domain is not human-designed and an explicit parametrization is unavailable. The second shape-encoding technique, instead, defines the shape codes as the collection of landmarks, corresponding to coordinates of key points of the domain. This approach too features some limitations: it requires a manual intervention for tagging these points and a case-by-case definition, which is unlikely to identify the minimum set of points most informative for the USM-Net about geometric variability. As a result, it is hardly generalizable.

To address these challenges, the goal of this work is to develop an enhanced version of USM-Nets, based on a shape encoding technique that is fully automatic and applicable to a wide range of cases. The proposed method, named SDF-USM-Nets, leverages DeepSDF [21], a NN-based architecture that is able to learn nonlinear shape models based on implicit representations. Specifically, DeepSDFs are trained to approximate the signed distance function (SDF) of the geometries belonging to the training set through a coordinate-based NN, conditioned on a latent code. By combining these two approaches, SDF-USM-Nets are equipped with these key innovations:

- Shape encoding is directly provided by the latent code of a geometrical DeepSDF model, removing the need for user-based geometrical feature extraction.

- The SDF field, employed to define the shape codes, also offers a convenient way of generalizing the concept of the universal coordinate system introduced in [27], by providing the USM-Net with an additional feature that improves, as testified by the numerical tests, its generalization properties across different shapes.

3

- Remarkably, this shape encoding technique is compatible with large deformations and topological changes, thus offering great flexibility.

The paper is organized as follows. First, in Sec. 2, we introduce the notation considered in this paper and we present the proposed method. Then, in Sec. 3, we detail the test cases considered in this work, and in Sec. 4 we provide the associated numerical results. Finally, in Sec. 5 we discuss them and provide some final remarks.

2 Methods

In this Section, we first define the problem of interest. Then, we introduce the proposed SDF-USM-Net architecture and the training strategy.

2.1 Problem definition

We define $\mathbf{u} : \Omega \to \mathbb{R}^{d_{\mathbf{u}}}$ as the quantity of interest (QoI), which may correspond, to provide a couple of practical examples, to the velocity field in a vessel or the displacement field of soft material. This QoI is defined in the spatial domain $\Omega \subset \mathbb{R}^d$, where typically $d = 2, 3$. As a paradigmatic (albeit not unique) case, the QoI might be governed by a PDE in the following form:

$$\begin{aligned} \mathcal{N}(\mathbf{u}) &= 0, \quad \text{in } \Omega, \\ \mathcal{B}(\mathbf{u}) &= 0, \quad \text{on } \partial\Omega, \end{aligned} \tag{1}$$

where $\mathcal{N}$ is a generic differential operator and $\mathcal{B}$ is the operator associated with boundary conditions, defined on the boundary $\partial\Omega$ of the domain.

In numerous practical problems, the solution field $\mathbf{u}(\mathbf{x})$ is intrinsically dependent on the spatial domain Ω itself. We will employ the notation $\mathbf{u}(\mathbf{x}; \Omega)$ to stress this aspect. With a well-posed differential problem (1), there exists a unique solution $\mathbf{u}(\mathbf{x}; \Omega)$, which can be numerically approximated through a full-order model (FOM). Our focus lies in exploring the manifold of fields $\mathbf{u}(\cdot; \cdot)$ generated upon variations of the domain Ω, and learning a solution map $\Omega \to \mathbf{u}(\cdot; \Omega)$ between the variable domain Ω and the QoI $\mathbf{u}(\mathbf{x}; \Omega)$ from a finite collection of $\{\Omega, \mathbf{u}(\cdot; \Omega)\}$ pairs, sampled from a predetermined subset of interest for the practical problems at hand, denoted by $\mathcal{G}$:

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

2.2 SDF-USM-Net

The architecture of our proposed SDF-USM-Net, shown in Fig. 1, consists of two sub-networks, $\mathcal{NN}_{\text{SDF}}(\mathbf{x}, \mathbf{z}_i; \boldsymbol{\theta}_{\text{SDF}})$ and $\mathcal{NN}_{\text{Phys}}(\mathbf{x}, \mathbf{z}_i, DF(\mathbf{x}); \boldsymbol{\theta}_{\text{Phys}})$, which are two fully connected neural networks (FCNN) with trainable parameters $\boldsymbol{\theta}_{\text{SDF}}$ and $\boldsymbol{\theta}_{\text{Phys}}$, respectively, and a trainable shape code vector $\mathbf{z}_i \in \mathbb{R}^k$ for each geometry. Here, $DF(\mathbf{x})$ denotes the distance from a point $\mathbf{x}$ to the nearest surface. An SDF-USM-Net defines a map from the domain $\Omega \in \mathcal{G}$ and the spatial coordinates input $\mathbf{x} \in \Omega$ to the QoI $\mathbf{u}$ associated with the input query point $\mathbf{x}$ and geometry Ω. The first NN, namely $\mathcal{NN}_{\text{SDF}}$, allows to associate to any shape $\Omega_i \in \mathcal{G}$ a finite-dimensional shape code vector $\mathbf{z}_i$ to give a compact
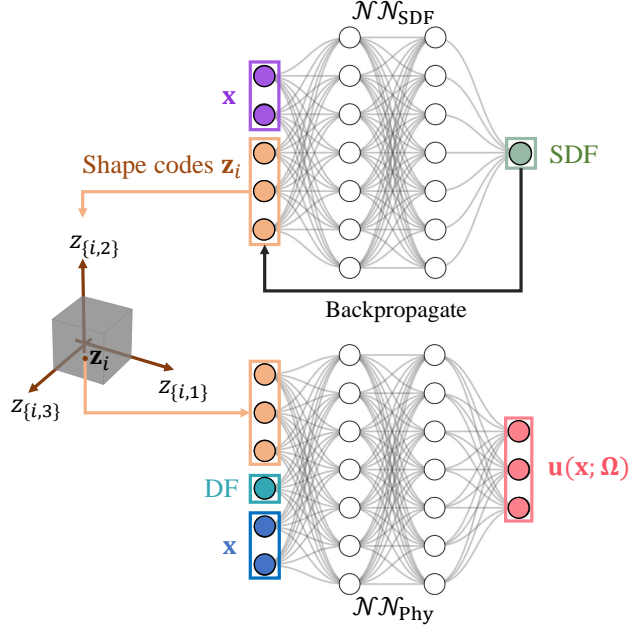
Figure 1: SDF-USM-Net architecture. The network $\mathcal{NN}_{\text{SDF}}$ receives the input spatial coordinates $\mathbf{x}$ and the shape code $\mathbf{z}_i$ and outputs the SDF value for the corresponding point. Within an auto-decoder framework, it learns the shape code distribution in a latent space by reconstructing the continuous SDF in the spatial domain. After the training of $\mathcal{NN}_{\text{SDF}}$, the network $\mathcal{NN}_{\text{Phys}}$ is trained in a supervised manner, taking as input the shape code $\mathbf{z}_i$, spatial coordinates $\mathbf{x}$, and a distance function $DF(\mathbf{x})$ value and returning the approximation of QoI $\mathbf{u}(\mathbf{x}; \Omega)$.

description of Ω_i. The second NN, $\mathcal{NN}_{\text{Phys}}$, associates to the shape code $\mathbf{z}$ the solution in terms of physical field $\mathbf{u}(\mathbf{x}; \Omega)$ corresponding to the input query point $\mathbf{x}$. The two NNs are trained sequentially.

2.2.1 Geometry feature extraction

The construction of $\mathcal{NN}_{\text{SDF}}$ is based on the concept of DeepSDF [21], an implicit function-based shape representation method, that we use here to embed the space of shapes into a low-dimensional latent space (see Figure 2 for a graphical explanation of the model).

In this context, signed distance functions (SDFs) are utilized to describe shapes in the Euclidean space, so that the zero level-set of the function implicitly defines the domain boundary. The SDF is defined as follows

$$SDF(\mathbf{x}) = \begin{cases} d(\mathbf{x}, \partial\Omega) & \text{if } \mathbf{x} \in \Omega, \\ -d(\mathbf{x}, \partial\Omega) & \text{if } \mathbf{x} \notin \Omega, \end{cases} \tag{2}$$

where d denotes the minimum distance of a point $\mathbf{x}$ from the boundary $\partial\Omega$. A NN, denoted as $\mathcal{NN}_{\text{SDF}}$, is trained on a set of N shapes $\{G_i\}_{i=1}^N$. Each shape G_i
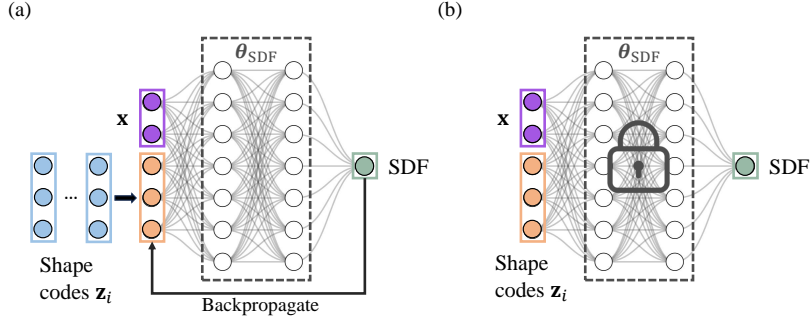
Figure 2: Working principle of $\mathcal{NN}_{\mathrm{SDF}}$. (a) During training, each sample geometry is associated with a randomly initialized shape code. The auto-decoder receives shape codes and coordinates as input. The shape codes are optimized along with the decoder trainable parameters $\boldsymbol{\theta}_{\mathrm{SDF}}$ through standard back-propagation. (b) During inference, the trainable parameters $\boldsymbol{\theta}_{\mathrm{SDF}}$ are fixed, and an optimal shape code $\mathbf{z}_i$ is estimated for any geometry by minimizing the discrepancy between predicted and expected SDF values.

is represented as a set of data pairs $G_i := \{(\mathbf{x}_i^j, s_i^j) : s_i^j = SDF_i(\mathbf{x}_i^j)\}_{j=1}^{K_i}$, where $\mathbf{x}_i^j$ is sampled around the boundary of the shape G_i, and s_i^j is the precomputed SDF value (SDF_i denotes the SDF associated with the domain Ω_i).

With an FCNN architecture, $\mathcal{NN}_{\mathrm{SDF}}$ receives as input the space coordinate $\mathbf{x}$ and a trainable vector $\mathbf{z}_i \in \mathbb{R}^k$ which will be later interpreted as the shape code that represents the target shape in the k-dimensional latent space. The FCNN output is an estimate of the SDF value associated with the input coordinate $\mathbf{x}$. The dimension of the latent space k is set by the user and should be regarded as a hyperparameter. The trainable parameters $\boldsymbol{\theta}_{\mathrm{SDF}}$ of FCNN and shape codes $\{\mathbf{z}_i\}_{i=1}^N$ are learned from the training set $G^{\mathrm{train}} := \{G_i\}_{i=1}^N$ to make the output of the NN a good approximator of the given SDF of the target domain Ω:

$$SDF_i(\mathbf{x}) \approx \mathcal{NN}_{\mathrm{SDF}}(\mathbf{x}, \mathbf{z}_i; \boldsymbol{\theta}_{\mathrm{SDF}}). \tag{3}$$

More precisely, the trainable parameters $\boldsymbol{\theta}_{\mathrm{SDF}}$ are learned simultaneously to the shape codes $\{\mathbf{z}_i\}_{i=1}^N$ associated with training shapes G_i within an *auto-decoder* framework. Hence, by following [21], we consider a zero-mean multivariate-Gaussian $\mathcal{N}(0, \frac{1}{k}I)$ as the prior distribution $p(\mathbf{z}_i)$ of shape codes, where k is the dimension of latent space, and we compute the parameters $\boldsymbol{\theta}_{\mathrm{SDF}}$ and $\mathbf{z}_i$ that maximize the posterior across all shapes:

$$
\begin{aligned}
\hat{\boldsymbol{\theta}}_{\mathrm{SDF}} &= \arg\max_{\boldsymbol{\theta}_{\mathrm{SDF}}} \sum_{i=1}^N \max_{\mathbf{z}_i} \log(p_{\boldsymbol{\theta}_{\mathrm{SDF}}}(\mathbf{z}_i|G_i)) \\
&= \arg\max_{\boldsymbol{\theta}_{\mathrm{SDF}}} \sum_{i=1}^N \max_{\mathbf{z}_i} \left[\log(p_{\boldsymbol{\theta}_{\mathrm{SDF}}}(G_i|\mathbf{z}_i)) + \log(p(\mathbf{z}_i))\right] \\
&= \arg\max_{\boldsymbol{\theta}_{\mathrm{SDF}}} \sum_{i=1}^N \max_{\mathbf{z}_i} \left[\sum_{j=1}^{K_i} \log(p_{\boldsymbol{\theta}_{\mathrm{SDF}}}(s_i^j|\mathbf{z}_i)) + \log(p(\mathbf{z}_i))\right].
\end{aligned}
\tag{4}
$$

We assume that the likelihood takes the form

$$p_{\boldsymbol{\theta}_{\mathrm{SDF}}}(s_i^j|\mathbf{z}_i) = \exp\{-\mathcal{L}(s_i^j, \mathcal{NN}_{\mathrm{SDF}}(\mathbf{x}_i^j, \mathbf{z}_i; \boldsymbol{\theta}_{\mathrm{SDF}}))\}, \tag{5}$$

where $\mathcal{L}$ is a loss function penalizing the difference between the predicted SDF and the reference one. Specifically, we consider two options. The first one consists in the L_1 loss function $\mathcal{L}(s, \tilde{s}) = |s - \tilde{s}|$, where s denotes the reference SDF and $\tilde{s}$ the approximated one. The second option consists in the smoothly clamped L_1 loss function:

$$\mathcal{L}(s, \tilde{s}) = |\mathrm{clamp}(s, \beta) - \mathrm{clamp}(\tilde{s}, \beta)|, \tag{6}$$

where, $\mathrm{clamp}_\beta(s) := \beta \tanh(s/\beta)$ is a function to smoothly truncate the SDF values at a controlled distance, aimed at focusing the training on the details of the boundary. Notice that, in the original DeepSFD work [21], a sharply truncated L_1 loss function is employed, corresponding to (6) where the above definition is replaced by $\mathrm{clamp}_\beta(s) := \min(\beta, \max(-\beta, s))$. Our tests suggested that the use of a smoothed clamping is helpful is escaping local minima, due to the lack a flat profile when the absolute value of the argument is larger than β. Furthermore, the use of the truncation function itself may be convenient or not depending upon the specific application. As a rule of thumb, when the geometric dataset under consideration manifests numerous variations in detail, opting to utilize the truncation function may be convenient. Conversely, in cases when the geometric dataset features alterations in the topology of shapes, abstaining from the truncation function likely renders the shape code inference more stable upon topological changes. In Sec. 4 we numerically investigate this aspect.

In conclusion, the final cost function that is minimized with respect to the network parameters $\boldsymbol{\theta}_{\mathrm{SDF}}$ and shape codes $\mathbf{z}_i$ reads:

$$\hat{\boldsymbol{\theta}}_{\mathrm{SDF}}, \{\hat{\mathbf{z}}_i\}_{i=1}^N = \underset{\boldsymbol{\theta}_{\mathrm{SDF}}, \{\mathbf{z}_i\}_{i=1}^N}{\arg\min} \sum_{i=1}^N \left(\sum_{j=1}^{K_i} \mathcal{L}(s_i^j, \mathcal{NN}_{\mathrm{SDF}}(\mathbf{x}_i^j, \mathbf{z}_i; \boldsymbol{\theta}_{\mathrm{SDF}})) + \frac{1}{\sigma^2}||\mathbf{z}_i||_2^2 \right), \tag{7}$$

where the regularization parameter σ balances in practice the SDF reconstruction and the sparsity of codes.

Once the parameters $\hat{\boldsymbol{\theta}}_{\mathrm{SDF}}$ of $\mathcal{NN}_{\mathrm{SDF}}$ are trained, the auto-decoder can be used to get the shape code of a new geometry by minimizing the discrepancy between the SDF values obtained from $\mathcal{NN}_{\mathrm{SDF}}$ and observed SDF samples. More precisely, suppose that a new geometry in the test dataset G^{test} is given in the form of

$$G_i := \{(\mathbf{x}_i^j, s_i^j) : s_i^j = SDF_i(\mathbf{x}_i^j)\}_{j=1}^{K_i}, \quad i > N.$$

Then, the associated shape code $\mathbf{z}_i$ can be obtained through:

$$\hat{\mathbf{z}}_i = \underset{\mathbf{z}}{\arg\min} \sum_{j=1}^{K_i} \mathcal{L}(s_i^j, \mathcal{NN}_{\mathrm{SDF}}(\mathbf{x}_i^j, \mathbf{z}; \hat{\boldsymbol{\theta}}_{\mathrm{SDF}})) + \frac{1}{\sigma^2}||\mathbf{z}||_2^2. \tag{8}$$

In this stage, the shape code $\mathbf{z}_i$ is randomly initialized from the empirical statistical multivariate-Gaussian $\mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\Sigma})$, where $\hat{\boldsymbol{\mu}}$ and $\hat{\Sigma}$ are the sample mean and covariance of the population of the N learned shape codes $\hat{\mathbf{z}}_i$ (in the training set G^{train}), to facilitate a swift convergence towards an optimal $\mathbf{z}_i$.

2.2.2 Physical field prediction

After the $\mathcal{NN}_{\text{SDF}}$ training stage, consisting of learning the distribution of shape codes in the latent space through a set of SDFs, a corresponding shape code vector $\mathbf{z}_i$ can be obtained for each geometry $\Omega_i \subset \mathcal{G}$ (see Fig. 1). We then leverage this association to train a second NN, denoted as $\mathcal{NN}_{\text{Phys}}$, to map each geometry to the QoI $\mathbf{u}(\mathbf{x}; \Omega)$.

$\mathcal{NN}_{\text{Phys}}$ is trained based on a set of M samples $\{U_i\}_{i=1}^{M}$, in which M can in principle be different from the number of training samples N used in $\mathcal{NN}_{\text{SDF}}$. Each physical field in the training set is represented by P_i data points in a set $U_i := \{(\mathbf{x}_i^j, \overline{s}_i^j, \mathbf{z}_i), \mathbf{u}(\mathbf{x}_i^j; \Omega_i)\}_{j=1}^{P_i}$, where $\mathbf{u}(\mathbf{x}_i^j; \Omega_i)$ is the FOM solution. Different from the data pairs considered to train $\mathcal{NN}_{\text{SDF}}$, $\mathbf{x}_i^j$ is sampled inside the computational domain. By $\overline{s}_i^j$ we denote the distance of each point from the domain physical boundary, with the aim to provide the NN with an additional feature to improve generalization among different geometries. Hence, depending on the problem at hand, different definitions for $\overline{s}_i^j$ can be considered, in order to better reflect the underlying physics. Practical examples will be provided in Sec. 4. In general, we define $\overline{s}_i^j = DF_i(\mathbf{x}_i^j)$, namely the distance function $DF_i(\mathbf{x}) = d(\mathbf{x}, \Gamma_i)$, where Γ_i is a suitable subset of $\partial\Omega_i$. In the case when $\Gamma_i = \partial\Omega_i$, we can set $DF_i \equiv SDF_i$, as only points internal to Ω_i are considered. Each data pair $\{\mathbf{x}_i^j, \overline{s}_i^j\}$ is stacked with the shape codes $\mathbf{z}_i$. If the considered geometry is among those used to train $\mathcal{NN}_{\text{SDF}}$ ($i \leq N$), then the shape codes obtained by (7) are employed. Otherwise ($i > N$), shape codes obtained through (8) are used.

Based on an FCNN architecture, $\mathcal{NN}_{\text{Phys}}$ is trained in a supervised manner based on the set $\{U_i\}_{i=1}^{M}$, in order to approximate the FOM solutions as:

$$\mathbf{u}(\mathbf{x}; \Omega_i) \approx \mathcal{NN}_{\text{Phys}}(\mathbf{x}, \mathbf{z}_i, \overline{s}_i^j(\mathbf{x}); \boldsymbol{\theta}_{\text{Phys}}), \tag{9}$$

More precisely, the trainable parameters $\boldsymbol{\theta}_{\text{Phys}}$ are obtained by minimizing the following mean-square error (MSE) cost function:

$$\hat{\boldsymbol{\theta}}_{\text{Phys}} = \underset{\boldsymbol{\theta}_{\text{Phys}}}{\arg\min} \sum_{i=1}^{M} \sum_{j=1}^{P_i} |\mathbf{u}(\mathbf{x}_i^j; \Omega_i) - \mathcal{NN}_{\text{Phys}}(\mathbf{x}_i^j, \mathbf{z}_i, \overline{s}_i^j; \boldsymbol{\theta}_{\text{Phys}})|_2^2. \tag{10}$$

After obtaining the parameters $\hat{\boldsymbol{\theta}}_{\text{SDF}}$ and $\hat{\boldsymbol{\theta}}_{\text{Phys}}$ from the training of $\mathcal{NN}_{\text{SDF}}$ and $\mathcal{NN}_{\text{Phys}}$, respectively, in the inference stage, the solution $\mathbf{u}(\mathbf{x}; \Omega_i)$ for a new geometry can be approximated through the following two steps:

(i) compute the shape code $\hat{\mathbf{z}}_i$, where $i > M$, by minimizing Eq. (8);

(ii) evaluate the solution $\mathbf{u}(\mathbf{x}; \Omega_i)$ through a single feedforward pass in $\mathcal{NN}_{\text{Phys}}$ for a given query point $\mathbf{x}$ together with the shape code $\hat{\mathbf{z}}_i$ and the associated value $\overline{s}$.

2.2.3 Centralization for discrete shape learning

To facilitate the learning of shapes, a centralization step may be implemented (this is exemplified in Test Case 2). In this case, the input coordinates $\mathbf{x}$ to $\mathcal{NN}_{\text{SDF}}$ are centralized using the equation:

$$\hat{\mathbf{x}} = \mathbf{x} - \mathbf{x_0},$$

where $\mathbf{x_0}$ represents the centroid of the shape. Subsequently, the centroid $\mathbf{x_0}$ is concatenated to the shape code, and thus provided as an additional input to $\mathcal{NN}_{\text{Phys}}$.

2.2.4 Normalization and input mapping

To facilitate the training of the NNs, we normalize the input coordinates $\mathbf{x}$ of both $\mathcal{NN}_{\text{SDF}}$ and $\mathcal{NN}_{\text{Phys}}$ and the output physical fields $\mathbf{u}$ to the interval [-1, 1]. Specifically, by considering the interval $[\alpha_{\min}, \alpha_{\max}]$, we preprocess each scalar variable α through the following normalization layer

$$\alpha \mapsto 2\frac{\alpha - \alpha_{\min}}{\alpha_{\max} - \alpha_{\min}} - 1. \tag{11}$$

Moreover, we optionally use Fourier feature mapping [34] γ to preprocess input coordinates before passing them through the FCNN. The function γ maps input points $\mathbf{x} \in \mathbb{R}^d$ to the set of $2m$ sine waves:

$$\gamma(\mathbf{x}) = [\cos(2\pi\mathbf{Bx}), \sin(2\pi\mathbf{Bx})]^{\text{T}}, \tag{12}$$

where each entry in $\mathbf{B} \in \mathbb{R}^{m \times d}$ is sampled from $\mathcal{N}(0, \sigma_{\text{FF}}^2)$. An example is shown in Test Case 2.

2.3 Training algorithm

We divide the model training into two phases, associated with $\mathcal{NN}_{\text{SDF}}$ and $\mathcal{NN}_{\text{Phys}}$ respectively, as described above. For the training of $\mathcal{NN}_{\text{SDF}}$, we initially perform some epochs employing the Adam optimizer, initialized with a learning rate of 10^{-3}. Subsequently, we transition the training process to full-batch training with the L-BFGS optimizer until the loss difference tolerance between two continuous epochs, which is set to be 10^{-8}, is achieved. The GeLU activation function is used in this work.

For the training of $\mathcal{NN}_{\text{Phys}}$, the combination of Adam and L-BFGS optimizer is still utilized. In this part, we consider hyperbolic tangent (tanh) activation functions. We remark that the use of differentiable activation functions, such as tanh, maintain the framework's flexibility for integrating physics-informed loss terms. The number of epochs considered in each test case is reported in Sec. 4.

2.4 Evaluation metrics

For the evaluation of $\mathcal{NN}_{\text{SDF}}$, we employ the Chamfer distance (CD). Each geometry in the test dataset is provided with a set of K data pairs $\{\mathbf{x}_i, s_i\}_{i=1}^K$ for the learning of the shape code $\hat{\mathbf{z}}_i$. With the learned shape code $\hat{\mathbf{z}}_i$, a uniform mesh grid in the space $\{\mathbf{x}_{i,j}\}_{i=1,j=1}^{N_x,N_y}$, where N_x and N_y are the discretization numbers along the x and y axes respectively, is passed through $\mathcal{NN}_{\text{SDF}}$, which outputs the predicted SDF value on this mesh grid. The marching cubes method [16] is then used to identify the zero level set on the grid, and the resulting coordinate set S_1 is considered as the reconstructed geometry by $\mathcal{NN}_{\text{SDF}}$. Given a point set S_2, which contains boundary coordinates of reference geometries, the CD metric is simply the average of the nearest-neighbor distances for each point

to the nearest point in the other point set with N_{S_l} denoting the cardinality of S_l $(l = 1, 2)$:

$$d_{CD}(S_1^i, S_2^i) = \frac{1}{N_{S_1^i}} \sum_{\mathbf{x} \in S_1^i} \min_{\mathbf{y} \in S_2^i} ||\mathbf{x} - \mathbf{y}||_2^2 + \frac{1}{N_{S_2^i}} \sum_{\mathbf{y} \in S_2^i} \min_{\mathbf{x} \in S_1^i} ||\mathbf{x} - \mathbf{y}||_2^2, \qquad (13)$$

where $i = 1, ..., N_t$, and N_t denotes the size of the test dataset. The average of CD is used for the evaluation of the test dataset which consists of N_t samples, given as:

$$\frac{1}{N_t} \sum_{i=1}^{N_t} d_{CD}(S_1^i, S_2^i). \qquad (14)$$

For the evaluation of $\mathcal{NN}_{\text{Phys}}$, we use the relative L_2 error metric. This evaluation method is applied to a test dataset containing M_t samples, denoted as $\{\Omega_i\}_{i=1}^{M_t}$. For each test sample, there are P_i test data points $\{\mathbf{x}_i^j, \overline{s}_i^j\}_{j=1}^{P_i}$, which are randomly sampled within the geometry. These data points, along with the corresponding shape code $\mathbf{z}_i$, are passed through $\mathcal{NN}_{\text{Phys}}$ to obtain the prediction. The relative L_2 error for the entire test dataset is given by:

$$\sqrt{\frac{\sum_{i=1}^{M_t} \sum_{j=1}^{P_i} |\mathcal{NN}_{\text{Phys}}(\mathbf{x}_i^j, \mathbf{z}_i, \overline{s}_i^j; \hat{\boldsymbol{\theta}}_{\text{Phys}}) - \mathbf{u}(\mathbf{x}_i^j, \Omega_i)|^2}{\sum_{i=1}^{M_t} \sum_{j=1}^{P_i} |\mathbf{u}(\mathbf{x}_i^j, \Omega_i)|^2}}. \qquad (15)$$

2.5 SDF-USM-Net workflow

Using SDF-USM-Net, like other surrogate modeling approaches, comprises offline and online stages. Having introduced all the components, we now provide a summary in this section. Algorithm 1 presents the summarized workflow for both the offline and online stages.

Algorithm 1 Workflow of SDF-USM-Net.

Offline stage:

Input: the number of training shapes N for $\mathcal{NN}_{\text{SDF}}$, the number of training shapes M for $\mathcal{NN}_{\text{Phys}}$

 1: Define the N training shapes $\{\Omega_i\}_{i=1}^N$ for $\mathcal{NN}_{\text{SDF}}$
 2: Generate the training set $G^{\text{train}} := \{G_i\}_{i=1}^N$
 3: Train $\mathcal{NN}_{\text{SDF}}$ by solving Eq. (7) until convergence
 4: **if** $M > N$ **then**
 5:     **for** $i = N + 1$ to M **do**
 6:         Define the training shape Ω_i
 7:         Compute the shape codes $\mathbf{z}_i$ by minimizing (8)
 8:     **end for**
 9: **end if**
10: Conduct FOM simulations for M training shapes for $\mathcal{NN}_{\text{Phys}}$ and generate the training set $\{U_i\}_{i=1}^M$
11: Launch the training of $\mathcal{NN}_{\text{Phys}}$ based on the pre-trained set of shape codes until convergence in Eq. (10)

Online stage:
Input: N_t new shapes
Output: Inference of physical fields for N_t new shapes

 1: Convert the new set of shapes into data pairs in test dataset $G^{test} := \{G_i\}_{i=1}^{N_t}$
 2: Infer the shape codes $\mathbf{z}_i$ by minimizing (8)
 3: Feed-forward the new shape codes $\mathbf{z}_i$ together with query points $\mathbf{x}$ and DF values through $\mathcal{NN}_{\text{Phys}}$

2.6   Alternative shape encoding approaches

To demonstrate the effectiveness of the SDF-USM-Net for predicting physical quantities, we compare three types of shape-encoders $P_g : \mathcal{G} \to \mathbb{R}^k$:

- **Latent shape codes**. Shape codes learned by $\mathcal{NN}_{\text{SDF}}$, as described in Sec. 2.2.1.

- **Explicit parameters**. 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 latent codes coincide with the geometrical parameters themselves. An example is provided in Test Case 2.

- **Geometrical Landmarks**. For geometries with complex shapes in which an explicit representation is not available, a straightforward choice is to take the coordinates of key points in the domain as landmarks, as proposed in [27]. An example is provided in Test Case 1.

3   Test cases

In this Section, we introduce two test cases and offer an overview on the methodology employed for data generation.
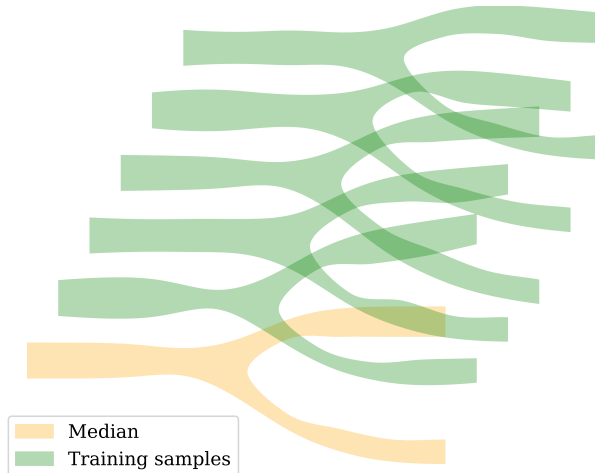
Figure 3: Test Case 1: Representation of some of the geometries $\Omega \in \mathcal{G}$ included in the training dataset in Test Case 1.

3.1 Test Case 1: Coronary bifurcation

As a first test case (Test Case 1), we consider a test proposed in [27], consisting of the prediction of blood flow and pressure fields within a coronary bifurcation afflicted with stenosis. More precisely, our investigation focuses on a computational domain Ω representing a two-dimensional cross-section of a coronary artery featuring a bifurcation.

3.1.1 Geometrical variability

We synthetically generate an extensive array of diverse computational domains, each representing virtual patients, by employing splines. These splines are derived by randomly varying their parameters within defined intervals, designed to emulate the realistic variability observed among patients. A subsample of the geometries obtained following this procedure is displayed in Fig 3. All distances used in this test case are expressed in centimeters (cm). The length of all bifurcations along the x-axis is 4 cm.

3.1.2 SDF calculation

For the initial phase of training $\mathcal{NN}_{\text{SDF}}$, we construct the SDF data pairs G_i for each geometry. These samples comprise 2D points $\mathbf{x}_i^j$ paired with their corresponding SDF values s_i^j.

This process involves the following steps: Firstly, we randomly sample 8000 points along the geometry boundary. Next, we perturb these points with Gaussian noise: 4000 points with a variance of 0.033 and the remaining 4000 points with a variance of 0.0033. This results in a total of 8000 points representing the geometry. This sampling strategy aims to capture detailed SDF information near the boundary. Additionally, we uniformly sample points within an area slightly larger than the computational domain, with a discrete distance of
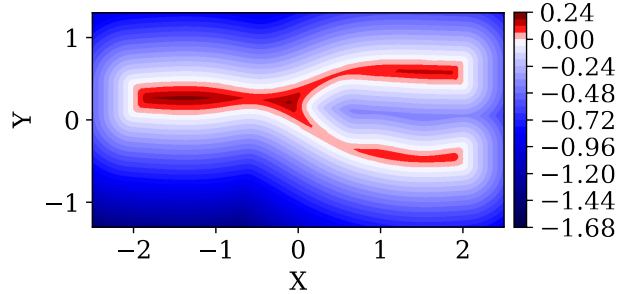
(a)
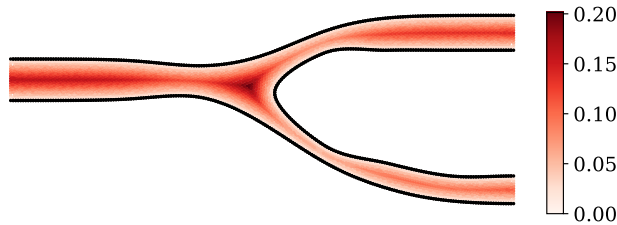
(b)

Figure 4: Test Case 1: (a) A sample of $SDF(\mathbf{x})$ on a uniform grid. (b) A sample of $DF(\mathbf{x})$ on an unstructured grid.

0.1, resulting in around 900 points. Subsequently, for each sampled point $\mathbf{x}$, we determine its SDF value by identifying the closest point in the set of boundary points denoted as P. This proximity search is facilitated using the KD-Tree algorithm[3]. Figure 4(a) shows a sample of generated SDF on a uniform grid.

For the second phase of training $\mathcal{NN}_{\text{Phys}}$, the DF values $\bar{s}$ are computed by determining the minimum distance from the query point $\mathbf{x}$ within the geometry to the physical boundary as shown in Fig. 4(b), which does not include inlet and outlets. Notice that this data generation process relies solely on the coordinates of the boundary. Consequently, the computation of the DF is applicable to both real and synthetic geometries.

3.1.3 Geometrical landmarks

As a benchmark for the present test case, we consider the approach proposed in [27], wherein the domain shape variability is encoded through geometrical landmarks, due to the lack of an explicit parameterization of these geometries in practical problems. Specifically, we define landmarks as the coordinates y of the vessel wall corresponding to a set of predefined coordinates x. In this test case, we will consider a set of landmarks containing 26 coordinates as shown in Fig 5.
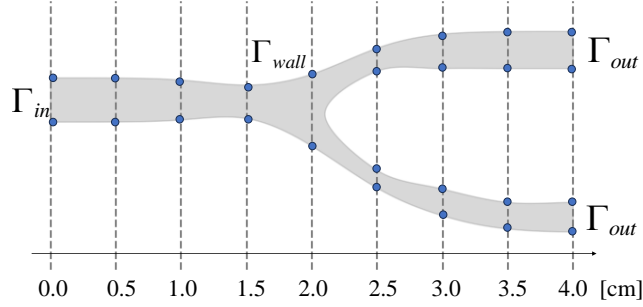
Figure 5: Test Case 1: geometrical landmarks.

3.1.4 Governing equation and solution generation

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

$$
\begin{cases}
-\nu\Delta\mathbf{v} + (\mathbf{v}\cdot\nabla)\mathbf{v} + \dfrac{1}{\rho}\nabla p = \mathbf{0} & \text{in } \Omega, \\[2mm]
\nabla\mathbf{v} = 0 & \text{in } \Omega, \\[2mm]
\mathbf{v} = \mathbf{v}_{\text{in}} & \text{on } \Gamma_{\text{in}}, \\[2mm]
\mathbf{v} = \mathbf{0} & \text{on } \Gamma_{\text{wall}}, \\[2mm]
\nu\dfrac{\partial\mathbf{v}}{\partial\mathbf{n}} - \dfrac{1}{\rho}p\,\mathbf{n} = \mathbf{0} & \text{on } \Gamma_{\text{out}},
\end{cases}
\tag{16}
$$

where $\nu = 4.72$ mL^{2}s^{-1} is the kinetic viscosity of blood and $\rho = 1060$ kg m^{-3} is the density. At the inlet, we consider a parabolic profile with a maximum velocity equal to 14 cm s^{-1}.

We solve Eq. (16) by the finite element method implemented in the free/open-source software FEniCS [2]. To generate training data in the second phase, we utilize a Taylor-Hood Finite Element solver. For spatial discretization, we adopt triangular computational meshes with a spatial resolution of approximately $h = 0.2$ mm. An example solution is shown in Fig. 6.

3.2 Test Case 2: Elastic plate

Test Case 2 involves predicting the displacement of an elastic body with a square shape, featuring geometrical and topological changes defined by random configurations of holes positioned at arbitrary locations.

3.2.1 Geometrical variability

We synthetically generate a wide range of computational domains, each representing plates with one or two randomly positioned holes. This test scenario aims to assess the effectiveness of our proposed model in addressing challenges related to the topological variation of the main features of a geometry.

Furthermore, in this test case, we compare the accuracy achieved with the proposed shape encoding technique to the accuracy obtained with an explicit
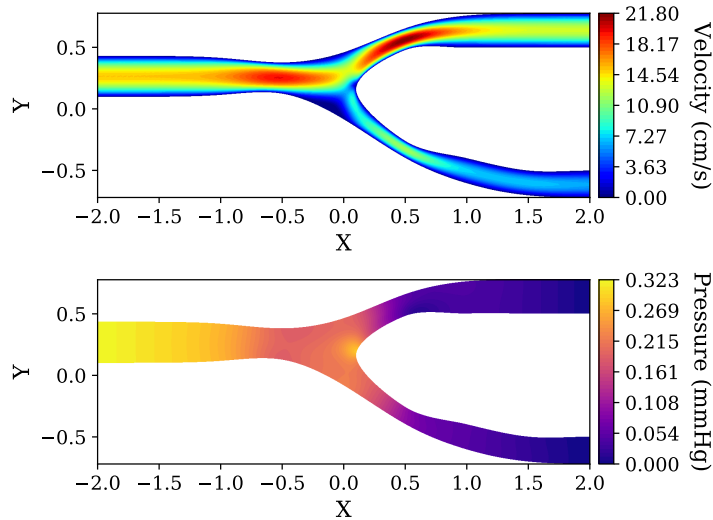
14

Figure 6: Test Case 1: An example of solution.

exact parametrization, to assess how closely the proposed method approaches this best-case scenario. Hence, for any given geometry in the set $\mathcal{G}$, the vector $\mu_g = [x_1, y_1, R_1, ..., x_i, y_i, R_i, N]$, where x_i and y_i are the coordinates of the circle center, R_i is the radius of the circle, and N is the number of the holes, are used as explicit parameters. We define the operator $P_g(\Omega) := \mu_g$ to represent the geometries in the given dataset.

Here we present two sub-cases:

- **Test Case 2a: fixed topology.** To tackle the challenges posed by discrete positional changes in geometry shapes, we train our model using a dataset exclusively comprising plates with two randomly positioned holes of varying radii, parameterized by $\mu_g = [x_1, y_1, R_1, x_2, y_2, R_2, 2]$, where the topology (i.e., the number of holes) remains fixed.

- **Test Case 2b: topology changes.** To address the challenges presented by topology changes, we train our model using a dataset consisting of plates with either one or two randomly positioned holes. In this case, the parameterization $\mu_g = [x_1, y_1, R_1, ..., x_i, y_i, R_i, N]$, where $N \in \{1, 2\}$, accounts for variations in both the positions and the number of holes, reflecting the changes in topology.

3.2.2 SDF calculation

For the initial phase of training our SDF-USM-Net, since each plate shares the same square frame shape, we exclude the square boundary during the generation of SDF values. This exclusion mitigates the influence of the uniform square feature on the model's ability to discern variations among circular shapes.

For the second phase of training our SDF-USM-Net, the DF values are calculated as the minimum distance between a given point inside the physical domain and all the physical boundaries.

3.2.3 Governing equation and solution generation

A square plate made of an isotropic and homogeneous hyperelastic material under finite deformation is considered. We denote by ζ the mapping from the reference configuration to the deformed one, so that the reference coordinate $\boldsymbol{X}$ and the deformed one $\boldsymbol{x}$ are linked by:

$$\boldsymbol{x} = \zeta(\boldsymbol{X}) = \boldsymbol{X} + \boldsymbol{u}(\boldsymbol{X}), \tag{17}$$

where $\boldsymbol{u} \colon \Omega \to \mathbb{R}^2$ denotes the displacement field. We consider traction forces on the left and right boundaries (Γ_{right} and Γ_{left}), and normal spring boundary conditions on the top and bottom boundaries (Γ_{top} and Γ_{bottom}). Hence, the equilibrium configuration is given by the solution of the following PDE, where $\mathbf{P}$ is the first Piola-Kirchhoff stress and $\mathbf{N}$ represents the outward normal unit vector in the initial configuration:

$$\begin{cases} \nabla \cdot \mathbf{P} = \mathbf{0} & \text{in } \Omega, \\ \mathbf{PN} = t\mathbf{N} & \text{on } \Gamma_{\mathrm{right}} \cup \Gamma_{\mathrm{left}}, \\ \mathbf{PN} + K\mathbf{N} \otimes \mathbf{N}\boldsymbol{u} = \mathbf{0} & \text{on } \Gamma_{\mathrm{top}} \cup \Gamma_{\mathrm{bottom}}, \end{cases} \tag{18}$$

with traction force $t = 0.1$, and spring stiffness $K = 0.01$. The constitutive law for the hyperelastic material is written as:

$$\mathbf{P} = \frac{\partial \psi(\mathbf{F})}{\partial \mathbf{F}}, \qquad \mathbf{F} = \nabla\zeta = I + \nabla\boldsymbol{u}, \tag{19}$$

where $\mathbf{F}$ presents the deformation gradient, and ψ denotes the strain energy density of a certain material. In this case, we consider a Neo-Hooken material response, and the strain energy density is given by:

$$\psi(\mathbf{F}) = \frac{1}{2}\mu\left(I_1 - 2\right) - \mu \log(J) + \frac{1}{2}\lambda\left(\log(J)\right)^2, \tag{20}$$

where the first principal invariant is determined by $I_1 = \mathrm{trace}(\mathbf{C})$, the right Cauchy-Green tensor $\mathbf{C}$ is defined as $\mathbf{C} = \mathbf{F}^T\mathbf{F}$, $J = \det(\mathbf{F})$ is the determinant of the deformation gradient. The Lamé parameters are set to $\mu = 0.5$ and $\lambda = 1.3$.

To generate training data, we employ a linear Finite Element solver based on Fenics [2]. For space discretization, we consider triangular computational meshes with a space resolution of nearly $h = 0.3$. Figure 7 shows examples of displacement solutions contained in the training set.

4   Results

In this Section, we report our numerical investigation of the performances of the proposed SDF-USM-Net on the two test cases outlined in Sec. 3.

4.1   Test Case 1: Coronary bifurcation

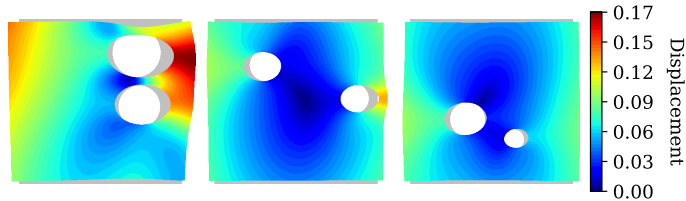In this test case, our training set comprises $N = 500$ distinct geometries.

Figure 7: Test Case 2: Three examples of the circle locations and the corresponding displacement of the elastic plates.

4.1.1 $\mathcal{NN}_{\mathbf{SDF}}$ training results

We employ an FCNN with four inner layers, each comprising 32 neurons, chosen based on a thorough hyperparameter tuning procedure (see A.1 for more details). The smoothed truncation function is utilized to let the training of $\mathcal{NN}_{\mathrm{SDF}}$ focus on the details of the geometries, and the truncation distance β is set to 0.1. The regularization parameter σ is set to 10^2.

The learning rate for the shape codes is set to $10^{-5} \times B$, where B represents the number of shapes in one batch. For each geometry, we select $K_i = 8900$, for $i = 1, ..., N$, randomly generated points for training $\mathcal{NN}_{\mathrm{SDF}}$. During training with the Adam Optimizer, we randomly subsample 625 SDF samples (out of 8900 available points) for each shape in a batch, over 1000 epochs. Subsequently, we transition to full-batch training with the L-BFGS Optimizer, utilizing 1000 subsamples for each shape until the loss difference reaches the tolerance of 10^{-8}.

The dimension of the latent space k emerges as a critical hyperparameter highly impacting the learning capability of the model. In this case, due to the dataset comprising variations in 11 parameters, we conducted experiments around $k = 11$. Figure 8 illustrates the Chamfer distances obtained by varying the latent dimensions from 4 to 25. As observed, an increase in dimension correlates with a decrease in Chamfer distance. However, when the dimension reaches or exceeds 11, the errors on both the training and test sets become nearly equal reaching down to the magnitude of 1×10^{-4}, with no further reduction in error observed with increasing dimension.

Figure 9 presents reconstructions for three samples, showcasing dimensions of 7, 10, 11, and 15, respectively. It is evident that, for dimensions 7 and 10, visible differences persist between the reconstructed shapes and the ground truth, with the case of dimension 7 exhibiting a more dispersed spatial distribution. Conversely, when the dimension is 11 or higher, the reconstructed shapes align closely with the true shapes, indicating that an 11-dimensional space is sufficient to effectively encode the shape distribution in the latent space. Since these geometries have been synthetically generated by varying 11 parameters, we can conclude that, through the analysis of Figure 9, we have been able to re-discover the intrinsic dimension of the set of shapes $\mathcal{G}$ in an automated manner.

4.1.2 Fluid velocity and pressure field prediction

For the training of $\mathcal{NN}_{\mathrm{Phys}}$, we utilize $M_i = 1000$ randomly generated points for each computational domain. An FCNN with four inner layers is employed, comprising 20, 15, 10, and 5 neurons, respectively. To train the FCNN weights
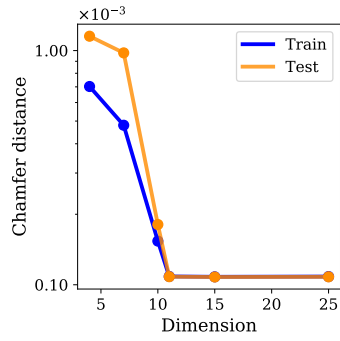
Figure 8: Test Case 1: impact of latent dimensions. Using Chamfer distance as an evaluation metric for the reconstructed shapes, illustration of the relationship between Chamfer distance and latent dimensions, ranging from 4 to 25.
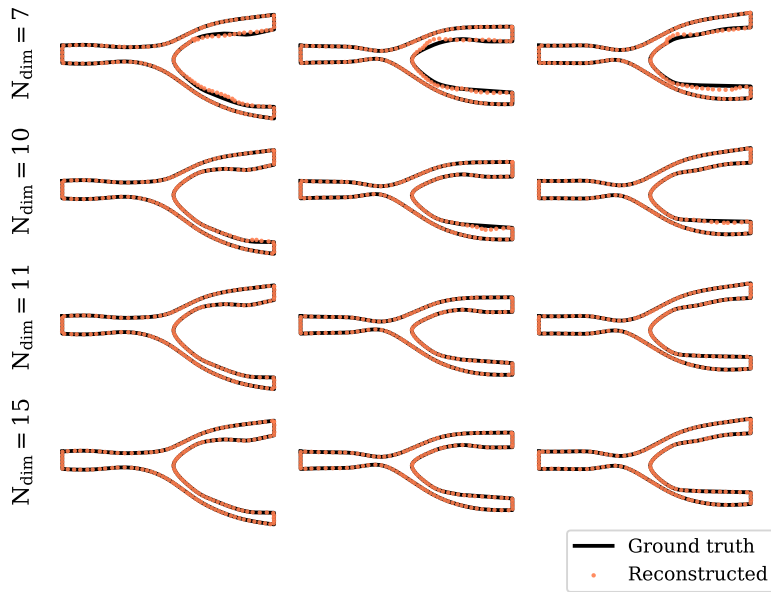

Figure 9: Test Case 1: impact of latent dimension. With each column representing an example from the test dataset, the results represent the reconstructed geometries with an increasing number of latent dimensions (7, 10, 11, 15).

and biases, we employ the Adam optimizer for 200 iterations and then switch to the L-BFGS optimizer until convergence with the loss difference tolerance set to 10^{-8}.

Impact of latent space dimension Based on the experiments of shape reconstruction discussed in Section 4.1.1, it is evident that the dimension of the latent space plays a crucial role in training the SDF-USM-Net. Here, we focus on dimensions around 11 and select 7, 10, 11, and 15. To illustrate the impact of the latent dimension on the reconstruction of the flow field, we randomly selected a sample from the test dataset for visualization in Fig. 10. It is apparent from the visualization that as the dimension increases, the prediction error, particularly that associated with the pressure, exhibits a significant decrease. The relative L_2 error as a function of latent dimension showcases a trend similar to the Chamfer distance. Notably, there is no significant decrease in error observed with dimensions exceeding 11. Based on these experiments, we choose the latent space dimension of 11 for the following experiments.

Impact of input features To highlight the effectiveness of the proposed SDF-USM-Net, we consider four types of input features:

- **USM-Net with landmarks**, which receives as input two spatial coordinates, and the geometrical landmarks.

- **SDF-USM-Net with shape codes**, which receives as input two spatial coordinates, and the latent expression of shape codes.

- **USM-Net with landmarks and DF values**, which receives as input two spatial coordinates, the geometrical landmarks, and the DF values.

- **SDF-USM-Net with shape codes and DF values**, which receives as input two spatial coordinates, the latent expression of shape codes, and the DF values.

We present bar plots depicting the relative L_2 error for velocity and pressure in Fig. 11. From these results, a surrogate model trained with shape codes exhibits comparable error levels to the model trained with landmarks, which encompass all variable parameters of the geometry. Notably, shape codes can outperform landmarks when training is supplemented with the DF values. Furthermore, regardless of whether landmarks or shape codes are used as input, training combined with DF values consistently enhances model performance.

To gain further insight into the influence of different input features, we compare the predicted velocity magnitude obtained with all the considered architectures for a representative geometry belonging to the test dataset in Fig. 12. The prediction results indicate that the error without DF values is notably larger near the boundary, particularly in regions adjacent to the stenosis, where significant shape variations occur. Conversely, models training with DF values exhibit more precise predictions near the boundary.

The comparative results for the same example on pressure are presented in Fig. 13. The disparity between different models is more pronounced here. Although all models can learn at least the coarse distribution of pressure, the SDF-USM-Net trained with shape codes and DF values demonstrates superior prediction capabilities.
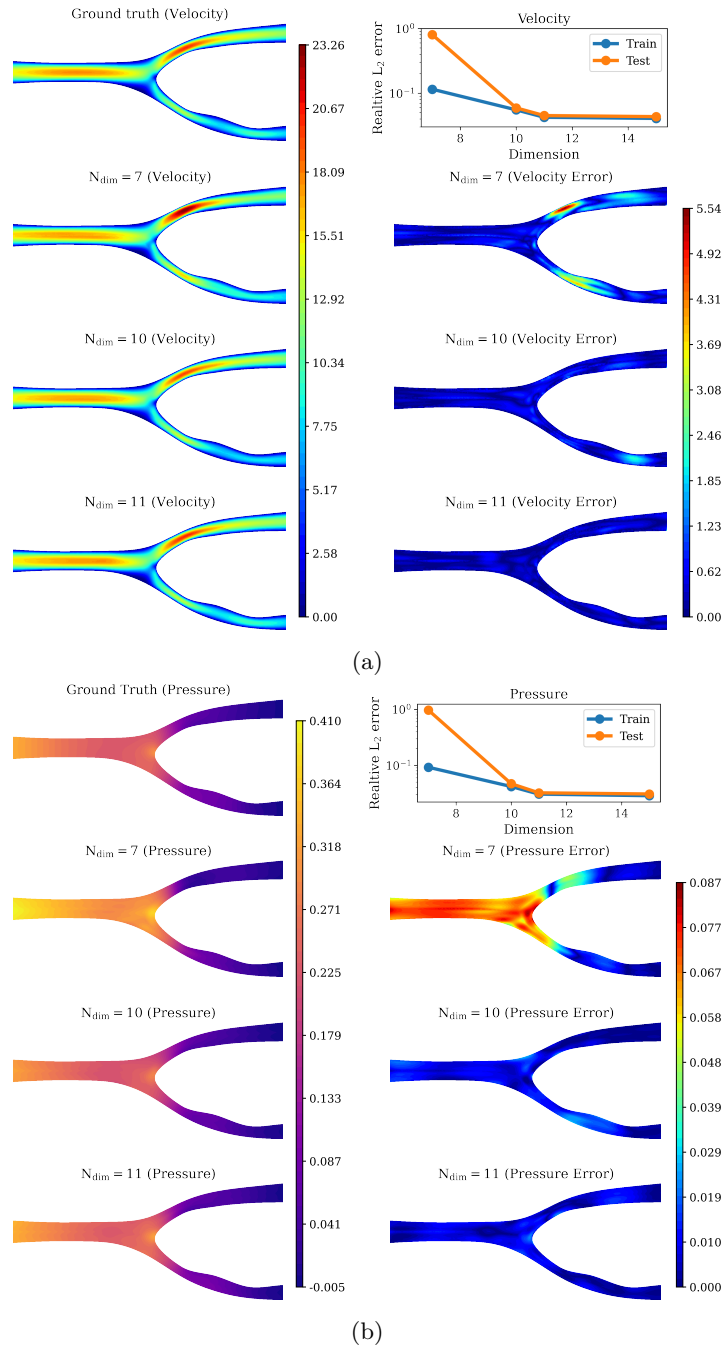
19

Figure 10: Test Case 1: impact of latent space dimensions. We compare, for a sample belonging to the test dataset, the results obtained by using SDF-USM-Net with an increasing number of latent states (7, 10, 11). For (a) velocity magnitude, and (b) pressure, the top-left corner reports the FOM solution, followed by the pressure prediction. The right column reports the absolute error concerning the FOM solution. The relative L_2 error of Test Case 1 as a function of the number of latent dimensions is shown at the top-right corner.
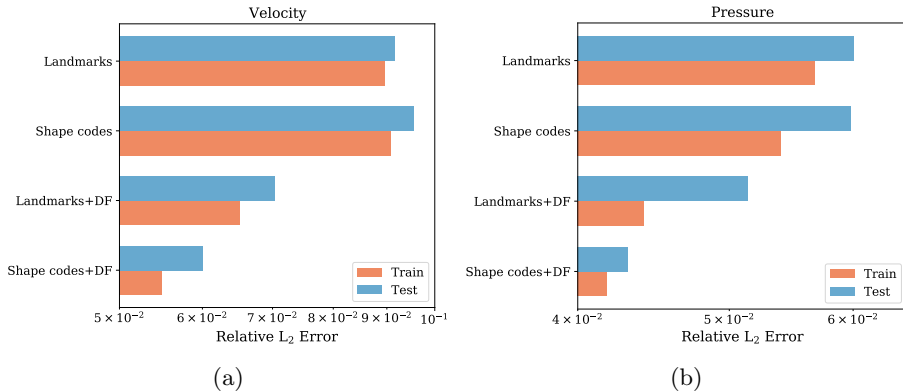
Figure 11: Test Case 1: bar plots of the errors on the test dataset obtained with geometric landmarks and shape codes, and with or without DF as additional feature. Left: error on the velocity field; right: error on the pressure field.

This test case highlights the superior performance in predicting the fluid flow solution when providing DF values as additional features, especially close to the boundary of the domain.

4.2   Test Case 2: Elastic plates

In this section, we first report the results for Test Case 2a (fixed topology) and then for Test Case 2b (variable topology).

4.2.1   Test Case 2a: fixed topology

In this test case, our training set comprises $N = 1000$ distinct geometries. For $\mathcal{NN}_{\text{SDF}}$, we employ an FCNN with four inner layers, each containing 64 neurons. In this test case, we do not make use of a truncation function in training $\mathcal{NN}_{\text{SDF}}$. This choice is explained later, and the impact of the truncation function is investigated.

The shape codes $\mathbf{z}_i$ are initialized from the same normal distribution as defined in Section 2.2.1. The dimension of the latent space is set to 5. We incorporate Fourier feature mapping and centralization, as described in Sections 2.2.4 and 2.2.3, respectively, in the training of $\mathcal{NN}_{\text{SDF}}$. All other training details of $\mathcal{NN}_{\text{SDF}}$ remain consistent with those described in Section 4.1.1.

For the training of $\mathcal{NN}_{\text{Phys}}$, we utilize $M_i = 1000$ randomly generated points for each computational domain. An FCNN with four inner layers is employed, comprising 30, 25, 20, and 15 neurons, respectively.

$\mathcal{NN}_{\text{SDF}}$ was trained on a dataset containing 750 samples and tested on 500 samples without experiencing overfitting (Chamfer distance on training dataset: 8.513×10^{-5}, on test dataset: 8.515×10^{-5}. These combined 1250 shapes were then divided into 80% training and 20% test data for training $\mathcal{NN}_{\text{Phys}}$. Some examples of contours of displacement resulting from SDF-USM-Net compared with the FOM results are shown in Fig. 14. SDF-USM-Net can effectively learn and predict continuous changes in displacement over a given range of geometrical configurations.
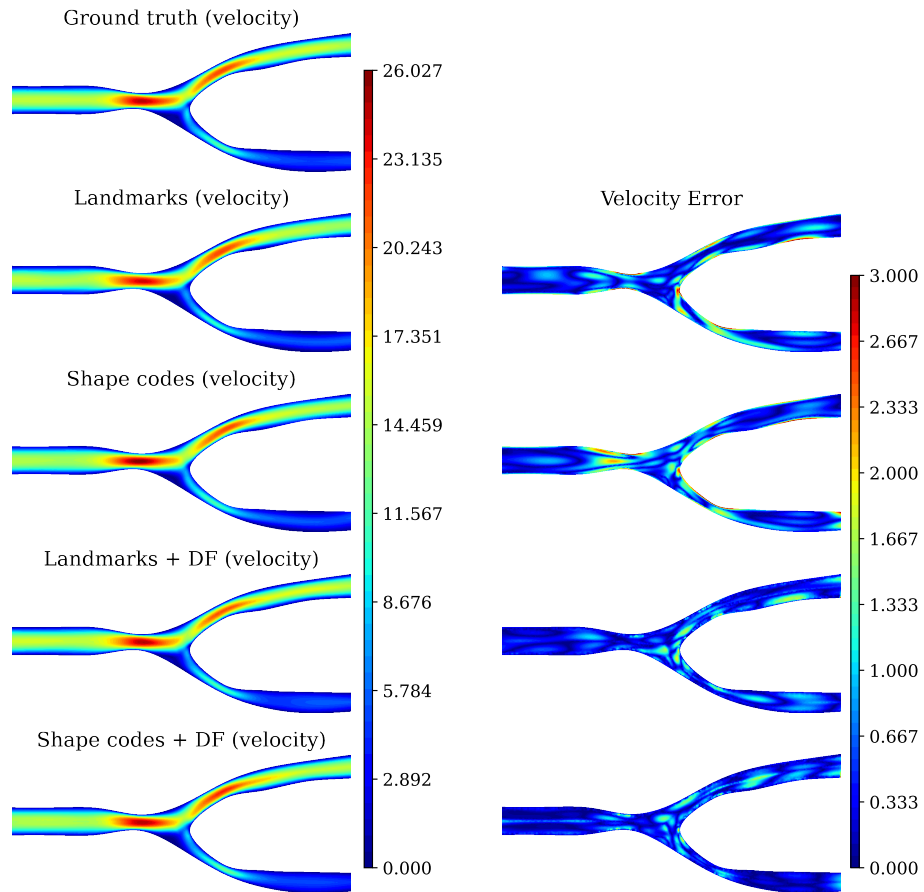
Figure 12: Test Case 1: impact of input features (velocity). We compare, for a sample belonging to the test dataset, the results obtained by using the original USM-Net (with geometrical landmarks) and the proposed SDF-USM-Net (with shape codes), with and without DF values. The top-left corner reports the FOM solution for the velocity magnitude, followed by the prediction obtained with different input features. The right column reports the absolute error associated with the FOM solution.
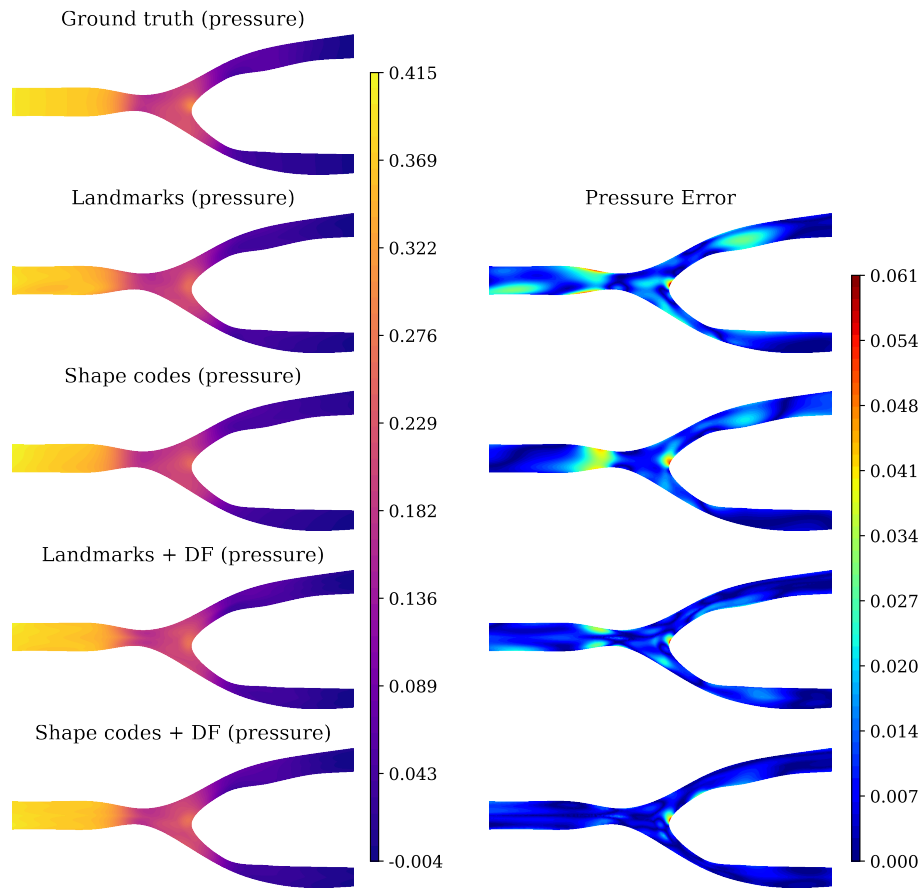
Figure 13: Test Case 1: impact of input features (pressure). We compare, for a sample belonging to the test dataset, the results obtained by using the original USM-Net (with geometrical landmarks) and the proposed SDF-USM-Net (with shape codes), with and without DF values. The top-left corner reports the FOM solution for the pressure, followed by the prediction obtained with different input features. The right column reports the absolute error associated with the FOM solution.
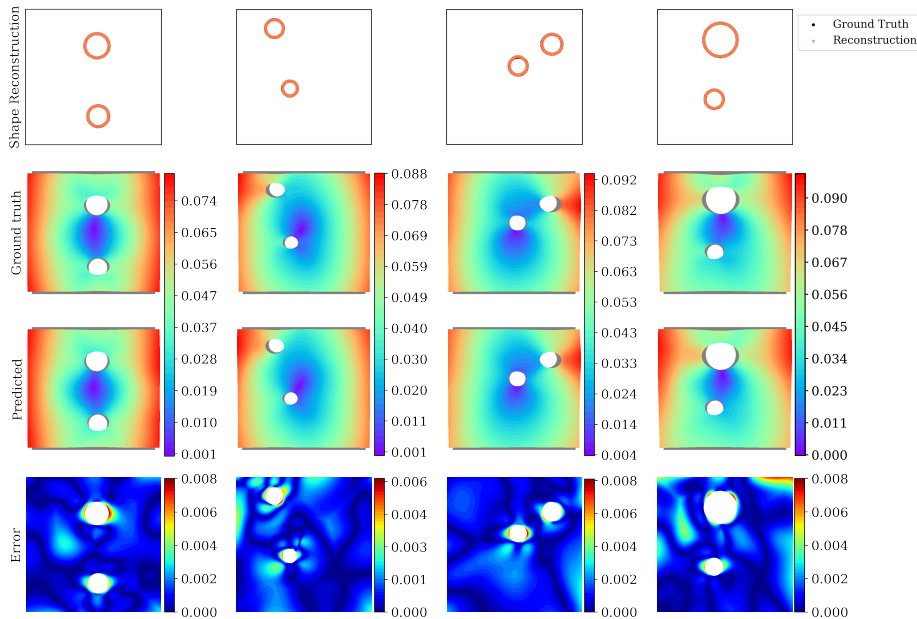
Figure 14: Test Case 2a: comparison of SDF-USM-Net reconstruction on four numerical solutions from the test set: from the best case scenario (first column, with a relative L_2 error of 0.0256) to the worst case scenario (last column, with a relative L_2 error of 0.1330).

Truncation function   Whether to use the truncation function for the training of $\mathcal{NN}_{\mathrm{SDF}}$ has a significant impact on the model's ability to capture macroscopic changes in shapes. Figure 15 illustrates failed reconstructions when the truncation function $\mathrm{clamp}_\beta(\cdot)$ is applied. Each reconstruction, involving the training of shape codes $\mathbf{z}_i$ in the inference stage, is repeated 5 times to select the best reconstruction from different initializations. The results suggest that using the truncation function leads to a higher likelihood of getting trapped in local minima during the training of shape codes in the latent space which results in the failed reconstruction of the geometry.

Impact of input features   We benchmark our proposed method with results obtained by replacing shape codes with the exact parameters used to generate the geometries. This idealized situation provides a way to obtain a reference level, a maximum level of accuracy achievable in principle, since explicit knowledge of the parameters allows the geometry to be encoded exactly. The training results are summarized in Fig 16.

All experiments were repeated three times, and the mean and variance were calculated. These results show that training with input structure that consists of shape codes, remarkably, can achieve almost the same accuracy as training with the explicit knowledge of the parameters that were used to generate the geometry at hand. Moreover, training with input structure that consists of DF values demonstrates slight improvements compared to training without DF values.
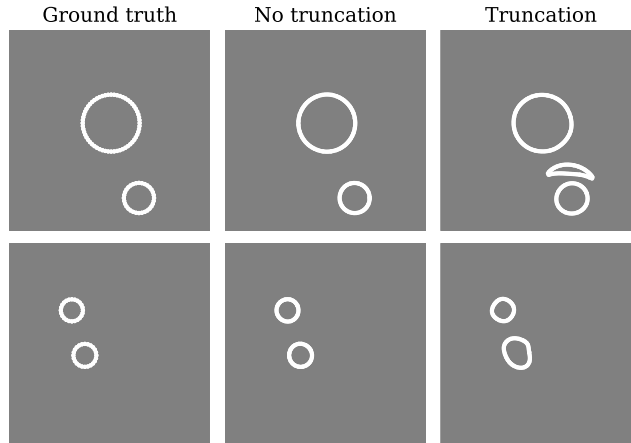
Figure 15: Test Case 2a: impact of truncation function. Two examples of corrupted reconstruction by applying truncation function in $\mathcal{NN}_{\text{SDF}}$.
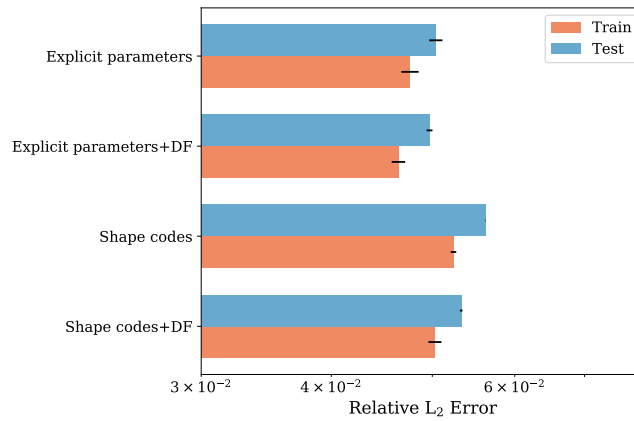

Figure 16: Test Case 2a: barplots of the errors on the training and test dataset obtained with different input features. The barplots refer to 3 training runs obtained starting from different random initializations of ANN weights and biases.
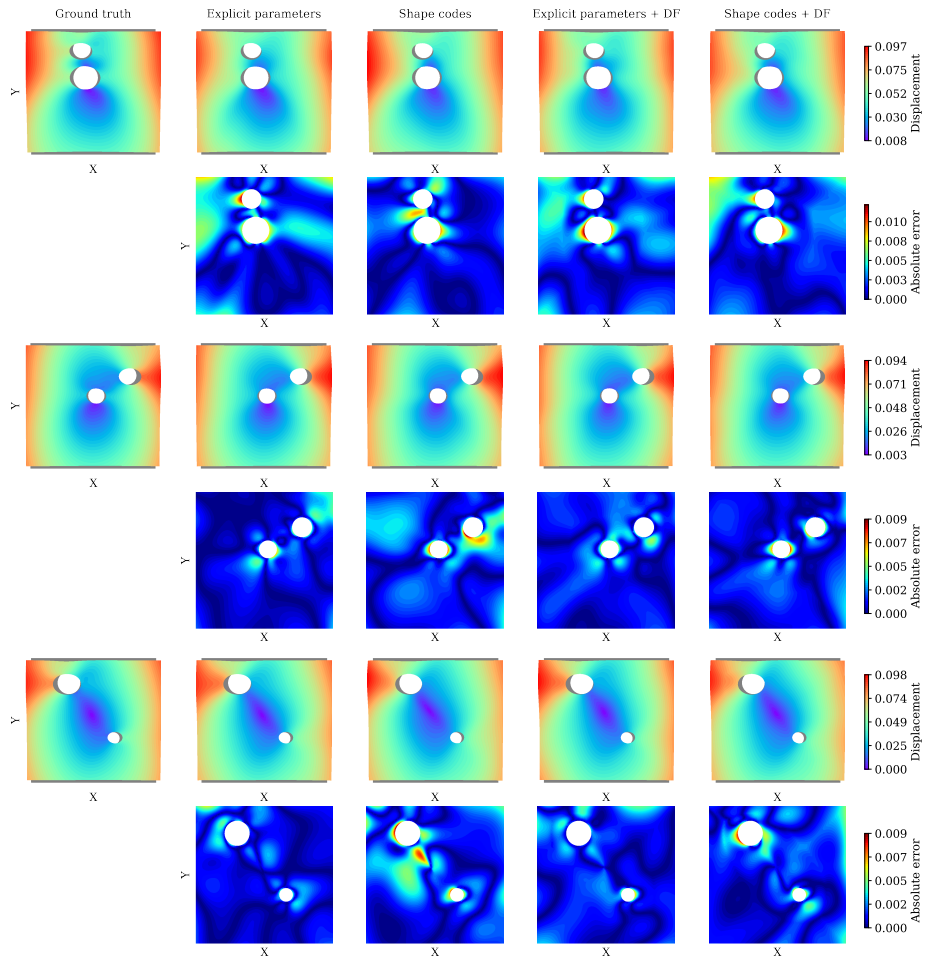
Figure 17: Test Case 2a: comparison of predictions from different input structures. From top to bottom displays three samples in the test dataset.

Additionally, the statistical results from multiple experiments suggest that training with shape codes offers slightly greater stability to the random initialization of trainable parameters compared to training with explicit parameters on the test dataset. In Fig. 17 we show the displacement fields predicted by different input structures.

4.2.2 Test Case 2b: variable topology

We now consider the case of topological changes. The training setting is similar to Test Case 2a, while the latent dimension is changed to 8 due to the additional complexity related to the number of holes on the elastic plate. Figure 18 shows examples of the reconstructed shape of the holes and predicted displacement. The reconstructions of displacement align consistently with the ground truth.
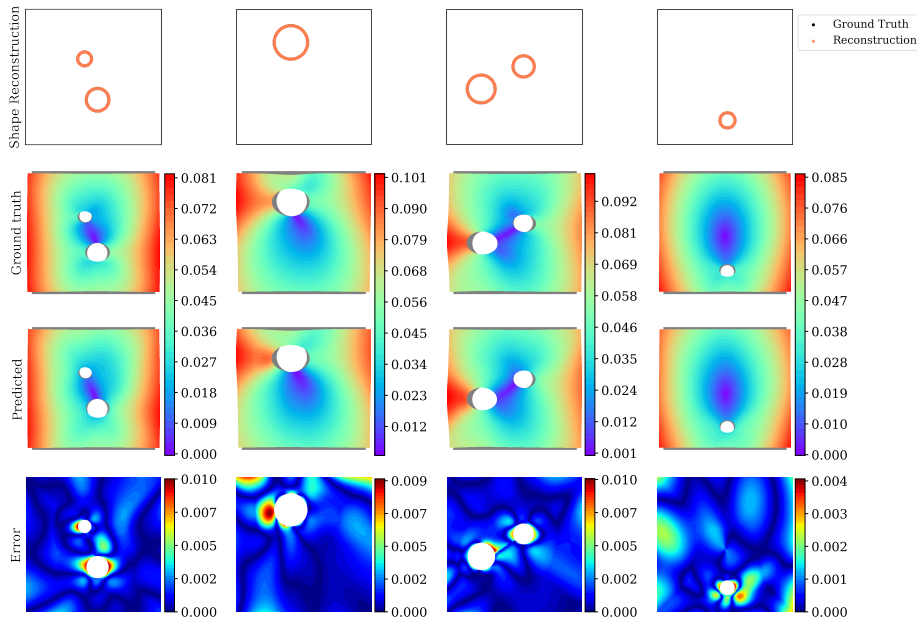
Figure 18: Test Case 2b: comparison of SDF-USM-Net reconstruction on four numerical solutions from the test set.

Impact of input features   We further compare the relative L_2 error in displacement between using shape codes and explicit parameters in Fig. 19a. The results indicate that the model's predictions can achieve a similar level of accuracy to that observed in Test Case 2a even in the presence of topology changes.

Impact of unbalanced dataset   We evaluate SDF-USM-Net on both balanced and unbalanced datasets. In the balanced dataset, 50% of the samples contain one hole, while the remaining 50% contain two holes. Conversely, in the unbalanced dataset, approximately 30% of the samples contain two holes, and the remaining 70% contain one hole. The model is evaluated on the same test set, which contains balanced data.

From the results presented in Fig. 19b, it is observed that the model achieves almost the same accuracy for both datasets which suggests that our model also demonstrates robustness to the unbalanced distribution of data.

5   Discussion

We introduced a new method, termed SDF-USM-Net and built upon USM-Net [27], which integrates DeepSDF to learn the solution manifold of PDEs in a data-driven and geometry-universal manner.

A first NN, termed $\mathcal{NN}_{\mathrm{SDF}}$, automatically learns a compact representation of shapes in a latent space, without the need for explicit encoder construction. The model merely requires pairs of coordinates and corresponding SDF values to characterize a given shape in a low-dimensional latent space, simplifying database construction (and reducing its dimensionality) compared to methods
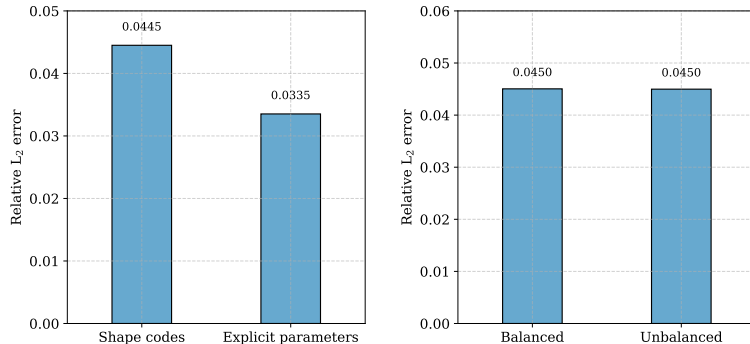
Figure 19: Test Case 2b: (a) Relative L_2 error of different input structures. (b) Relative L_2 error of balanced and unbalanced dataset.

like auto-encoders. By implicitly representing geometries in the Euclidean space, the model is capable of extracting shape codes for geometries not included in the training set, demonstrating very good generalization capabilities. This renders our model an exceptionally flexible technique capable of addressing a broad spectrum of real-world applications, encompassing both shape and topology changes.

Unlike standard approaches that reconstruct a high-dimensional discretization of the output, such as evaluating the solution on the nodes of a computational mesh or of a Cartesian grid, our approach is meshless in nature. The reconstruction NN, termed $\mathcal{NN}_{\text{Phys}}$, is queried independently for each point in space. The meshless nature of SDF-USM-Nets, coupled with automatic feature extraction in low-dimensional latent spaces, makes it possible to learn the solution map underlying PDE models with lightweight architectures and with low training costs.

Specifically, our approach demonstrates the potential for clinical applications, as illustrated in Test Case 1. Given the complexity of the geometry, artificially defined geometrical landmarks may lack accuracy in representing it. In contrast, we notice that shape codes accurately encode the geometrical features of the domain, enhancing representational accuracy. With an 11-dimensional latent space, in fact, the model achieves a prediction error for the flow field of 6.01%, showing an improvement over predictions based on more than 20 geometrical landmarks, where the error is 7.03%. Additionally, we have presented tests of our model on geometry databases undergoing topological changes, as shown in Test Case 2. With the inclusion of an additional input layer based on Fourier features, the model achieves a better representation capacity, achieving a Chamfer distance of 8.69×10^{-5}. Our results indicate that the proposed method, even when relying on learned shape codes rather than the knowledge of explicit geometric parameters, can achieve a level of accuracy nearly equivalent to the ideal scenario with known parameters, with errors of 5.33% and 4.97%, respectively. Considering that an explicit parametrization may be not available in many practical problems, SDF-USM-Net exhibits a remarkable performance.

A current limitation of this work is that it does not consider the case of time-dependent inputs and variable physical parameters, which will be the subject of future research. However, we emphasize that the proposed method demonstrates

flexibility, automation, and accuracy within its current scope.

A   Comparison study

In this section, we list the detailed experiment setups and parameter searching for test cases in Sec. 4.

A.1   Test Case 1: coronary bifurcation

We explored the NN architecture design space and optimization strategy through the following experiments.

Network depth and width   We tested network depths ranging from 3 to 5 layers and widths from 32 to 64 neurons for training $\mathcal{NN}_{\mathrm{SDF}}$. The results of these experiments, conducted on 300 training samples, are presented in Table 1. In the following experiments for Test Case 1, the depth is set to 4 with width set to 32.

Depth×width	Chamfer distance
3×32	1.075×10^{-3}
4×32	3.117×10^{-4}
5×32	6.298×10^{-4}
4×48	8.014×10^{-4}
4×64	6.307×10^{-4}

Table 1: Network architecture vs Chamfer distance on the test dataset.

Optimization strategy   With the network depth and width determined, we trained $\mathcal{NN}_{\mathrm{SDF}}$ using both a one-step optimization strategy (Adam only) and a two-step optimization strategy (Adam followed by L-BFGS), as shown in Table 2. The NN in this experiment was trained on 500 samples. Our findings indicate that the two-step optimization strategy achieves higher accuracy compared to the one-step optimization strategy.

k	Adam	Adam+L-BFGS
15	2.479×10^{-4}	1.389×10^{-4}
25	1.743×10^{-4}	1.388×10^{-4}

Table 2: Chamfer distance on the test dataset after one-step vs two-step optimization.

Activation function   We tested several types of activation functions, and the results on the test dataset are summarized in Table 3. Our findings indicate that $\mathcal{NN}_{\mathrm{SDF}}$ using the GeLU activation function outperforms other activation functions in the current application scenario.

Activation function	ReLU	Tanh	GeLU
Chamfer distance	1.389×10^{-4}	2.144×10^{-4}	1.091×10^{-4}

Table 3: Performance comparison of activation functions

A.2 Test Case 2: elastic plate

The discrete and topological changes in the geometry present significant challenges to the training of $\mathcal{NN}_{\mathrm{SDF}}$. In this section, we evaluate the effects of centralization and Fourier feature mapping on improving the model's performance.

Centralization The centralization method, as defined in Sec. 2.2.3, is designed to improve the model's robustness to geometric variations by aligning the centroid of each shape. This alignment ensures that the network can focus on the intrinsic properties of the shapes rather than their arbitrary positions. The results on the test dataset presented in Table 4 demonstrate a notable improvement in accuracy when the centralization method is applied, indicating its effectiveness in handling the discrete changes in geometry.

Coordinate input	No centralization	Centralization
Chamfer distance	1.430×10^{-4}	8.763×10^{-5}

Table 4: Performance improvement of centralization

Fourier feature mapping Fourier feature mapping, described in Sec. 2.2.4, enhances the network's ability to capture low-frequency variations in the input data. The results on the test dataset shown in Table 5 illustrate the performance gains achieved through this technique.

Input mapping	No mapping	m=16	m=8	m=4
Chamfer distance	7.44×10^{-4}	2.33×10^{-4}	8.72×10^{-5}	8.69×10^{-5}

Table 5: Performance improvement of Fourier feature mapping

Acknowledgement

References

[1] Siamak Ardekani, Robert G Weiss, Albert C Lardo, Richard T George, Joao AC Lima, Katherine C Wu, Michael I Miller, Raimond L Winslow, and Laurent Younes. Computational method for identifying and quantifying shape features of human left ventricular remodeling. *Annals of biomedical engineering*, 37:1043–1054, 2009.

[2] Igor A. Baratta, Joseph P. Dean, Jørgen S. Dokken, Michal Habera, Jack S. Hale, Chris N. Richardson, Marie E. Rognes, Matthew W. Scroggs, Nathan Sime, and Garth N. Wells. DOLFINx: the next generation FEniCS problem solving environment. *preprint*, 2023.

[3] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, sep 1975.

[4] Steven L Brunton, Bernd R Noack, and Petros Koumoutsakos. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52:477–508, 2020.

[5] Martin D Buhmann. Radial basis functions. *Acta numerica*, 9:1–38, 2000.

[6] M. Guo and J. S. Hesthaven. Data-driven reduced order modeling for time-dependent problems. *Computer methods in applied mechanics and engineering*, 345:75–99, 2019.

[7] Xiaoxiao Guo, Wei Li, and Francesco Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 481–490, 2016.

[8] Tobias Heimann and Hans-Peter Meinzer. Statistical shape models for 3d medical image segmentation: a review. *Medical image analysis*, 13(4):543–563, 2009.

[9] Jan S Hesthaven, Gianluigi Rozza, and Benjamin Stamm. *Certified reduced basis methods for parametrized partial differential equations*, volume 590. Springer, 2016.

[10] Thomas JR Hughes, John A Cottrell, and Yuri Bazilevs. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer methods in applied mechanics and engineering*, 194(39-41):4135–4195, 2005.

[11] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[12] Henry J Lamousin and NN Waggenspack. Nurbs-based free-form deformations. *IEEE Computer Graphics and Applications*, 14(6):59–65, 1994.

[13] Toni Lassila and Gianluigi Rozza. Parametric free-form shape design with pde models and reduced basis method. *Computer Methods in Applied Mechanics and Engineering*, 199(23-24):1583–1592, 2010.

[14] Kookjin Lee and Kevin T Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

[15] Stefano Longobardi, Alexandre Lewalle, Sam Coveney, Ivar Sjaastad, Emil KS Espe, William E Louch, Cynthia J Musante, Anna Sher, and Steven A Niederer. Predicting left ventricular contractile function via gaussian process emulation in aortic-banded rats. *Philosophical Transactions of the Royal Society A*, 378(2173):20190334, 2020.

[16] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. 1998.

[17] Andrea Manzoni, Alfio Quarteroni, and Gianluigi Rozza. Model reduction techniques for fast blood flow simulation in parametrized geometries. *International journal for numerical methods in biomedical engineering*, 28(6-7):604–625, 2012.

[18] Romit Maulik, Bethany Lusch, and Prasanna Balaprakash. Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. *Physics of Fluids*, 33(3):037106, 2021.

[19] Jan Oldenburg, Finja Borowski, Alper Öner, Klaus-Peter Schmitz, and Michael Stiehm. Geometry aware physics informed neural network surrogate for solving navier–stokes equation (gapinn). *Advanced Modeling and Simulation in Engineering Sciences*, 9(1):8, 2022.

[20] Vivek Oommen, Khemraj Shukla, Somdatta Goswami, Remi Dingreville, and George Em Karniadakis. Learning two-phase microstructure evolution using neural operators and autoencoder architectures. *arXiv preprint arXiv:2204.07230*, 2022.

[21] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.

[22] Luca Pegolotti, Martin R Pfaller, Natalia L Rubio, Ke Ding, Rita Brugarolas Brufau, Eric Darve, and Alison L Marsden. Learning reduced-order models for cardiovascular simulations with graph neural networks. *Computers in Biology and Medicine*, 168:107676, 2024.

[23] Benjamin Peherstorfer and Karen Willcox. Dynamic data-driven reduced-order models. *Computer Methods in Applied Mechanics and Engineering*, 291:21–41, 2015.

[24] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[25] Alfio Quarteroni, Andrea Manzoni, and Federico Negri. *Reduced basis methods for partial differential equations: an introduction*, volume 92. Springer, 2015.

[26] F. Regazzoni, L. Dedè, and A. Quarteroni. Machine learning for fast and reliable solution of time-dependent differential equations. *Journal of Computational Physics*, 397:108852, 2019.

[27] Francesco Regazzoni, Stefano Pagani, and Alfio Quarteroni. Universal solution manifold networks (usm-nets): Non-intrusive mesh-free surrogate models for problems in variable domains. *Journal of Biomechanical Engineering*, 144(12):121004, 2022.

[28] Francesco Regazzoni, Stefano Pagani, Matteo Salvador, Luca Dede, and Alfio Quarteroni. Learning the intrinsic dynamics of spatio-temporal processes through latent dynamics networks. *Nature Communications*, 15(1):1834, 2024.

[29] Cristobal Rodero, Marina Strocchi, Maciej Marciniak, Stefano Longobardi, John Whitaker, Mark D O'Neill, Karli Gillette, Christoph Augustin, Gernot Plank, Edward J Vigmond, et al. Linking statistical shape models and simulated function in the healthy adult human heart. *PLoS computational biology*, 17(4):e1008851, 2021.

[30] Sami Romdhani, Shaogang Gong, Alexandra Psarrou, et al. A multi-view nonlinear active shape model using kernel pca. In *BMVC*, volume 10, pages 483–492, 1999.

[31] Laura M Sangalli, Piercesare Secchi, Simone Vantini, and Alessandro Veneziani. A case study in exploratory functional data analysis: geometrical features of the internal carotid artery. *Journal of the American Statistical Association*, 104(485):37–48, 2009.

[32] Thomas W Sederberg and Scott R Parry. Free-form deformation of solid geometric models. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 151–160, 1986.

[33] L. Sirovich. Turbulence and the dynamics of coherent structures. i. coherent structures. *Quarterly of Spplied Mathematics*, 45(3):561–571, 1987.

[34] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains, 2020.

[35] Pantelis R Vlachas, Georgios Arampatzis, Caroline Uhler, and Petros Koumoutsakos. Multiscale simulations of complex systems by learning their effective dynamics. *Nature Machine Intelligence*, 4(4):359–366, 2022.

[36] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

[37] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. In *Computer Graphics Forum*, volume 41, pages 641–676. Wiley Online Library, 2022.

MOX Technical Reports, last issues

Dipartimento di Matematica

Politecnico di Milano, Via Bonardi 9 - 20133 Milano (Italy)

Galliani, G.; Secchi, P.; Ieva, F.

Estimation of dynamic Origin–Destination matrices in a railway transportation network integrating ticket sales and passenger count

69/2024 Galliani, G.; Secchi, P.; Ieva, F.

Estimation of dynamic Origin–Destination matrices in a railway transportation network integrating ticket sales and passenger count data

70/2024 Panzeri, L.; Fumagalli, A.; Longoni, L.; Papini, M.; Diego, A.

Sensitivity analysis with a 3D mixed-dimensional code for DC geoelectrical investigations of landfills: synthetic tests

68/2024 Gambarini, M.; Ciaramella, G.; Miglio, E.

A gradient flow approach for combined layout-control design of wave energy parks

65/2024 Possenti, L.; Vitullo, P.; Cicchetti, A.; Zunino, P.; Rancati, T.

Modeling Hypoxia Induced Radiation Resistance and the Impact of Radiation Sources

64/2024 Cavazzutti, M.; Arnone, E.; Ferraccioli, F.; Galimberti, C.; Finos, L.; Sangalli, L.M.

Sign-Flip inference for spatial regression with differential regularization

63/2024 Vitullo, P.; Franco, N.R.; Zunino, P.

Deep learning enhanced cost-aware multi-fidelity uncertainty quantification of a computational model for radiotherapy

62/2024 Roknian, A.A.; Scotti, A.; Fumagalli, A.

Free convection in fractured porous media: a numerical study

60/2024 Temellini, E.; Ferro, N.; Stabile, G.; Delgado Avila, E.; Chacon Rebollo, T.; Perotto, S.

Space - time mesh adaptation for the VMS - Smagorinsky modeling of high Reynolds number flows

61/2024 Speroni, G.; Ferro, N.

A novel metric - based mesh adaptation algorithm for 3D periodic domains