A non-parametric regularization framework for surrogate learning based on deep encoding

RESEARCH ARTICLE | JANUARY 26 2026

A non-parametric regularization framework for surrogate learning based on deep encoding

Laura Balasso ; Alessandro Alaia

Physics of Fluids 38, 016121 (2026)
https://doi.org/10.1063/5.0312954

Physics of Fluids ARTICLE pubs.aip.org/aip/pof

A non-parametric regularization framework for surrogate learning based on deep encoding

Cite as: Phys. Fluids **38**, 016121 (2026); doi: 10.1063/5.0312954
Submitted: 19 November 2025 · Accepted: 31 December 2025 ·
Published Online: 26 January 2026

Laura Balasso[1,2,a] and Alessandro Alaia[2,b]

AFFILIATIONS

[1]Politecnico di Torino, Torino, Italy
[2]Optimad Engineering, Torino, Italy

[a]Author to whom correspondence should be addressed: laura.balasso@optimad.it
[b]Electronic mail: alessandro.alaia@optimad.it

ABSTRACT

Many engineering applications rely on numerical methods to simulate complex physical systems, which often entail prohibitive computational costs. Additionally, problems involving parametrized domains require the generation of a computational mesh for each configuration, which is an expensive and error-prone process. Surrogates based on proper orthogonal decomposition can alleviate this burden providing rapid predictions of partial differential equations (PDEs) solutions, but they rely on explicit geometric parametrizations and shared reference meshes, requirements rarely satisfied in practice. In this paper, we present a machine-learning-based workflow for predicting PDEs solutions and derived quantities in the context of geometric parametrization, without explicit knowledge of the geometric parameters or a common discretization of the geometries. The workflow is composed of two encoding blocks that independently encode the geometries and the associated outputs, and a mapping block that learns the relationship between the two encodings. The main contribution of this work is the introduction of a regularization method for INR-based encoders, designed to preserve the structure of the encoded spaces in the learned latent space. The workflow was tested on two problems. The prediction of one-dimensional signals (radial and thrust force) on a vertical axis turbine blade with varying thickness and curvature, and the prediction of the two-dimensional pressure coefficient fields associated with geometric variations of the Royal Aircraft Establishment airfoil 2822 (RAE2822). The experimental results demonstrate that the proposed regularization substantially enhances the overall predictive accuracy, reducing the prediction error of unseen configurations by one order of magnitude compared to more standard approaches.

I. INTRODUCTION

A. Motivations

Many engineering problems encountered in the industrial practice involve the simulation of complex physical systems, which are usually modeled by partial differential equations (PDEs). PDEs are typically used to model spatiotemporal phenomena, such as fluid flow and heat transfer, and play a pivotal role in the analysis and optimization of real-world processes. In most cases, PDEs are solved numerically using discretization-based methods, since their analytical solution is not available. However, achieving accurate numerical approximations often requires fine-grained spatial and temporal resolutions, leading to prohibitive computational cost. Furthermore, in problems involving geometric variations of the computational domain, an additional challenge stems from the fact that each configuration requires the generation of a new computational mesh. Although significant advancements have been made, automatic meshing remains a costly and error-prone process, particularly when dealing with the complex geometries typical of real-world applications.

Such computational burden renders high-fidelity simulations unfeasible in real time and many-query applications, such as shape optimization, interactive design, and uncertainty quantification. These challenges motivate the development of efficient surrogate models, which can learn from high-fidelity simulations and are able to rapidly predict PDEs solutions for new geometric configurations or varying parameters (e.g., boundary conditions and physical parameters). Among the most classical methods, proper orthogonal decomposition (POD)[1,2] combined with radial basis functions (RBF) interpolation[3,4] or Gaussian processes,[5–7] is particularly attractive because it offers an interpretable and computationally inexpensive approach. However, the linear nature of POD limits its effectiveness to a relatively narrow

class of problems. Moreover, in the context of geometric variation, POD-based methods typically compute a modal decomposition of the PDE solution and predict the coefficients starting from the geometric parameters of the problem. This is a huge limitation in industrial applications, since the knowledge of such parameters is not granted. In real-world scenarios, it is desirable to train the surrogates exploiting all the available simulation data, even when the user-defined parametrization is not readily available. These data may include geometries generated at different times, using heterogeneous methods and parametrization techniques, which may no longer be accessible or documented. In addition, POD is discretization-dependent, i.e., it relies on a reference mesh shared across all the training geometries. This requirement is enforced because assembling the so-called *snapshot matrix* requires that all the computational meshes have the same cardinality and numbering, and therefore, it is difficult to satisfy in practice especially in the case of significant geometric variations in the training dataset. Interpolation on a common *template* mesh is possible by using registration techniques, which are inherently error-prone and difficult to fully automate. These techniques often introduce artifacts and localized errors, especially when the geometries differ substantially. Such (localized) registration errors inject noise into the training data, ultimately degrading prediction accuracy and increasing training complexity. In order to circumvent these limitations, a flexible approach is required, capable of approximating complex, non-linear relationships without relying on a common discretization or any explicit knowledge of the underlying parameterization of the geometry.

B. Related work

In recent times, advances in the field of deep learning (DL) have enabled the development of more sophisticated surrogates, capable of approximating highly non-linear relationships with a flexible handling of geometric variations. Among these models, convolutional neural networks (CNN)[8] have been largely employed to predict PDEs solutions on 2D domains with variable shape, treating both the input domain and the target solution as image-like data. CNNs have demonstrated their effectiveness in various applications, including the prediction of pressure field around airfoils,[9,10] stress field,[11,12] and heat transfer.[13] Among the convolutional architectures, U-Net[14] model, initially developed for biomedical image segmentation, has been successfully employed in solid mechanics for predicting the stress field in a structure[15] and in fluid dynamics for the prediction of both 2D[16] and 3D[17] flow fields. Other works in the current literature make use of generative adversarial networks (GANs)[18] for the prediction of more realistic 2D fields[12,19–22] and applied to structural optimization problems.[23] Despite their success, grid-based architectures are intrinsically limited by the uniform resolution of the (image-like) input data. In order to capture fine-grained details in the target solution, the cells of the Cartesian grid should be as large as the smallest cell in the computational mesh, leading to a large grid size. However, convolutional networks tend to perform poorly with aggressive (de)convolutions, so a large input resolution would require a deeper architecture and result in a significantly larger model. Conversely, computationally affordable grid sizes induce an interpolation error that becomes significant in the presence of sharp gradients or discontinuities, complex geometries with curved boundaries and multi-scale phenomena. To overcome this limitation, geometric deep learning[24,25] techniques have been leveraged to handle directly unstructured data. Graph neural networks (GNNs) have been broadly adopted for the prediction of 2D[26,27] and 3D[28,29] PDEs solution directly on structured[30] and unstructured[31] grids. Despite their flexibility, GNNs are strongly dependent on the graph topology and suffer from high computational cost. Moreover, the irregular data structure limits the scalability and parallelization of GNN-based models. Implicit neural representations (INRs) constitute an appealing alternative to mesh-based approaches. INRs operate on point-cloud data, modeling continuous signals as neural networks that map spatial or temporal coordinates directly to the corresponding signal value. These models can also represent 2D or 3D scenes through their radiance fields[32] by learning their occupancy function[33] or the signed distance function (SDF).[34] Unlike discrete representations, such as grids or meshes, INRs provide a continuous, resolution-independent data representation with limited memory consumption. INRs can be extended to learn entire families of functions by conditioning the network on a latent code associated with each instance. This code, concatenated to the spatial coordinates, identifies the object or signal to which a point belongs, allowing the network to capture both the spatial mapping and a compact latent representation of the object. For instance, DeepSDF model[34] learns a latent representation for 3D shapes through the SDF, while[33] learns a family of occupancy functions. INRs have been recently employed for the prediction of PDEs solutions. In Ref. 35, an unsupervised INR is trained against a physics-informed loss function for the prediction of one-dimensional (1D) and two-dimensional (2D) parametric PDEs solutions. In Ref. 36, INRs are employed to perform mesh-free topology optimization to find the optimal material distribution. An end-to-end framework for the prediction of 2D and 3D flow fields though INRs is presented in Ref. 37. Here, implicit representations are learned for both the geometry (through its SDF) and the flow field and are subsequently mapped by a multi-layer perceptron (MLP). Similarly to INRs, physics-informed neural networks (PINNs)[38] are mesh-free models that predict continuous signals directly on spatial or temporal coordinates. PINNs are designed to approximate solutions to PDEs[39,40] and to solve inverse problems[41–44] such as estimating model parameters from limited data. Instead of relying only on data supervision, PINNs embed the governing physical laws into the learning process by penalizing deviations from the equations and boundary conditions in the loss function. Despite their conceptual appeal, PINNs often exhibit poor convergence and limited accuracy when trained over complex domains. Moreover, the loss term must be explicitly formulated for each specific problem or intrusively coupled with the underlying solver. These requirements make PINNs unsuitable in purely data-driven contexts or when relying on commercial black-boxed solvers.

C. Our contribution

In this work, we build upon the workflow introduced in Ref. 37 for predicting the solution of a target PDE formulated on a parametrized domain. Specifically, by leveraging INRs, the method learns latent representations of both the geometries and the associated solutions and maps the two latent codes using a feed-forward NN. This approach does not require explicit knowledge of the geometric parameters, as the latent parametrization is learned directly from the data. Therefore, the workflow can be trained using simulations computed on geometries generated with different methods and parametrizations. Thanks to the continuous representation provided by INRs, the model is not tied to any specific discretization and can predict the solution at

arbitrary spatial or temporal locations, without resolution limitations. The workflow described above is adopted from Ref. 37 and serves as the baseline of this work. Our contribution focuses instead on the regularization of the latent spaces learned by INRs, which is introduced in the following.

We propose a novel regularization framework for learning latent representations via INRs. Models like DeepSDF[34] learn a latent space that lacks a meaningful notion of distance, as distances in the latent space are not necessarily consistent with those in the original geometric or solution space. Such irregularity renders the mapping between the two latent spaces highly non-linear and can deteriorate the overall prediction accuracy. In order to address this issue, we present two regularization methods that preserve the geometric (or solution) distances in the latent space through a modified loss function. The first method introduces a penalization term that forces the latent codes to maintain a distance from a fixed reference sample that is as close as possible to the distance in the original space. The second method extends and improves the first one by preserving the pair-wise distances between all the possible couples of training samples.

The workflow is tested on two fluid dynamics problems involving 2D airfoils with geometric variations. The first problem pertains to the prediction of forces acting on a blade of a vertical axis wind turbine. The second concerns the prediction of the entire 2D pressure coefficient field around the transonic Royal Aircraft Establishment airfoil 2822 (RAE2822). Both experiments demonstrate that the proposed regularization methods improve the representation error of the geometries and the solutions in their respective latent spaces and substantially enhance the prediction accuracy of the mapping between the two spaces.

D. Paper structure

The remainder of this paper is organized as follows: In Sec. II, we present in detail the workflow that is built upon the related work on INRs. In Sec. III, we present the novel regularization methods for the latent representation of geometries and PDEs solutions. In Sec. IV, we describe the datasets used to test our workflow. Finally, in Sec. V, we report the results of our numerical experiments. Finally, in Sec. VI, conclusions and future perspectives are discussed.

II. BASELINE PREDICTION WORKFLOW

The aim of this work is to define a workflow for predicting a physical quantity (such as a signal or a scalar field) computed from the solution of a PDE defined on a parametrized domain. In general, the physical field $u(\mathbf{x};\mu)$ satisfies a PDE of the following form:

$$\mathcal{L}_\mu(u(\mathbf{x};\mu,\mathbf{p})) = f(\mathbf{x};\mu), \quad \mathbf{x}\in\Omega(\mathbf{p}), \tag{1}$$

where $\mathcal{L}_\mu$ is a differential operator depending on a set of parameters μ, f is the source term, and $\Omega(\mathbf{p})$ denotes the parametrized domain. As mentioned in the introduction, our goal is to predict the solution without direct knowledge of the domain parametrization $\mathbf{p}$, and without relying on a common mesh across all the geometries in the dataset. To this end, we adopt a representation-learning approach that first learns a *latent* representation for both the input geometries and the output solutions, and then the mapping between the latent space of the geometries and the latent space of the corresponding PDE solutions. At inference time, a new geometry is first encoded. Next, its latent code is mapped to the encoding of the corresponding solution, which is finally decoded to obtain the target quantity for the given geometry.

In this section, we describe the machine-learning-based workflow introduced in Ref. 37, which we adopt as the baseline for our approach. The workflow is composed of three main building blocks: geometric encoding, output encoding, and mapping. In the geometric encoding block, a shape parametrization is learned from the set of training geometries. Analogously, the output quantity of interest (i.e., a signal or a scalar field computed from the solution of the PDE) is encoded in a fixed-size vector in the output encoding block. In the mapping block, a fully connected feed-forward NN learns the mapping between the latent space of the input geometries and the latent space that encodes the solution of the target PDE. The machine learning (ML) models are trained in a supervised fashion on a dataset composed of variable-shape geometries each associated with the solution of the target PDE.

A. Geometric encoding

In the geometric encoding block, a deep encoding model is trained to learn the latent space that represents the training geometries. The latent representation is learned directly from the training data, without any explicit knowledge of the user-defined geometric parametrization. In order to achieve that, our approach relies only on an implicit description of the training geometries in terms of signed distance function (SDF). The SDF is computed for each geometry on a point cloud, which can differ for each training geometry. In this way, we can by-pass the expensive and error-prone meshing phase, which is difficult to robustly automate for complex geometries. The SDF associated with a given closed manifold $\Gamma = \partial\Omega \subset \mathbb{R}^n$ $(n = 2, 3)$ is a continuous function $SDF : \mathbb{R}^n \to \mathbb{R}$ that maps any spatial point in $\mathbb{R}^n$ to the distance from the closest point on Γ. In this work, we adopt the convention that a negative signed distance function indicates a point inside the manifold, that is,

$$SDF(\mathbf{x}) = \begin{cases} \text{dist}(\mathbf{x},\Gamma), & \mathbf{x}\in\Omega, \\ -\text{dist}(\mathbf{x},\Gamma), & \mathbf{x}\notin\Omega, \\ 0 & \mathbf{x}\in\partial\Omega. \end{cases} \tag{2}$$

The manifold, Γ, is implicitly represented by the zero level set of the SDF, i.e., $\Gamma = \{\mathbf{x}\in\mathbb{R}^n | SDF(\mathbf{x}) = 0\}$.

In order to train the encoding model for a set of N shapes, M points are samples in the physical space for each shape, both inside and outside the surface Γ, and the SDF is computed on each point,

$$X_i = \{\mathbf{x}_j, s_j\} : s_j = SDF_i(\mathbf{x}_j), \quad j = 1,\dots,M, \quad i = 1,\dots,N. \tag{3}$$

The DeepSDF model[34] was employed to learn a fixed-size encoding from point-cloud SDF data. The architecture, represented in Fig. 1, is a fully connected NN $f_\theta(\mathbf{z},\mathbf{x}) \approx SDF(\mathbf{x})$, where $\mathbf{x}\in\mathbb{R}^n$ is a spatial point, $\mathbf{z}\in\mathbb{R}^d$ is an encoding of the shape, and θ represents the vector of trainable weights. Every latent code $\mathbf{z}_i$ is associated with the training shape represented by the point cloud X_i. The output of the NN is the SDF value computed at $\mathbf{x}$ for the shape encoded by $\mathbf{z}_i$.

Note that counter-intuitively, the latent code of a shape is an input of the model; hence, this architecture is referred to as *auto-decoder* to stress the difference with respect to the standard encoder–decoder structure. For simplicity and consistency in terminology, in this work we refer to this model as *encoder* or encoding model.

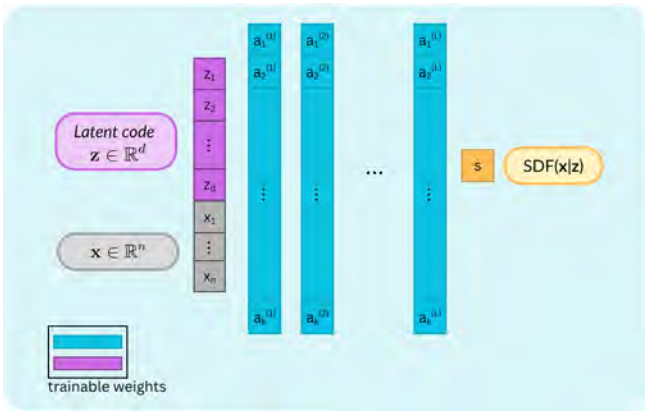

FIG. 1. DeepSDF auto-decoder architecture.

The formulation for DeepSDF model is derived using a probabilistic approach. The posterior probability distribution over the shape encoding $\mathbf{z}_i$, given the SDF sample X_i, can be written as

$$p_\theta(\mathbf{z}_i|X_i) = p(\mathbf{z}_i) \prod_{(\mathbf{x_j}, \mathbf{s_j}) \in X_i} p_\theta(s_j|\mathbf{z}_i; \mathbf{x}_j), \tag{4}$$

where θ can also be interpreted as a parametrization for the SDF likelihood and $p(\mathbf{z}_i)$ is the prior distribution over the latent codes, which is assumed to be a multivariate Gaussian with spherical covariance $\sigma^2 I$. SDF likelihood is expressed by the feed-forward NN $f_\theta(\mathbf{z}_i, x_i)$ and takes the following form:

$$p_\theta(s_j|\mathbf{z}_i; \mathbf{x}_j) = \exp(-\mathcal{L}(f_\theta(\mathbf{z}_i, \mathbf{x}_j), s_j)), \tag{5}$$

where $\mathcal{L}(\tilde{s}_j - s_j)$ is a loss function that quantifies the deviation of the predicted SDF from the true value. During training, the joint log posterior over the training samples is maximized with respect to the shape encoding and the NN weights, i.e.,

$$\underset{\theta, \{\mathbf{z}_i\}_{i=1}^N}{\operatorname{argmin}} \sum_{i=1}^{N} \left(\sum_{j=1}^{M} \mathcal{L}(f_\theta(\mathbf{z}_i, \mathbf{x}_j), s_j) + \frac{1}{\sigma^2} ||\mathbf{z}_i||_2^2 \right). \tag{6}$$

At inference, the latent code for a new shape is optimized via maximum *a posteriori* (MAP) estimation with the learned weights, θ, fixed,

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmin}} \sum_{(\mathbf{x}_j, s_j) \in X} \mathcal{L}(f_\theta(\mathbf{z}, \mathbf{x}_j), s_j) + \frac{1}{\sigma^2} ||\mathbf{z}||_2^2. \tag{7}$$

This inference step requires to solve an additional optimization problem for each new shape, meaning that predictions are not performed at real time. However, latent code optimization is significantly faster than the NN training, since only the latent code is optimized and requires few iterations to converge (see Appendix A). Note that the formulation of this encoding model does not require the spatial distribution of the points $\mathbf{x}_j$ to be uniform, nor for the point cloud to be constant across for all the training shape. In this work, points were sampled more densely near the surface of the training geometries to capture the zero level set in more detail (as suggested in Ref. 34).

B. Output encoding

The encoder presented in Sec. II A can be generalized to encode any target quantity including the solution of a target PDE provided that the quantity of interest is distributed in a spatial or temporal space.

To this end, the architecture presented in Sec. II A is adapted by replacing the SDF value with the target scalar field/signals that must be encoded. In this case, the independent variable, $\mathbf{x}$, represents the spatial (or temporal) coordinate on which the output depends.

The SDF is continuous and differentiable almost everywhere and has a unit gradient where differentiable. The only exception is represented by the set of the points located on the medial axis. Thanks to these property, neural networks can generally approximate the SDF with a good accuracy. Conversely, the target function can be highly non-linear and exhibit abrupt gradient changes, discontinuities, or incorporate different frequencies. Consequently, a fully connected network may struggle to approximate such functions, due to the "spectral bias."[45] To overcome this limitation, the $\mathbf{x}$ coordinate is transformed using a positional Fourier embedding characterized by the frequency spectrum,

$$\gamma(\mathbf{x}) = \left[\ldots, \cos(2^l \pi \mathbf{x}), \sin(2^l \pi \mathbf{x}), \ldots\right], \quad l = 0, \ldots, L, \tag{8}$$

where L is an hyper-parameter, which requires proper tuning.

With this modification, the input layer of the encoding block becomes $[\mathbf{z}, \gamma(\mathbf{x})]$. As demonstrated in Ref. 46, the projection onto this particular higher-dimensional embedding allows to correct the spectral bias and capture the different frequencies, which are embedded in the function to be encoded.

C. Mapping

After the training of the encoding models, all the geometries and the associated outputs are encoded in a fixed-size vector. Thus, a fully connected network can be trained to learn the mapping between the two latent spaces. A schematic overview of the complete training workflow is shown in Fig. 2. At inference time, the output field corresponding to a new geometry can be predicted starting from the associated SDF sample. First, the geometric latent code is estimated by solving the optimization problem in Eq. (7). Then, the latent code is passed to the mapping block that predicts the latent code of the output field. Finally, the predicted latent vector is fed to the output encoder, together with the spatial or temporal coordinates on which the output is defined. This last step decodes the latent vector and provides a prediction of the desired output.

D. General considerations

In Secs. II A–II C, we presented a workflow that predicts fields of interest for geometries with variable shape by learning a latent representation of both the input geometry and the solution. From now on, we will refer to this workflow as *complete workflow* (CWF). If the output of interest is a scalar or is defined on a one-dimensional space, the workflow can be further simplified by removing the output encoding block and modifying the mapping block to directly predict the output.

In the case of a scalar output, it is sufficient to train a NN that maps directly the geometric encoding to the target scalar. If the target output is a *one-dimensional* signal, we can train a model that takes as input the learned geometric latent code alongside with the value of the

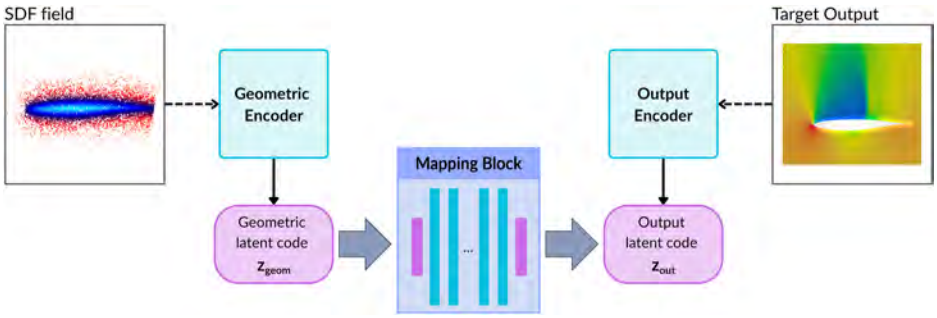

FIG. 2. Diagram of the complete workflow.

spatial coordinate and provides point-wise predictions of the output signal. The architecture of this modified mapping block is analogous to the encoding model in Fig. 1 with the only difference being the training strategy. In this mapping block, the input latent code **z** is not unknown, but is obtained from the pre-trained geometric encoder, and a Fourier embedding can also be applied to the 1D coordinate to improve signal approximation. Hereafter, we will refer to this approach as *simplified workflow* (SWF). A diagram of the SWF is depicted in Fig. 3.

CWF and SWF allow to predict scalar outputs and fields associated with shape-varying geometries without relying on an explicit parametrization of the training shapes. Moreover, since both workflows are designed to generate point-wise predictions starting from the SDF computed on a point-cloud, they are not bounded to a specific discretization of the training geometries.

Another advantage is represented by the modularity. In the presented approach, each specialized block is trained independently and can be reused to solve a different task without retraining *from scratch*. For instance, if a simplified workflow is trained to predict a scalar output given a dataset of geometries, the pre-trained geometric encoder can be used for other predictions task, such as predicting surface or volume fields. In such cases, only the output encoder and the intermediate NN must be trained, while the trained geometric encoder can be re-used *off-the-shelf*.

III. DISTANCE PRESERVING REGULARIZATION FOR DEEP GEOMETRIC ENCODING

In Sec. II, we presented a machine-learning-based strategy for the prediction of PDEs solutions in problems involving geometrical variations. The approach is based on three independently trained ML-based blocks: two encoding blocks that learn a latent representation for the geometries and the output of interest, and a mapping block that learns the relation between the two latent spaces. In this section, we focus on the encoding blocks and we propose a method to enforce regularity in the learned latent space. We argue that an irregular latent space makes the training of the mapping block more difficult and cause poor prediction accuracy. On the contrary, introducing a regularization during the training of the geometric encoder improves significantly the prediction accuracy both qualitatively and quantitatively, as will be demonstrated later in our numerical experiments.

DeepSDF model learns a latent space representation that encodes the geometric variability of the training shapes. However, it is not guaranteed that two points (i.e., geometric shapes) that are close in the geometric space remain close in the latent space. In general, we would expect that the latent codes associated with similar shapes (from a geometric point of view) are close also in the latent space. Nevertheless, this property is not formally imposed as a constraint during the training of the encoder or the optimization of the latent representation. In other words, a good correlation between geometric distances and latent codes distances is not guaranteed.

As an example, consider Fig. 4, which shows an example of this behavior for the vertical axis turbine dataset (described in more detail in Sec. IV A). For each shape in the training dataset, both the SDF distances and latent codes distances are computed with respect to all the other shapes. The correlation plot between these two quantities is depicted in Fig. 4 and shows that these two measures are strongly uncorrelated, that is, two shapes that are close in the SDF space may be far apart in the latent space and vice versa.

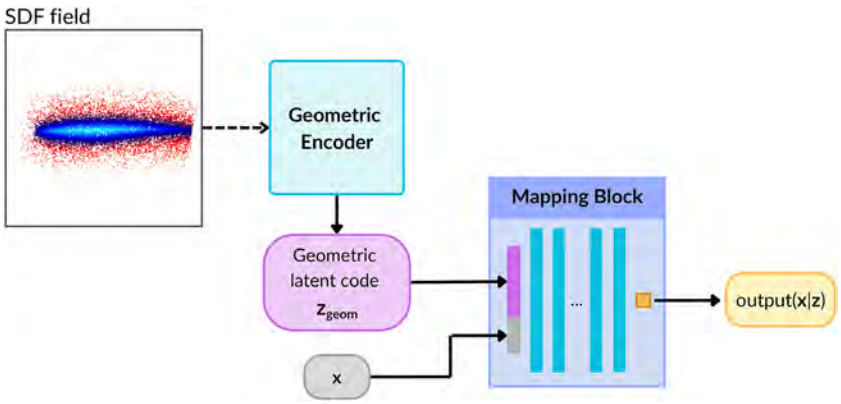

FIG. 3. Diagram of the simplified workflow.

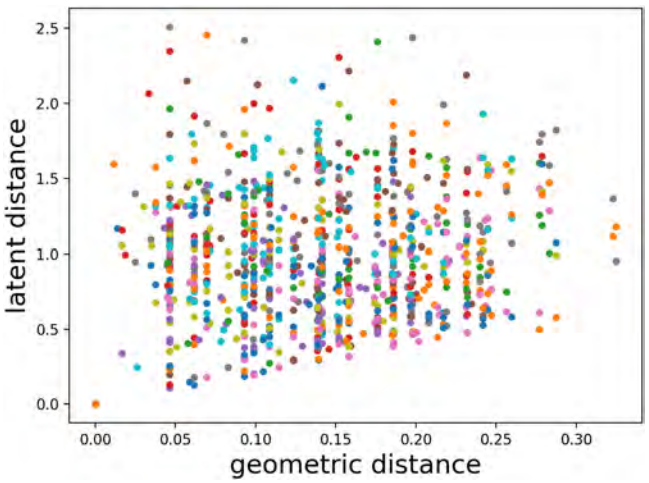

FIG. 4. Correlation plot of geometric distance and latent codes distance. Each color represents the distance of training samples from a fixed reference sample.

Since the latent space represents the input space for the mapping block, irregularities in the geometric encoding may induce strong nonlinearities in the mapping from latent codes to outputs. Of course, since NNs are universal approximators, the mapping can still be learned, but at a cost of a more complex model that would require more training data and might exhibit poor generalization capabilities. This limitation is not problematic only for regression tasks. It can potentially affect any approach based on geometric encoding, including optimization workflows that employ surrogates built in the geometric latent space.

To address this issue, we introduce a regularized version of DeepSDF model that preserves the geometric distances in the latent space through a penalization term in the loss function. First, we need to define the geometric distance, which measures the dissimilarity between two shapes. If the geometric parameters are available, this quantity could be expressed in terms of Euclidean distance in the parameters space. However, in a general setting, this information is not readily available. In order to preserve the parameter-agnostic nature of the present approach, we exploit the SDF representation of the geometric shapes, which is already used in the original loss. In order to compute the distances between two shapes given in terms of SDF, SDF samples need to be defined on the same point cloud. If this is not the case, the training samples in Eq. (3) can be interpolated on a common set of spatial points. At this point, we can compute the geometric distance between training shapes in terms of Euclidean distance between the associated SDF samples,

$$d_{SDF}^{(i,j)} = \frac{||SDF_i - SDF_j||_2}{\max_{i,j\in\{1,\dots,N_{train}\}} ||SDF_i - SDF_j||_2}, \tag{9}$$

where $SDF_i \in \mathbb{R}^M$ is the SDF of the ith training shape computed or interpolated on the common point cloud of size M. The distance in Eq. (9) is normalized by the maximum distance in the training dataset, so that the measure is independent from the geometric scale of the training geometries.

A. Fixed reference distance regularization

With the distance between signed distance function defined in Eq. (9), we can now define a penalization term in the loss function that drives the distance between latent codes toward the normalized geometric distance. The idea is to select a reference sample that can be considered as a centroid for the training dataset and assume that its latent code is fixed and corresponds to the origin of the latent space. During the training, the additional penalization forces the latent codes of the other shapes to maintain a distance from the origin that is as close as possible to the normalized distance from the reference sample in the geometric space. This method will hereafter be referred to as fixed reference distance (FRD) regularization. The centroid is selected with using a min–max strategy, i.e., as the training sample that has the smallest maximum distance from the other samples in the dataset,

$$i^* = \underset{i}{\text{argmin}}(\max_j d_{\text{SDF}}^{(i,j)}). \tag{10}$$

In the original DeepSDF formulation, NN weights and latent codes are optimized concurrently against the same loss function $\mathcal{L}_{\text{DeepSDF}}$, which includes a regularization term penalizing large latent vector norms. In the proposed regularized version, an additional penalization term forces the distance from the centroid latent code (i.e., the origin) to be equal to the normalized SDF distance from the centroid. The penalization term is defined as follows:

$$\mathcal{L}_{\text{FRD}} = \frac{\sum_{i=1,i\neq i^*}^{N_{train}} \left| d_{\text{SDF}}^{(i,i^*)} - d_{latent}^{(i,i^*)} \right|}{N_{train} - 1} = \frac{\sum_{i=1,i\neq i^*}^{N_{train}} |d_{\text{SDF}}^{(i,i^*)} - ||\mathbf{z}_i||_2|}{N_{train} - 1}. \tag{11}$$

Since the latent code of the reference sample is fixed at the origin, the distance in the latent space between the ith sample and the reference can be expressed as the norm of the ith latent code. The updated loss function is thus defined as follows:

$$\mathcal{L} = \mathcal{L}_{\text{DeepSDF}} + \lambda \mathcal{L}_{\text{FRD}}. \tag{12}$$

In order to implement the above-mentioned regularization, the vector $\mathbf{d} = [d_{\text{SDF}}^{(i,i^*)}]_{i=1}^{N_{train}}$ of SDF distances from the reference is

ALGORITHM 1. Fixed reference distance regularization.

Input:
- batch latent codes $Z_{\text{batch}} \in \mathbb{R}^{B\times d}$ at the current training iteration,
- batch indices $I_{\text{batch}} \in \{1,\dots,N\}^B$,
- reference index i^*,
- normalized SDF distances from reference $\mathbf{d} \in \mathbb{R}^{N_{train}-1}$.

Output: Regularization loss $\mathcal{L}_{\text{FRD}}$

Initialize latent_dist $\in \mathbb{R}^B$
Initialize sdf_dist $\in \mathbb{R}^B$
for $i = 1 \rightarrow B$ **do**
    latent_dist$[i] \leftarrow \| Z_{\text{batch}}[i]\|_2$
    sdf_dist$[i] \leftarrow \mathbf{d}[I_{\text{batch}}[i]]$
end for
$\mathcal{L}_{\text{FRD}} \leftarrow$ mean(|latent_dist − sdf_dist|)

pre-computed, so that during training only the norm of the current latent codes needs to be evaluated. The steps followed at each training iteration for a batch of size B and latent codes of size d, are described in Algorithm 1. Note that the latent code of the reference geometry is fixed. Thus, the reference sample is excluded from the training dataset.

The regularization presented in this section is a simple and straightforward method to ensure that the learned latent codes preserve the geometric distance from a given reference geometry. However, for each training sample, the distance is maintained only with respect to the centroid of the training dataset; hence, the "coherence" of the input geometric space is still not fully guaranteed. During the training, the latent codes are constrained to preserve a fixed distance from the reference sample. As a result, latent vectors are free to move on the surface of a d-dimensional sphere with radius equal to the geometric distance from the reference geometry, where d is the dimension of the latent space. The most immediate consequence, is that while we expect a good correlation between the distances $[d_{\text{SDF}}^{(i,i^*)}]_{i=1}^{N_{train}}$ and $[d_{\text{latent}}^{(i,i^*)}]_{i=1}^{N_{train}}$, the same cannot be told for an arbitrary couple of training geometries.

B. Pair-wise distance regularization

In order to overcome the limitations of the regularization described in Sec. III A, one possible solution is to constrain the distances between all the couples of latent codes $(\mathbf{z}_i, \mathbf{z}_j)$. In other words, constraining the relative position of all the training geometries simultaneously allows the latent space to *inherit the structure* of the input geometric space. This approach is referred to as pair-wise distances (PWD) regularization.

To achieve this, the penalization term in Eq. (11) is modified so that all the training samples are considered as reference simultaneously. The PWD penalty term is defined as follows:

$$\mathcal{L}_{\text{PWD}} = \frac{\sum_{i=1}^{N_{\text{train}}} \sum_{j=1, j\neq i}^{N_{\text{train}}} |d_{\text{SDF}}^{(i,j)} - d_{\text{latent}}^{(i,j)}|}{N_{\text{train}}(N_{\text{train}} - 1)}, \tag{13}$$

where the latent distance is the Euclidean norm between vectors, i.e.,

$$d_{\text{latent}}^{(i,j)} = ||\mathbf{z}_i - \mathbf{z}_j||_2. \tag{14}$$

This modified penalty term can be also be interpreted as an average of the term in Eq. (11) computed over all possible choices of the reference geometry, i^*.

The advantage of the above-mentioned regularization with respect to the FRD regularization is twofold. First, the reference geometry is not removed from the dataset. In general, the removal of a single training sample has negligible impact on the model accuracy. However, when the dataset is small and training data points are evenly distributed, the removal of a single sample may create a gap in the training domain. Second, the choice of the reference sample is not trivial and has a non-negligible impact on the model performance. For instance, in Sec. III A, we proposed a selection strategy based on a *min–max* criterion. However, a different selection strategy might have a significant effect on the trained model.

From an implementation perspective, the $N \times N$ matrix $D = [d_{\text{SDF}}^{(i,j)}]$ of pair-wise geometric distances between training samples is computed during the offline stage. The scalability of this distance matrix computation with respect to the dataset size is analyzed in E. The steps followed at each training iteration for computing the penalization term are described in Algorithm 2. Latent codes distance is computed between the codes in the current batch and all the training codes at the previous iteration, namely, the references. Note that a mask is applied to ensure that the distance of the sample from itself is not considered, otherwise each latent code would be biased toward its own value at the previous iteration.

ALGORITHM 2. Pair-wise distance regularization.

Input:
- batch latent codes $Z_{\text{batch}} \in \mathbb{R}^{B\times d}$ at the current training iteration,
- batch indices $I_{\text{batch}} \in \{1, \ldots, N\}^B$,
- all latent codes $Z_{\text{all}} \in \mathbb{R}^{N\times d}$ at the previous training iteration,
- normalized SDF distance matrix $D \in \mathbb{R}^{N\times N}$

Output: Regularization loss $\mathcal{L}_{\text{PWD}}$

Initialize latent_dist $\in \mathbb{R}^{\text{B}\times\text{N}}$
Initialize sdf_dist $\in \mathbb{R}^{\text{B}\times\text{N}}$
for $i = 1 \rightarrow B$ **do**
  for $j = 1 \rightarrow N$ **do**
    latent_dist$[i,j] \leftarrow\!|| Z_{\text{batch}}[i] - Z_{\text{all}}[j]||_2$
    sdf_dist$[i,j] \leftarrow D[I_{\text{batch}}[i], j]$
  end for
end for
Construct mask $M \in \{0,1\}^{B\times N}$ with $M[i,j] = 0$ if $I_{\text{batch}}[i] = j$, else 1
latent_dist $\leftarrow$ latent_dist $\odot M$
sdf_dist $\leftarrow$ sdf_dist $\odot M$
$\mathcal{L}_{\text{PWD}} \leftarrow$ mean(|latent_dist − sdf_dist|)

By replacing the FRD with the PWD regularization, the global loss becomes

$$\mathcal{L} = \mathcal{L}_{\text{DeepSDF}} + \lambda \mathcal{L}_{\text{PWD}}. \tag{15}$$

Regularization in Eq. (13) is expected to translate geometric proximity into latent space proximity provided that the hyper-parameter λ is large enough. Note that the penalty term is applied only to the training loss, while the inference step is performed as described in Sec. II A. The reason for this choice is twofold. First, learning a regular latent space during the training step is in general sufficient to improve inference accuracy. Second, computing the distance penalty at inference time would require the access to the training SDF samples. This is practically unfeasible in many real-world scenarios, or if a pre-trained model is used *off-the-shelf*.

C. Output distance regularization

In the complete workflow presented in Sec. II, the target solution is encoded in the output encoding block. The output encoder and the geometric encoder are characterized by the same architecture and training strategy and differ only in the target value to be encoded. Therefore, also in the output encoder, it is not guaranteed that two

solutions that are close in solution space are also close in the learned latent space. In order to improve the coherence within the latent space that encodes the solutions, we can apply PWD regularization. To achieve this, it is necessary to define a measure for the distance between two solutions as follows:

$$d_{out}^{(i,j)} = \frac{\| \mathbf{u}_i - \mathbf{u}_j \|_2}{\max_{i,j\in\{1,\dots,N_{train}\}} ||\mathbf{u}_i - \mathbf{u}_j||_2}, \tag{16}$$

where $\mathbf{u}_i$ is the solution associated with the ith geometry, computed on a set of points X_i, such as a pressure field around an airfoil. In order to compute this distance between all the pairs of solutions in the training dataset, the field $\mathbf{u}$ must be evaluated at the same spatial locations. However, when geometric variations are present, each solution is associated with a distinct physical domain. Consequently, points that lie outside one airfoil configuration may fall inside another for a different training or test geometry. Nevertheless, an approximation of this distance can still be computed. To this end, each solution field $\mathbf{u}$ is first interpolated onto a common reference set of spatial locations, sampled in the region where all the training solutions are defined. While this procedure neglects geometry-specific regions, it provides a sufficient approximation of the distance for the purpose of regularization. With the distance between solution defined in Eq. (16), we can compute the output regularization term as follows:

$$\mathcal{L}_{\text{PWD}} = \frac{\sum_{i=1}^{N_{\text{train}}} \sum_{j=1, j\neq i}^{N_{\text{train}}} |d_{\text{out}}^{(i,j)} - d_{\text{latent}}^{(i,j)}|}{N_{\text{train}}(N_{\text{train}} - 1)}, \tag{17}$$

where the latent distance is the Euclidean norm between vectors. The global loss function for the output encoder with PWD regularization becomes

$$\mathcal{L} = \mathcal{L}_{\text{DeepSDF}} + \lambda \mathcal{L}_{\text{PWD}}. \tag{18}$$

The above-mentioned regularization of the output encoder loss is expected to improve the coherence within the latent space representing the solutions and ensure that similar solutions are remain close in the latent space. As explained at the end of Sec. III B, the penalty term is applied only to the training loss and not in the optimization of validation codes.

In this work, we applied the regularization to the output encoding only in the RAE2822 experiment C, and we observed a small improvement in the prediction accuracy. For the wind turbine experiment B instead, we employed the simplified workflow, in which the output encoding is omitted and therefore the loss regularization cannot be applied.

IV. DATASETS

In this work, we consider two test cases to demonstrate the capabilities of the proposed framework. The first test case pertains to the prediction of the thrust and radial forces in a vertical axis wind turbine (VAWT), where the goal is to predict the behavior of both forces as a function of the rotation angle for a given turbine blade geometry. In the second test case, we apply our workflow for the prediction of the entire pressure field around a transonic RAE2822 airfoil. In this section, we describe the datasets used in both numerical experiments. Numerical results are presented in Sec. V.

A. Vertical axis wind turbine (VAWT)

The first experiment was conducted on a three-bladed VAWT with geometric variations. The VAWT dynamics was approximated with a 2D CFD solution on the domain illustrated in Fig. 5.

The blade's baseline geometry is a NACA0015 airfoil with chord of 420 mm, rotating at a distance $R = 1.4$ m from the rotational axis at a constant speed. Further details about the geometry can be found in Ref. 47. Since this work is focused on geometric variations, all the operational parameters were kept constant, specifically wind speed $U = 10$ m/s, Reynolds number $Re = 290\,000$, and a rotational speed $\omega = 14.29$ rad/s. The simulations were performed using Ansys Fluent[48] by solving the transient RANS equations coupled with the $k-\omega$ SST turbulence model. Blade rotation was represented through a cylindrical sliding mesh centered on the rotational axis of the VAWT. Time step used in the CFD simulations is defined based on the angular velocity, ω, in order to ensure a rotation of 1° per time step. In Sec. V B, we compare our numerical results with the results obtained with a multi-fidelity POD approach, which was trained on both high-fidelity simulation (performed on a mesh with 411k cells) and low-fidelity solutions (computed on a 107k cells computational mesh). For the training of our regularized DeepSDF workflow, we only employed high fidelity simulations. More details about the simulation setup can be found in Ref. 49.

To obtain geometric variations of the blade profile, the baseline airfoil was deformed by modifying both thickness and curvature. Since the profile is symmetric, thickness is modulated by multiplying the y coordinate by a scaling coefficient, c_{thick}. Curvature is controlled via a quadratic Bezier curve with three control points,

$$B(x) = (1-x^2)P_0 + 2(1-x)xP_1 + x^2P_2, \tag{19}$$

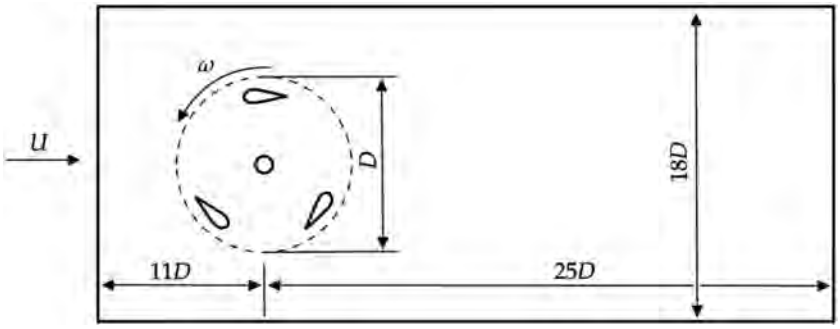

FIG. 5. Schematic depiction of the 2D computational domain used for the three-bladed Darrieus VAWT.

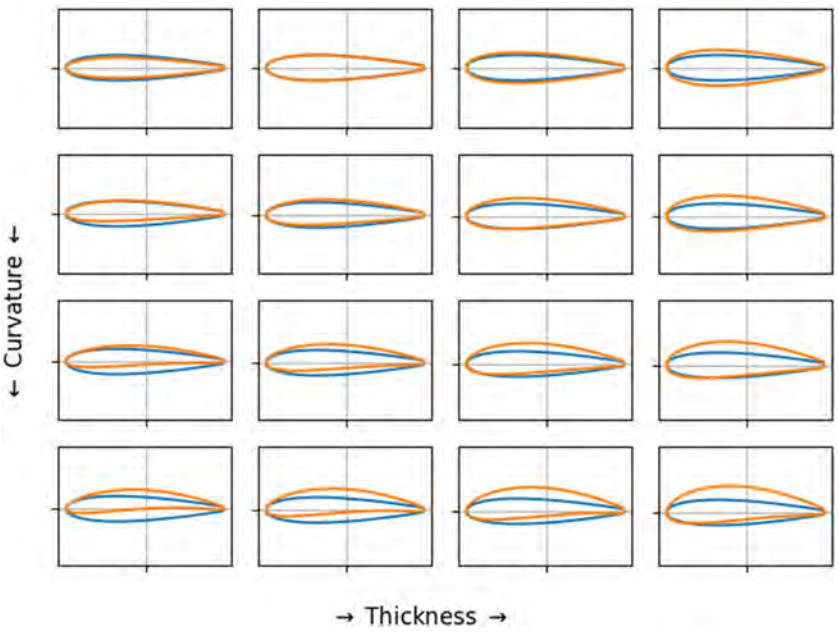

FIG. 6. Baseline NACA0015 (blue) and geometrical variations obtained by varying c_{thick} and c_{curv} (orange).

where P_0 is the leading edge, P_2 is the trailing edge, and P_1 is the point $(x_{maxThickness}, c_{curv}c)$, where c denotes the chord length. The dimensionless coefficient c_{curv} is used to modulate the curvature of the profile. A deformed profile can be obtained by multiplying the y coordinates of the base airfoil by $B(x)$. Thus, the upper, y_u, and lower, y_l, part of the profile are obtained as follows:

$$\begin{aligned} y_u(x) &= c_{thick}B(x)\bar{y}_u(x), \\ y_l(x) &= c_{thick}B(x)\bar{y}_l(x). \end{aligned} \tag{20}$$

In the previous equation, $\bar{y}_u(x)$ and $\bar{y}_l(x)$ represent the y-coordinate of the baseline NACA0015 airfoil. The dataset consists of 16 geometrical variations of the base NACA0015, obtained by sampling 16 points (c_{thick}, c_{curv}) in the range $[0.8, 1.4] \times [0.0, 0.05]$ with a four levels full factorial sampling. The 16 airfoil variations are displayed in Fig. 6.

Aerodynamics fields, such as the velocity field in Fig. 7, are obtained for each airfoil geometry via CFD simulation. The outputs of interest for this test case are the forces acting on one blade during a full revolution of the turbine, namely, the thrust force (F_T) and the radial force (F_R), for a rotation angle $\phi \in \{0, \ldots, 360\}$ (see Fig. 8).

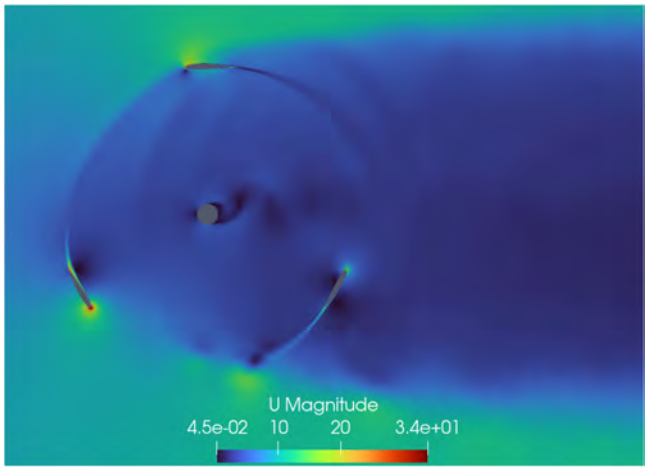

FIG. 7. Example of velocity field computed for a deformed airfoil geometry with thickness coefficient $c_{thick} = 0.8$ and curvature coefficient $c_{curv} = 0.016$.

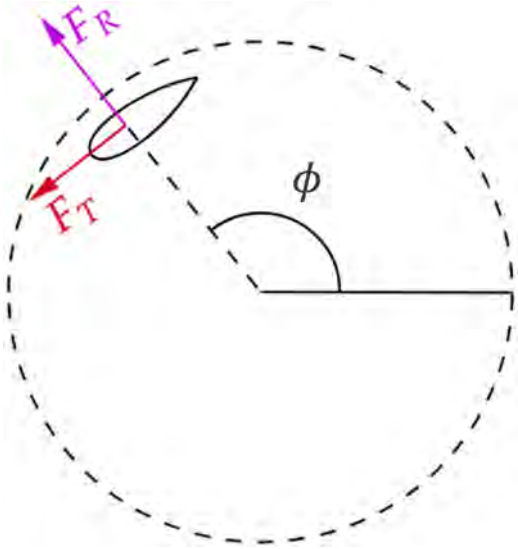

FIG. 8. Thrust force (F_T) and radial force (F_R) acting on a rotating turbine blade.

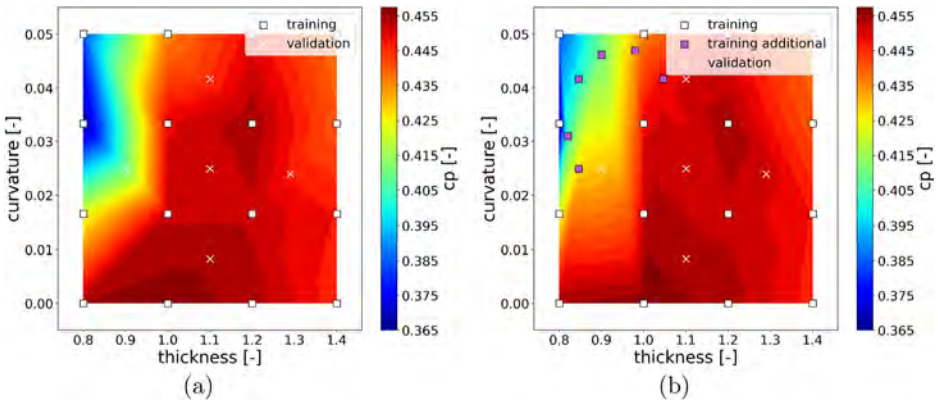

FIG. 9. Power coefficient as a function of the deformation parameters for the (a) initial and (b) complete dataset.

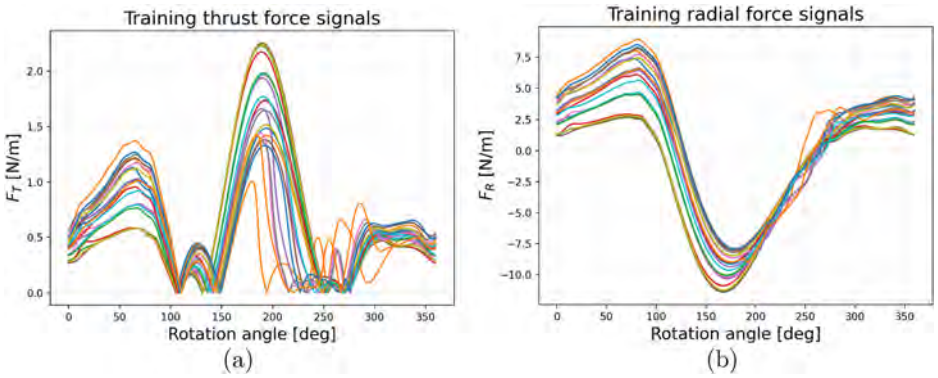

FIG. 10. (a) F_T and (b) F_R signals for each airfoil in the training dataset.

The resulting dataset is composed of 16 geometries, each associated with $F_T(\phi)$ and $F_R(\phi)$ with $\phi \in \{0, \ldots, 360\}$. The thrust component contributes to the power generation that can be measured through the power coefficient C_P and is computed as the ratio between the turbine power P_t and the wind power P_w as follows:

$$C_P = \frac{P_t}{P_w} = \frac{\omega R N_b \frac{1}{2\pi}, \int_0^{2\pi} F_T(\phi) d\phi}{\frac{1}{2}\rho U^3 3RH}. \tag{21}$$

Here, N_b is the number of blades, ρ is the air density, and H is the height of the wind turbine's swept area. This work will focus primarily on the thrust force component since F_T is the only contributor to the power coefficient (which is the quantity usually considered for optimization), while the radial component is only relevant from a structural point of view.

The C_P value computed from the CFD simulations is plotted in Fig. 9(a) for different values of the geometric parameters. As shown in Fig. 9(a), the region corresponding to airfoils characterized by small thickness and large curvature exhibits high values of the C_P gradient,

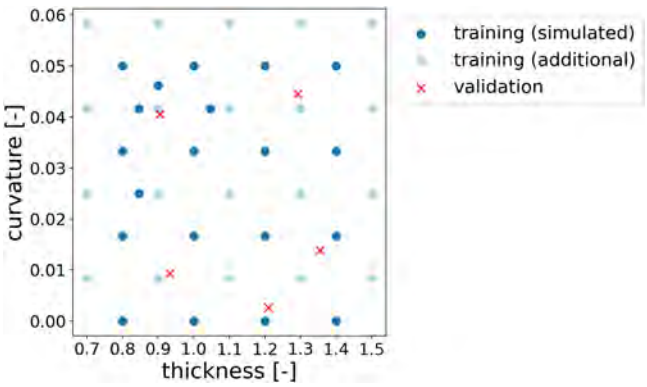

FIG. 11. Training dataset for the geometric encoder.

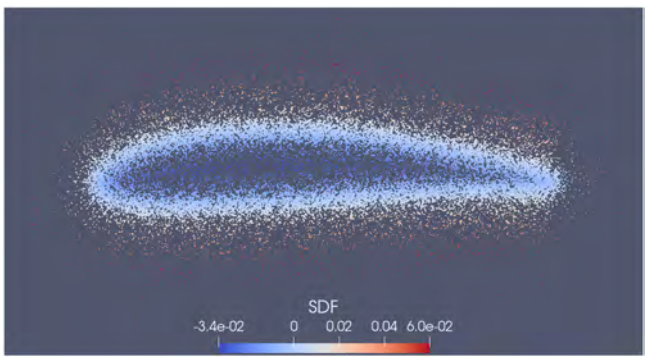

FIG. 12. Sample points where the SDF is computed for a training airfoil.

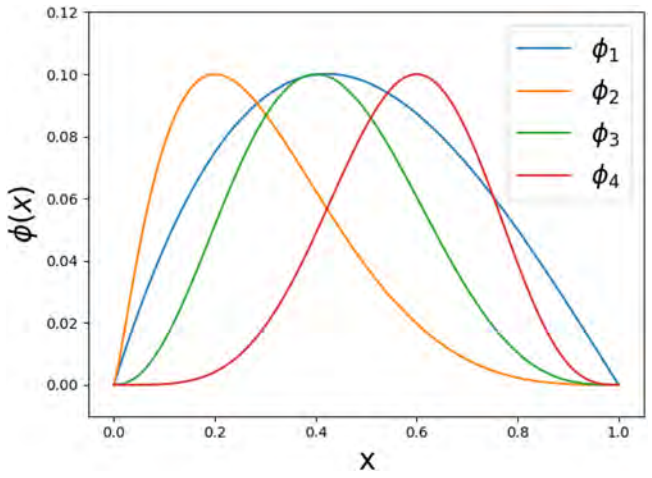

FIG. 13. RAE2822 deformation modes used to generate geometric variations of the base airfoil.

suggesting the need for a denser sampling. Therefore, six additional designs were sampled in the high-gradient region using a space filling strategy, for a total number of 22 data points. Additionally, five validation samples were selected in the center of the cells defined by the four levels full-factorial grid. The complete dataset is shown in Fig. 9(b), where the original 16 training data points are plotted as white square markers, the additional six data points sampled from the high-gradient region are shown as purple square markers, while validation data points are indicated by white cross. Thrust and radial forces for each airfoil in the training/validation dataset are shown in Fig. 10 as a function of the rotation angle.

In order to train the geometric encoder, a broader dataset of airfoils was used. The training dataset for the geometric encoder is composed of the 22 airfoils described earlier with the addition of 20 geometric variations generated from the base NACA0015, for a total of 42 airfoils. Five additional airfoils were generated by randomly sampling the parameters space for validation. The dataset used to train the geometric encoder is depicted in Fig. 11.

The SDF for each of the 42 training geometries was computed on a point cloud consisting of 35 000 points. For each geometry, the 35 000 points were selected on the airfoil curve and randomly divided in three sets. Points were scattered around the surface of the airfoil, by adding a random Gaussian perturbation with zero mean and a different variance for each of the three groups. An example of a point cloud sampled for a training geometry is shown in Fig. 12.

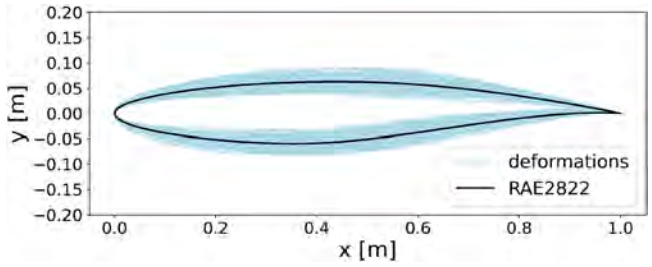

FIG. 14. Baseline RAE2822 and deformed airfoils in the training dataset.

B. Transonic RAE2822

In the second experiment, we consider a transonic RAE2822 airfoil. Compared to the previous case, the geometric variations are more nuanced, and the output of interest is the 2D pressure coefficient field around the airfoil. The baseline airfoil is the RAE2822, shown in Fig. 14. The reference airfoil has normalized chord length $c = 1.0[-]$ and maximum thickness of $0.12[-]$.

Geometric variations of the base airfoil were obtained by adding a linear combination of four deformation modes to both the lower and the upper surface of the baseline profile, as follows:

$$\begin{aligned} y_l(x) &= k(\bar{y}_l + \sum_{i=1}^{4} c_i^l \phi_i(x)), \\ y_u(x) &= k(\bar{y}_u + \sum_{i=1}^{4} c_i^u \phi_i(x)), \quad x \in [0,1], \end{aligned} \tag{22}$$

where $\bar{y}_l$ and $\bar{y}_u$ are the lower and upper curve of the baseline RAE2822 and k is a scaling coefficient that is computed in order to preserve the original maximum thickness. The four deformation functions are defined as follows (see Fig. 13):

$$\phi_1(x) = (0.5x^3 - 1.5x^2 + x)0.52, \tag{23}$$

$$\phi_2(x) = 0.1 \sin(\pi x^{0.431})^3, \tag{24}$$

$$\phi_3(x) = 0.1 \sin(\pi x^{0.757})^3, \tag{25}$$

$$\phi_4(x) = 0.1 \sin(\pi x^{0.357})^3. \tag{26}$$

The weights $c_1^l, \ldots, c_4^l, c_1^u, \ldots, c_4^u$ govern the deformation modes and can be interpreted as the geometric parameters of the problem. In order to generate the training dataset, 1000 sets of deformation coefficients were sampled with a Halton sequence in the eight-dimensional parameters space. Halton sequence is nested, which means the subset composed of the first N_{sub} training points (with $N_{sub} \ll N_{train}$) is still a Halton sequence. This allows to train the models with an incremental number of training points so that an error analysis can be performed at any level with the nested subsets still providing a good coverage of the parameter space. The following ranges for the deformation coefficients are considered:

$$\begin{aligned} c_1^l, c_1^u &\in [-2, 2], \\ c_2^l, c_2^u &\in [-0.5, 0.5], \\ c_3^l, c_3^u &\in [-1, 1], \\ c_4^l, c_4^u &\in [-1, 1]. \end{aligned} \tag{27}$$

Note that since c_1^l and c_1^u control the low frequency deformation modes, they can vary over a wider range without producing unrealistic shapes. Conversely, c_2^l and c_2^u modulate high frequency deformation modes and have a strong influence on the airfoil shape near the leading edge. Therefore, the range of variations for these parameters is tighter in order to produce realistic shapes. Figure 14 illustrates the 1000 training deformations applied to the baseline airfoil compared with the base RAE2822.

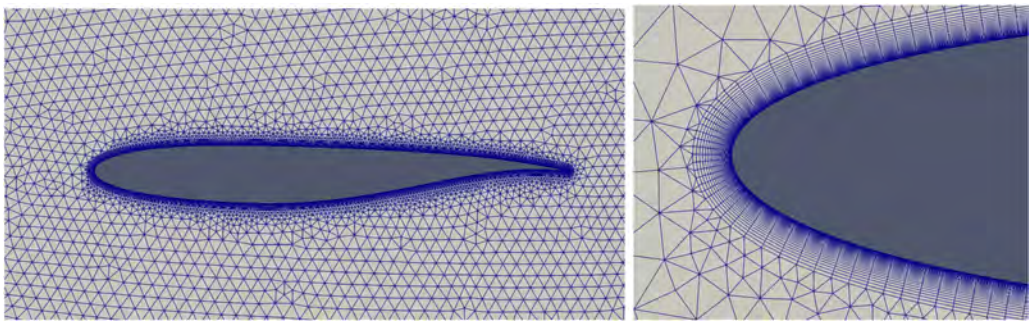

FIG. 15. Snapshots of the computational mesh for a geometric variation of the base RAE2922 in the training dataset.

The validation dataset was built by randomly sampling 300 additional airfoil variations within the convex hull of the training samples to avoid extrapolation. For each training and validation geometry, the SDF was computed on a point cloud consisting of 20 000 points sampled with the same strategy used for the VAWT dataset.

For each training and validation airfoil, the flow field was computed using the software SU2.[50] The angle of attack is $\alpha = 3.5°$, the Reynolds number $Re = 6\,500\,000$, the free-stream Mach number $Ma = 0.734$, and temperature $T = 16.65$ K. The computational domain is 40-times the aerodynamic chord. An example of computational mesh around a training geometry can be observed in Fig. 15. The training data were generated by considering a subset of the CFD computational domain consisting of a smaller rectangular region centered around the airfoil. Such a window represents our inference domain. Figure 16 shows the C_p field for one of the training airfoils. The pressure coefficient computed on the surface of the airfoil is shown in Fig. 17 for 40 different variations of the base RAE2822. The effect of the varying geometry is evident in Fig. 17, where both the position and strength of the shock on the suction side of the airfoil vary significantly.

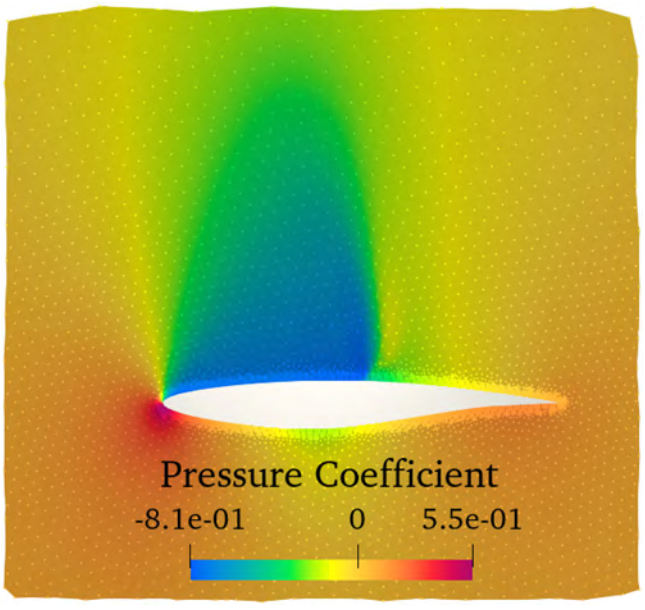

FIG. 16. Example of the pressure coefficient computed around a training airfoil and restricted to the inference domain.

V. RESULTS

In this section, we test the workflow presented in Sec. II on two problems involving 2D airfoils with geometric deformations. In the first test case, we apply the simplified workflow (SWF) to predict the forces acting on a blade of a vertical axis wind turbine for a full turbine revolution. In this test case, geometry variations are generated with two parameters and we predict 1D signals given a geometric variation of the turbine blade. We train the models on the dataset presented in Sec. IV A, which consists of 42 training geometries for the geometric encoding block and 22 airfoils for the mapping block. The second test case pertains to the prediction of the entire 2D pressure coefficient field on geometric variations of a RAE2822 airfoil controlled by eight deformation parameters. For this case, we employ the complete workflow (CWF), which includes the output encoding block.

For each test problem, we test the prediction workflow and we compare the effect of both the regularization methods introduced in Sec. III on the prediction accuracy. All experiments were implemented in PyTorch[51] v2.1. The training was performed on a single NVIDIA A10 GPU.

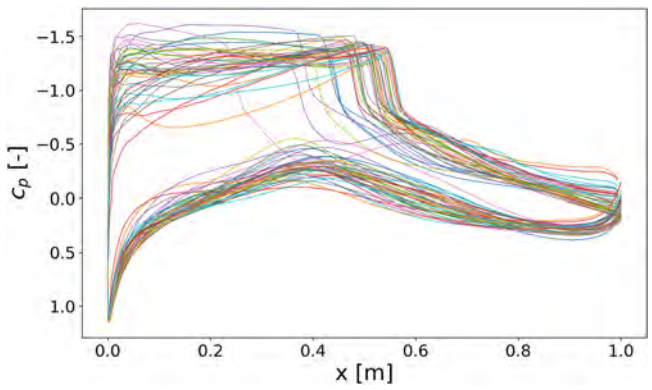

FIG. 17. Surface pressure coefficient for 40 geometrical variations of the base RAE2822 in the training dataset.

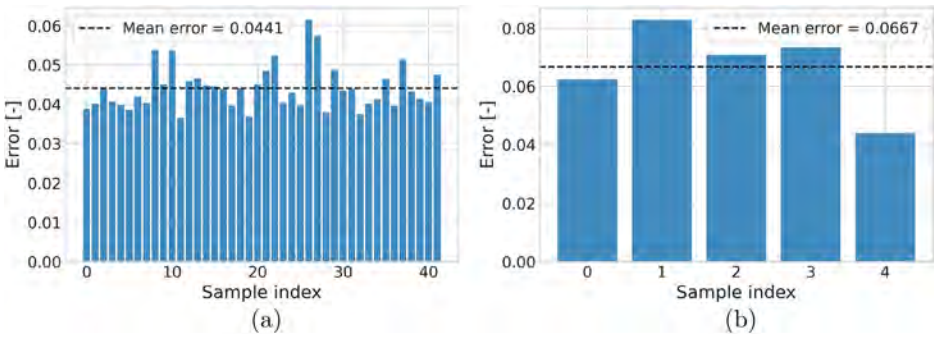

FIG. 18. Relative SDF prediction error on (a) training and (b) validation samples.

A. Models validation and error metrics

In both experiments, each block in the workflow is trained independently and from scratch.

The accuracy of the encoding blocks is evaluated in terms of SDF (or output) reconstruction, by comparing the predicted SDF(/output) to the ground-truth value. In order to assess the prediction accuracy for a training sample, the SDF is predicted from the latent codes learned during the training. For a validation geometry, first the latent code is computed by solving Eq. (7) (with encoder weights fixed), and then, the associated SDF is predicted and compared with the true value.

The mapping block accuracy, i.e., the overall accuracy of the workflow, is assessed by computing the prediction error directly in terms of the output of interest. More precisely, the geometric latent code is fed into the mapping block, which produces a prediction of the latent code for the target output. Finally, the predicted latent code is passed to the trained output encoder, and the output field is reconstructed and compared with the ground-truth value.

The performance is evaluated in relative terms, that is,

$$error_i = \frac{||\mathbf{y}_i - \hat{\mathbf{y}}_i||_2}{||\mathbf{y}_i||_2}, \tag{28}$$

where $\mathbf{y}_i$ is the ground-truth value and $\hat{\mathbf{y}}_i$ is the predicted value, i.e., the predicted SDF for the geometric encoder or the output field of interest for the output encoder and the mapping block.

Finally, given the error distribution on the training/validation dataset, we consider as outliers the error that deviates from the mean error by more than five standard deviations. For better visualization, outliers are omitted from the error histograms. However, outlier count and the highest outlier value are reported in the error tables. Outlier detection was not performed in VAWT experiment since the dataset is too small to reliably identify outliers.

B. VAWT forces prediction

In this experiment, we consider the vertical axis wind turbine (VAWT) dataset A and predict the thrust force F_T and radial force F_R acting on one blade during a full revolution of the turbine. Since the outputs are 1D signals, we use the simplified workflow that consists of the encoding block and the mapping block only (see Sec. II D).

In the encoding block, a DeepSDF model with four hidden layers of size 128, ReLU activations, and final tanh activation was trained for 2000 epochs. The learning rate was initialized at 5×10^{-4} and divided by 2 every 200 epochs with a piece-wise constant scheduler. For the latent codes, the same scheduling was used with a starting learning rate of 10^{-3}. The best results were obtained with a latent space of size 3.

As reported in Fig. 18, the trained model predicts training and validation SDF samples with a mean error of 3.7% and 6.7%, respectively. For a qualitative assessment, we compared the airfoil reconstructed from the predicted latent code with the ground-truth SDF. As shown in Fig. 19, even in the worst case, the reconstructed surface cannot be visually distinguished from the ground-truth geometry. This suggests that the learned latent space is, in fact, able to represent both training and validation airfoils.

The model used to map the geometric encoding to the output signals is a feed-forward NN with four hidden layers of size 256. The NN was trained to predict the forces acting on the blade for a given value of the rotation angle. More precisely, the NN takes as input the latent code representing the airfoil shape and the value of the rotation angle and outputs the corresponding scalar target, namely, the thrust force or the radial force. Two different models featuring the same architecture were trained separately for 2000 epochs with an initial learning rate of 3×10^{-4} and a stepwise constant scheduler that reduces the learning rate by half every 200 epochs.

For each geometric latent code, we predicted the force for each value of the rotation angle in $[0°, 360°]$. Then, we concatenated predictions over all angle values to reconstruct the target signals for a full turbine revolution. Figure 20 shows the error distribution on the

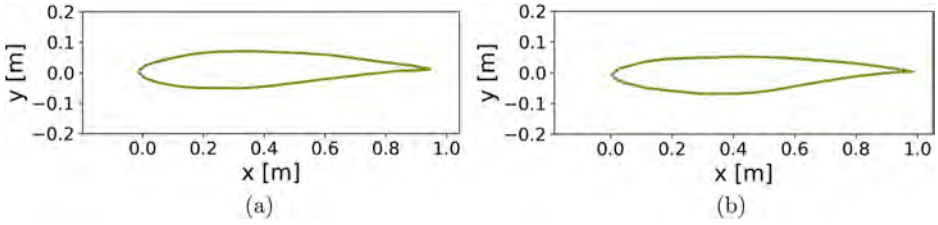

FIG. 19. Airfoil surface reconstruction of the (a) training and (b) validation cases corresponding to the largest error. The green line is used to show the ground-truth airfoil, while the encoded–decoded geometry is shown in orange.

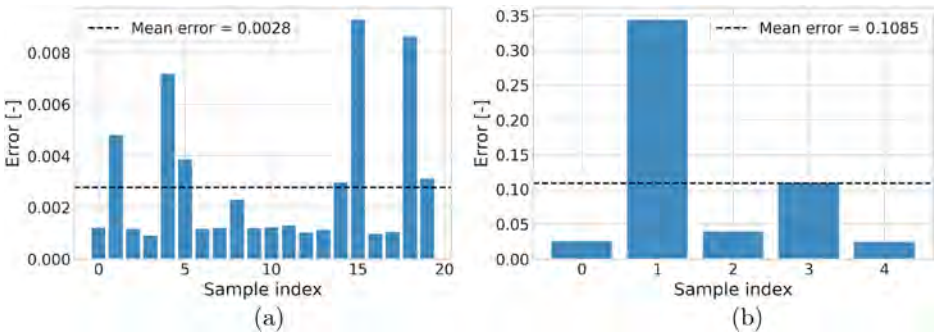

FIG. 20. F_T relative prediction error on (a) training and (b) validation samples.

training datasets. The average prediction error is 0.28% with a maximum error below 1%. However, the mean error on validation cases is two orders of magnitude higher, reaching approximately 35% for the worst prediction. The discrepancy between training and validation accuracy is more evident in Figs. 21 and 22 where the predicted signals are compared to the ground-truth for both a training and a validation case.

In order to quantify the benefit of the regularized encoding, we re-trained the DeepSDF model using the same datasets and hyperparameters. The only difference consists in the additional regularization term appearing in the loss function with a weight $\lambda = 0.1$. The weight λ was selected as the value yielding the lowest validation error. The results of this tuning procedure are reported in Appendix D. Figure 23 shows that the FRD regularization has a negligible effect on the correlation between the geometric distance and the latent distance. On the other hand, the method employing the PWD regularization improves substantially the correlation.

Another interesting effect of the PWD regularization can be observed from the principal component analysis (PCA) of the regularized latent codes. Although the best results were obtained with latent codes of size 3, Fig. 24 clearly shows that most of the variance (precisely 99.64%) is explained by the first two PCA components, implying that the latent space could be reduced to a two-dimensional subspace with negligible information loss. This result is consistent with the fact that the true parameters space is in fact bi-dimensional. A qualitative comparison between the true parameter space and the latent space is shown in Fig. 25, which shows the power coefficient plotted on the

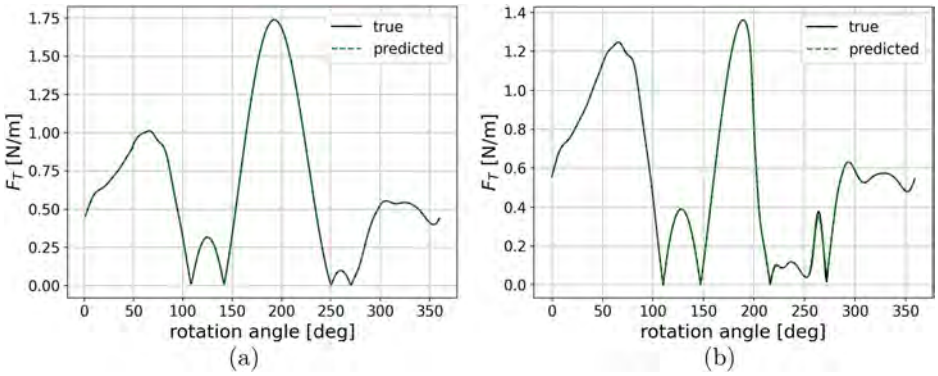

FIG. 21. F_T true and predicted signal for (a) average case and (b) worst case in the training dataset.

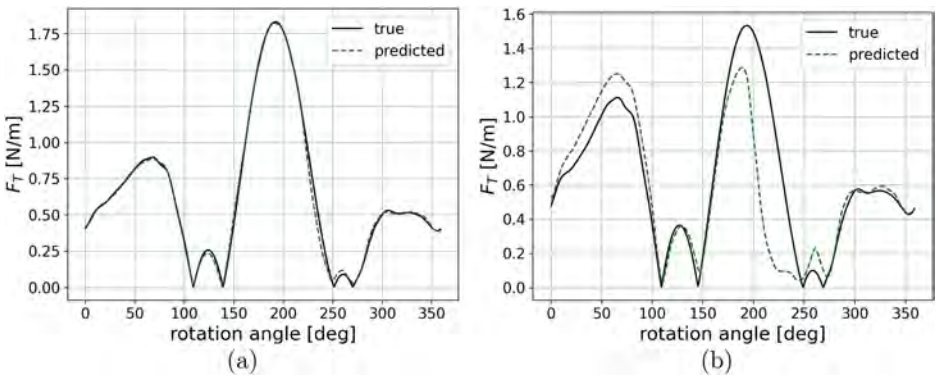

FIG. 22. F_T true and predicted signal for (a) average case and (b) worst case in the validation dataset.

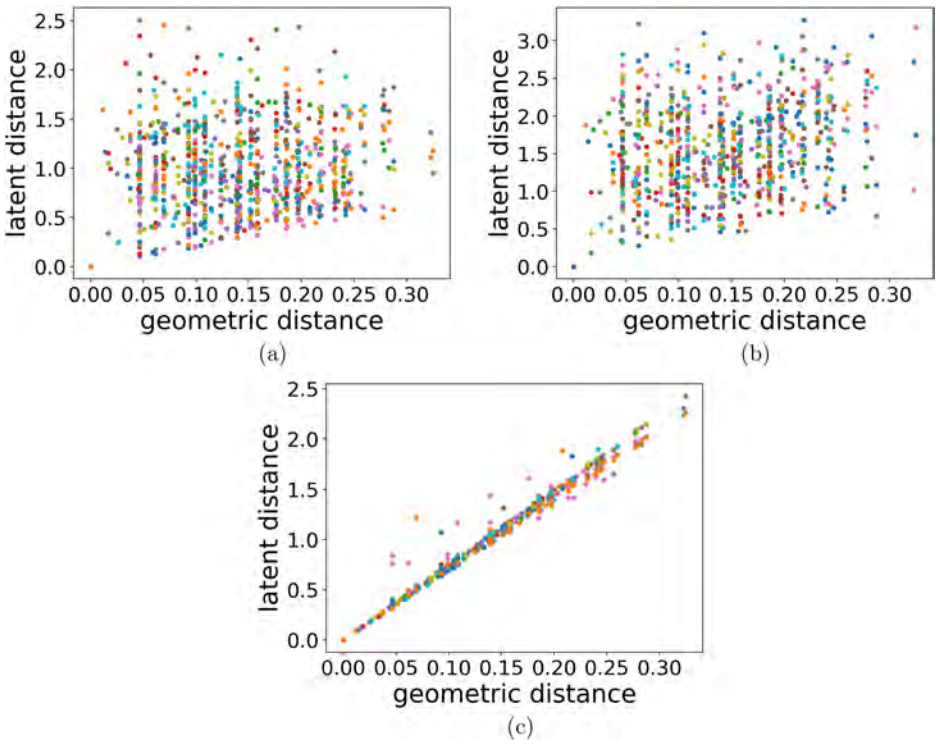

FIG. 23. Geometric distance vs latent codes distance for all possible couples of training geometries. Comparison between (a) non-regularized DeepSDF, (b) FRD regularization, and (c) PWD regularization.

original parameter space vs the power coefficient plotted on the latent space. Remarkably, the regularized encoder learned a latent space that mirrors the original parameter space without any direct or prior knowledge about the user-defined parametrization. Although the goal of the regularization is not to discover the true parametrization, this result is a further demonstration of its beneficial effect.

Figure 26 reports the SDF validation errors obtained with both regularization strategies. Table I shows the SDF reconstruction errors obtained with the original DeepSDF model and with the two regularization methods. FRD regularization method increased the prediction error on unseen geometries by about one order of magnitude compared to the non-regularized model. Conversely, with the PWD regularization the error on SDF prediction is reduced by approximately 50% in both training and validation datasets.

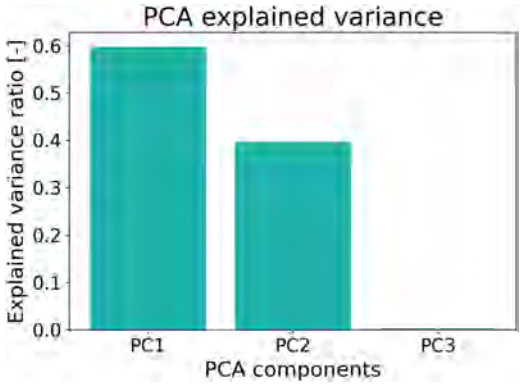

FIG. 24. Explained variance of the 3 PCA components computed on the three-dimensional regularized latent codes.

Next, the mapping block was re-trained with the regularized latent codes (and the rotation angle) as input. Relative errors on thrust force prediction are displayed in Fig. 27, from which we can observe that the average prediction error obtained with the FRD regularization is comparable to the prediction error produced by the original model. On the contrary, with the PWD regularization a substantial improvement of the prediction error is achieved, with the average error reduced by one order of magnitude.

Finally, the results obtained with the PWD regularization were compared with more standard approaches, namely, single and multi-fidelity POD combined with Gaussian process. As opposed to our approach, such models require a knowledge of the user-defined parametrization, in the sense that both regression models produce a prediction of the POD coefficients given the shape parameters and the rotation angle. From POD coefficients, the signal can be reconstructed by projection onto the POD basis computed during the offline training. This approach was applied in Ref. 49 to a test case involving the same wind turbine, considering a fixed geometry at varying rotational speeds. More details about the multi-fidelity model and its coupling with POD can be found in Ref. 52.

Figure 28 compares the predictions obtained with the three models. Table II reports the prediction error on F_T obtained with POD + GP models (single and multi-fidelity) and the errors obtained with our methods, namely, the baseline SWF and the SWF with FRD and PWD regularization. Not only PWD regularization improves the overall workflow accuracy, but our approach outperforms standard methods,

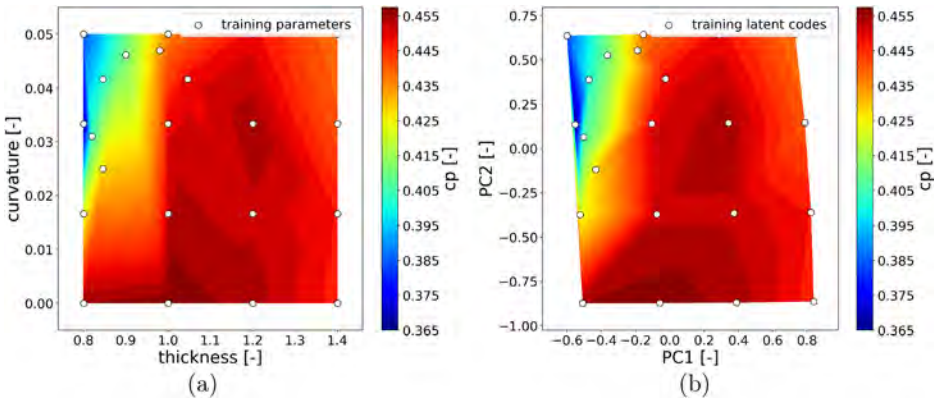

FIG. 25. Power coefficient on (a) true parameters space and (b) first two PCA components of regularized latent space. PCA components are rotated to facilitate the visual comparison.

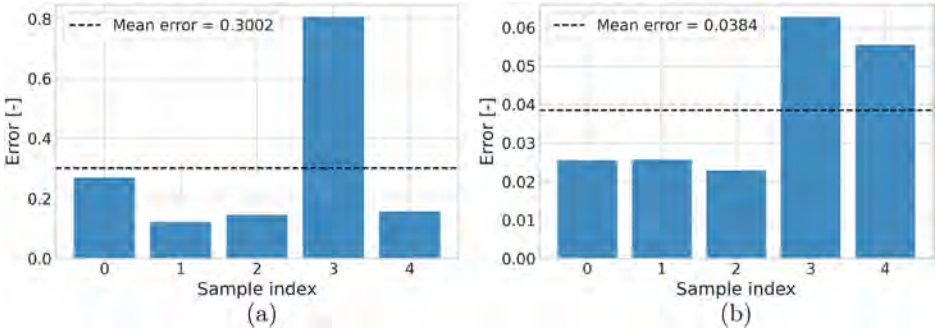

FIG. 26. Relative error on SDF prediction obtained with (a) FRD regularization and (b) PWD regularization.

TABLE I. SDF reconstruction errors. Boldface denotes the best result in terms of mean validation error.

Method	Dataset	Mean error	Standard deviation
DeepSDF	Training	0.0441	0.0055
	Validation	0.0667	0.0131
FRD reg.	Training	0.0611	0.0304
	Validation	0.3002	0.2585
PWD reg.	Training	0.0253	0.0036
	Validation	**0.0384**	0.0171

which leverage directly the underlying geometric parametrization. The same behavior is observed in the prediction of F_R signal (see Appendix B).

C. RAE2822 pressure prediction

In the second experiment, our goal is to predict the pressure coefficient, c_p, around an airfoil with variable shape. Geometric variations of the base airfoil are obtained by applying the deformation modes to the base RAE2822 as described in Sec. IV B. For this case, we use the complete workflow presented in Sec. II, which is composed of the geometric encoding, the output encoding, and the mapping block. For the geometric encoding, a DeepSDF model with four layers of size 128,

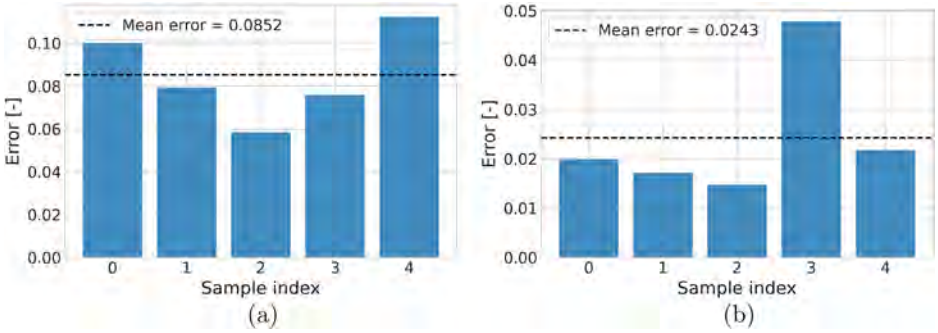

FIG. 27. Relative error on the prediction of the thrust force F_T on the validation dataset obtained with (a) FRD regularization and (b) with PWD regularization.

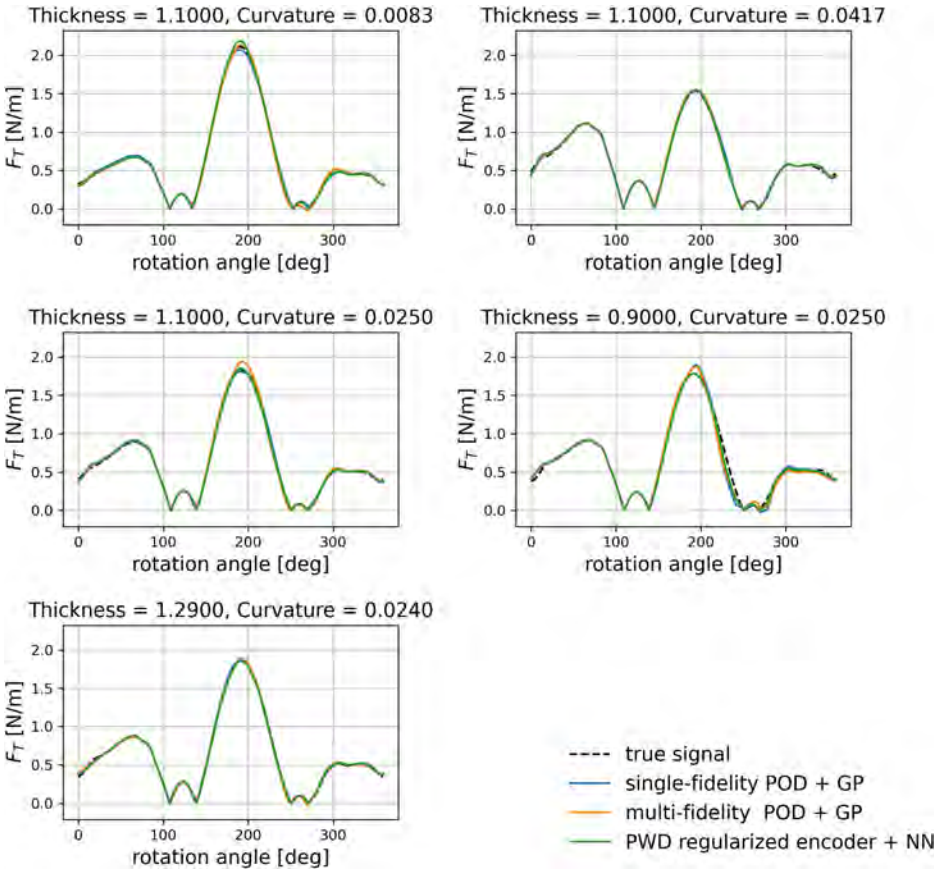

FIG. 28. Thrust force predictions obtained with regularized DeepSDF + NN, compared with single-fidelity and multi-fidelity POD + GP.

ReLU activations, and final tanh activation was trained for 2000 epochs. The learning rate was initialized at 1×10^{-4} for the NN weights and at 1×10^{-3} for latent code optimization. Both learning rates were halved every 200 epochs using a stepwise scheduler. The latent codes size was tuned as an hyperparameter to achieve the minimum error on the validation dataset and finally fixed at 8. The error distribution on the training and validation datasets for the SDF prediction is shown in Fig. 29. The mean relative error obtained is around 4% on the training dataset and 4.7% on the validation dataset, suggesting a very good representation capability.

TABLE II. F_T prediction errors. Boldface denotes the best result both in terms of mean validation error and standard deviation.

Method	Dataset	Mean error	Standard deviation
POD+GP	Training	0.0005	0.0006
single-fidelity	Validation	0.0357	0.0254
POD+GP	Training	0.0007	0.0004
multi-fidelity	Validation	0.0400	0.0206
DeepSDF + NN	Training	0.0028	0.0026
	Validation	0.1085	0.1221
FRD reg. + NN	Training	0.0060	0.0047
	Validation	0.0852	0.1123
PWD reg. + NN	Training	0.0059	0.0136
	Validation	**0.0242**	**0.0120**

For the output encoder, a similar architecture was employed with a layer size of 256. In this block, the spatial coordinates on which the pressure coefficient is defined, are transformed using the Fourier feature embedding method described in Sec. II B, with a number of frequencies $L = 4$. The model was trained for 2400 epochs starting with learning rates of 6×10^{-4} and 1×10^{-3} for NN weights and latent codes optimization, respectively. Both learning rates were halved every 400 training epochs. The errors are reported in Fig. 30 and show that the model is able to accurately represent the latent space for both training and validation pressure coefficient fields with an average error of 2.7% and 2.3%, respectively. Note that this error is not representative of the final prediction error, as it quantifies the inaccuracy due to the latent representation alone. Note also that in this initial experiment, no regularization has been applied in neither the geometric encoder, nor in the output encoder.

At this point, the training geometries and the associated c_p fields are both represented by latent vectors of size 8. In the mapping block, a NN with four layers of size 64 was trained to map the geometric latent space to the c_p latent space. By evaluating the trained model, we compute a prediction for the encoding of the target output for each training and validation airfoil. Once the predicted encoding is

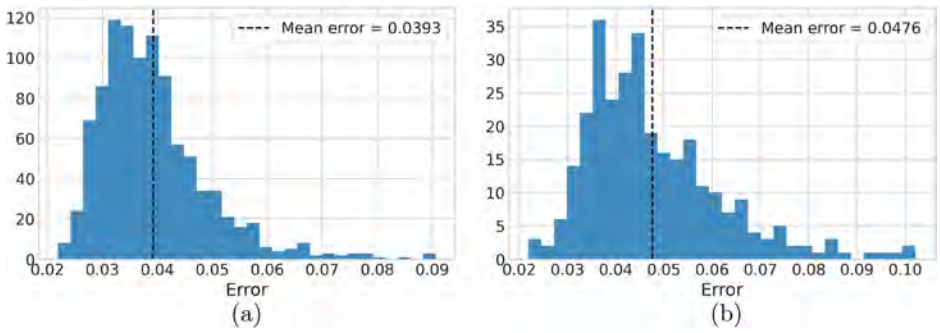

FIG. 29. Relative SDF prediction error on (a) training and (b) validation samples.

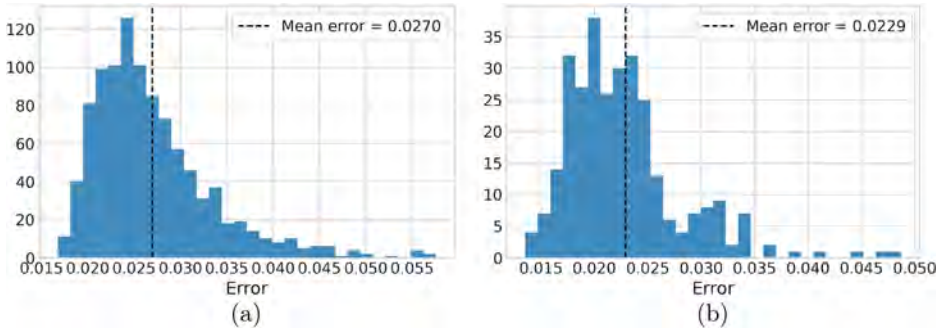

FIG. 30. Relative c_p representation error on (a) training and (b) validation samples.

reconstructed by the output encoder, the c_p field at the desired spatial location is reconstructed. The distribution of the prediction error is shown in Fig. 31, where the error is computed as the difference between the reconstructed c_p fields and the ground-truth value. Both training and validation errors are one order of magnitude higher than the reconstruction errors, reaching 50% in the worst cases. This suggests that, although the two encoders are able to represent accurately both the geometric space and the solution space, the mapping model struggles in learning the relation between the two latent spaces.

Therefore, the encoding blocks were re-trained using the regularization introduced in Sec. III. All the other hyper-parameters and the training data are kept to the same values. Appendix D reports the results of the tuning of the regularization weight. The comparison between the two proposed regularization methods, namely, the FRD and the PWD regularization, are shown in Fig. 32. There, we can observe that the FRD regularization induces a better correlation between the geometric distance and the latent space distance. Such correlation becomes even stronger with the PWD regularization method. The same behavior is observed in the output encoder, as depicted in Fig. 33.

Figures 34 and 35 show that both regularization methods lead to a significant improvement in the reconstruction of unseen samples. In particular, the mean SDF reconstruction error is reduced by one order of magnitude when compared to the non-regularized encoding [see Fig. 29(b)]. Tables III and IV report the error statistics for both the training and validation datasets, along with the count and maximum value of the outliers that were omitted from the histograms.

The mapping model was also re-trained to map the regularized geometric latent space to the output latent space using both regularization methods. The training loss and test loss are shown in Fig. 36. The training loss of non-regularized model is also shown in the same figure for comparison. The FRD regularization method substantially improved the mean loss evaluated on the test dataset. However, loss

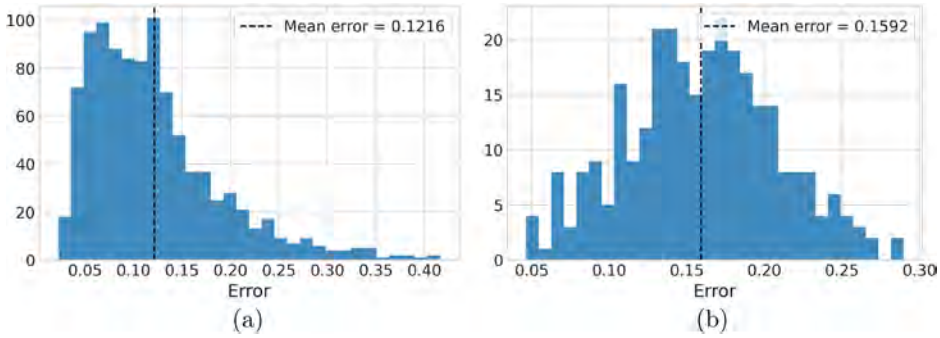

FIG. 31. Relative c_p prediction error on (a) training and (b) validation samples.

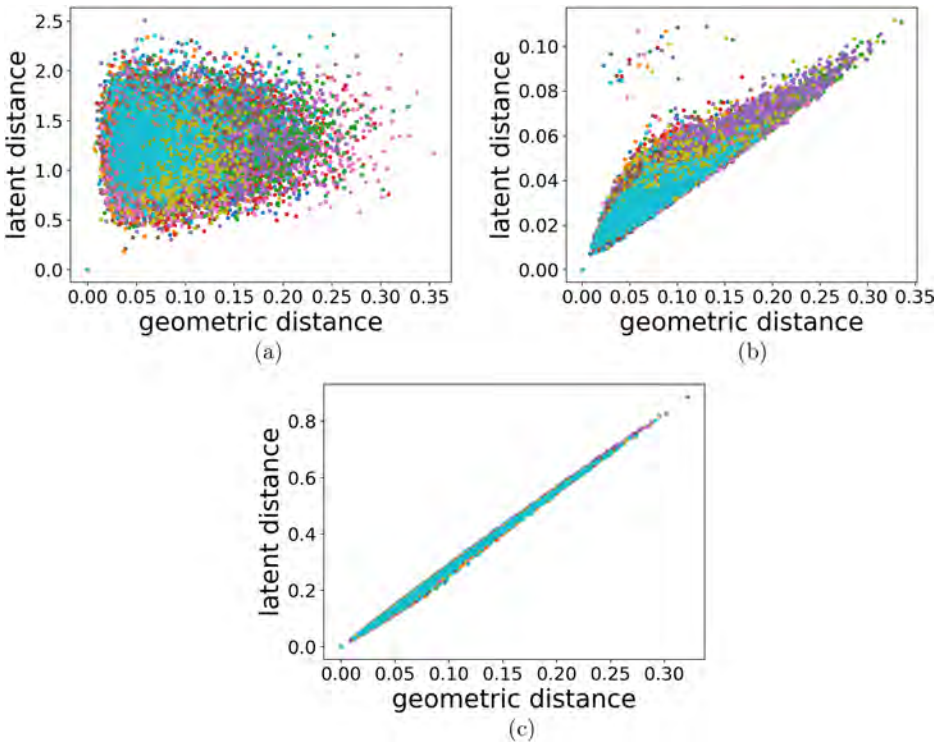

FIG. 32. Correlation between geometric distance and latent codes distance obtained by training the geometric encoding block (a) without regularization, (b) with FRD regularization, and (c) with PWD regularization. Each color represents the distance of training samples from a fixed reference sample.

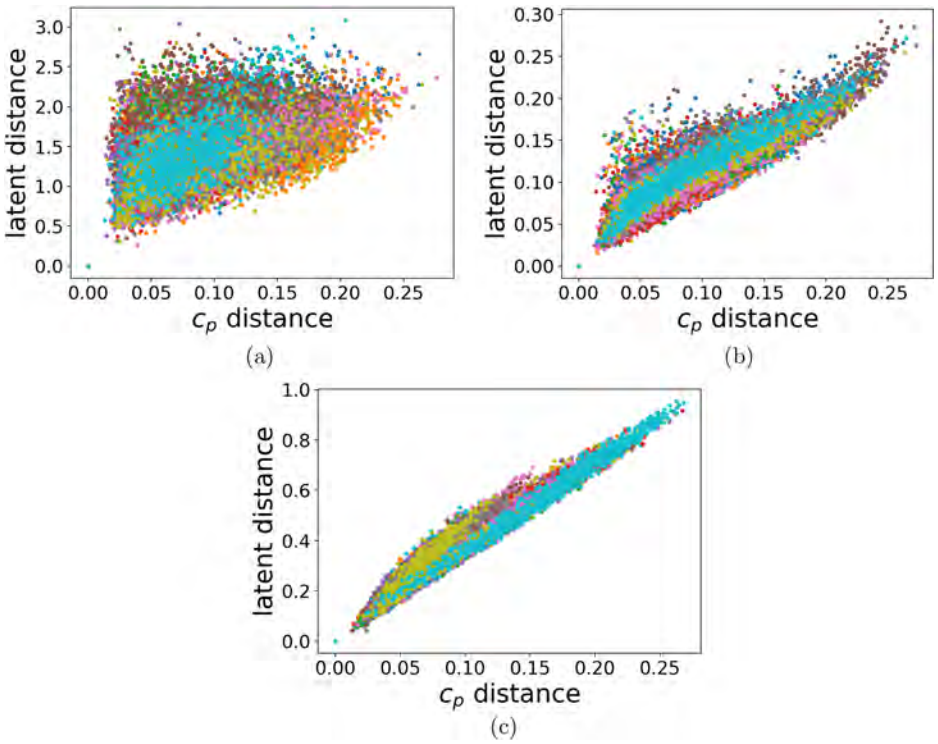

FIG. 33. Correlation between c_p distance and latent codes distance obtained by training the output encoding block (a) without regularization, (b) with FRD, and (c) with PWD regularization. Each color represents the distance of training samples from a fixed reference sample.

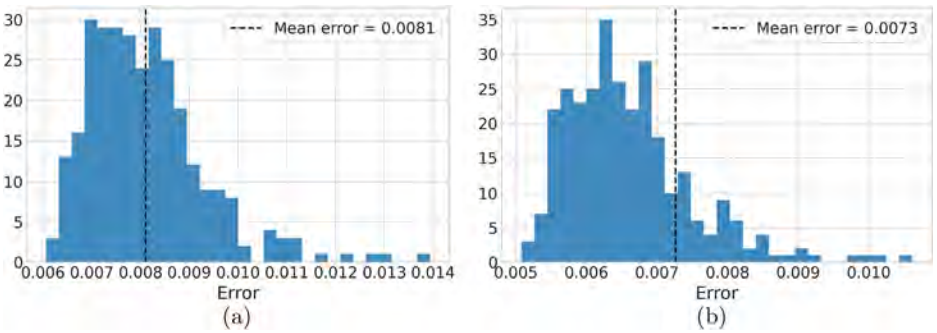

FIG. 34. Distribution of the relative error on SDF reconstruction on the validation dataset obtained with (a) FRD and (b) PWD regularization.

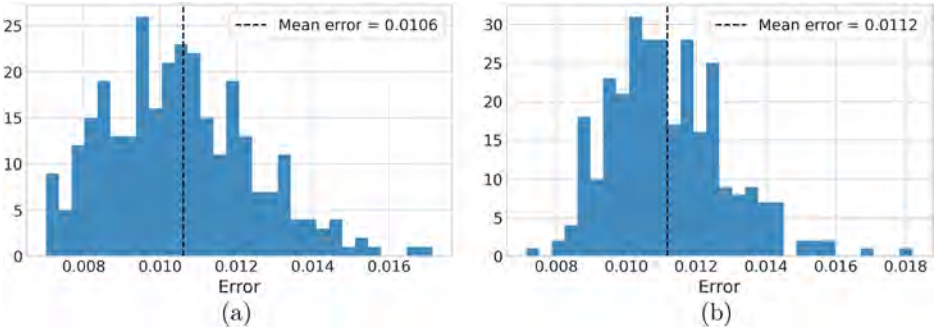

FIG. 35. Distribution of the relative c_p error on the validation dataset obtained with (a) FRD and (b) PWD regularization.

variability is still large. On the contrary, with the PWD regularization, we observe a significant reduction of both the mean test error and the loss variability. This pattern is also reflected in the distributions of prediction errors shown in Fig. 37. Both regularization methods exhibit a mean error that is one order of magnitude lower than the corresponding error obtained with the non-regularized encoders (shown in Fig. 31). However, the PWD regularization is characterized by a smaller variability. Table V summarizes the c_p prediction errors in training and validation datasets obtained with the baseline complete workflow and with the two regularization methods. Figures 38 and 39 display the true, predicted, and error fields for the best case and the median case within the validation dataset, obtained with the non-regularized workflow. The same visualization is produced for the workflow with PWD regularization and can be observed in Figs. 40 and 41. We can see that with PWD regularization the median case outperforms even the best case of the non-regularized model. Moreover, the model with PWD regularization achieves a higher accuracy in the boundary layer, which is typically challenging to predict and contributes to the computation of integral values, such as lift and drag coefficient. Additional c_p prediction for validation samples produced without regularization and with the two regularization methods is compared in Appendix C.

TABLE III. SDF reconstruction errors. Boldface denotes the best result both in terms of mean validation error.

Method	Dataset	Mean	Standard deviation	# outliers	Max outlier
DeepSDF	Training	0.0393	0.0104	3	0.0976
	Validation	0.0476	0.0140	0	···
FRD reg.	Training	0.0100	0.0031	1	0.0749
	Validation	0.0081	0.0012	0	···
PWD reg.	Training	0.0088	0.0023	3	0.0229
	Validation	**0.0072**	0.0082	2	0.1114

TABLE IV. c_p reconstruction errors. Boldface denotes the best result both in terms of mean validation error.

Method	Dataset	Mean	Standard deviation	# Outliers	Max Outlier
DeepSDF	Training	0.0269	0.0065	1	0.0721
	Validation	0.0229	0.0059	2	0.0554
FRD reg.	Training	0.0103	0.0020	1	0.0223
	Validation	**0.0106**	0.0023	2	0.0273
PWD reg.	Training	0.0119	0.0021	2	0.0234
	Validation	0.0112	0.0017	0	···

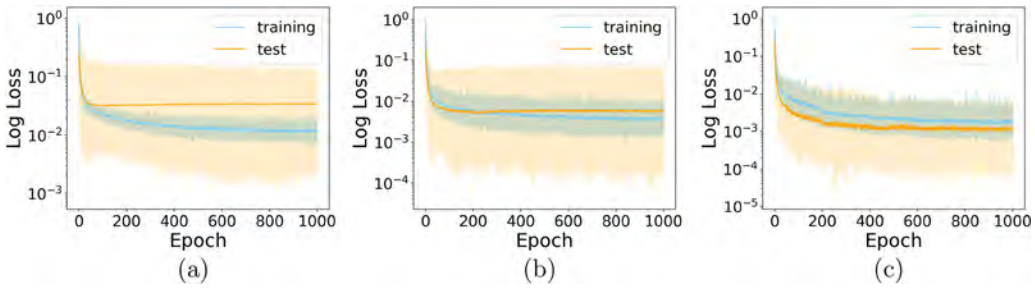

FIG. 36. Train and test loss during the training of the mapping block (a) without regularization, (b) with FRD regularization, and (c) with PWD regularization. Shaded areas represent the variation of the loss across batches within each epoch.

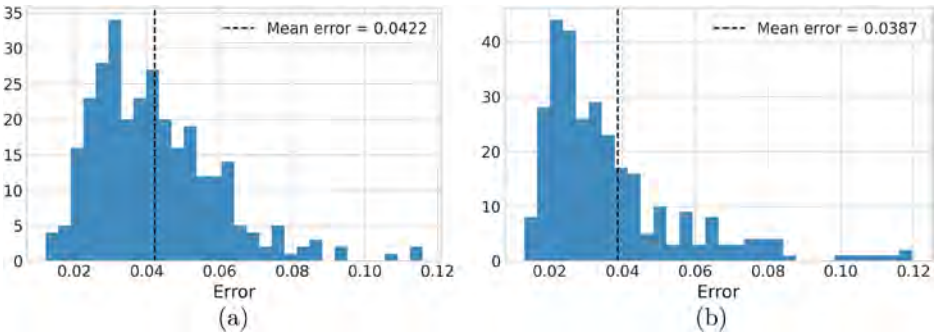

FIG. 37. Distribution of the relative prediction error on c_p for the validation dataset obtained with (a) FRD and (b) PWD regularization.

TABLE V. c_p prediction errors. Boldface denotes the best result both in terms of mean validation error.

Method	Dataset	Mean	Standard deviation	# outliers	Max outlier
DeepSDF	Training	0.1216	0.0693	2	0.4938
	Validation	0.1592	0.0489	0	···
FRD reg.	Training	0.2113	0.1398	0	···
	Validation	0.0422	0.0173	0	···
PWD reg.	Training	0.0362	0.0195	0	···
	Validation	**0.0387**	0.0283	3	0.2959

VI. CONCLUSIONS

A. Summary and key results

In this work, we presented a modular workflow for predicting PDEs solutions and derived quantities computed on domains with variable geometry. The input geometries and the associated solutions are encoded in fixed-size latent vectors through a specialized INR-based encoder. The resulting latent spaces are subsequently mapped via a feed-forward neural network. We also proposed a simplified workflow for the prediction of 1D signals, which maps the geometric encoding directly to the quantity of interest evaluated at the desired space/time locations without the need of the output encoder. Moreover, we introduced two methods for regularizing the INR-based encoding models,

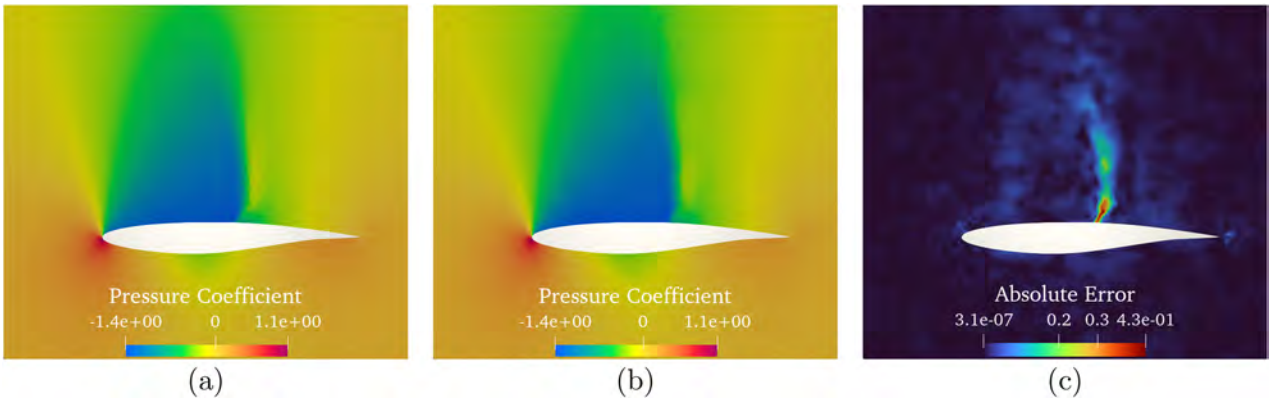

FIG. 38. Best c_p prediction obtained without regularization on the validation dataset. (a) True, (b) predicted, and (c) error fields.

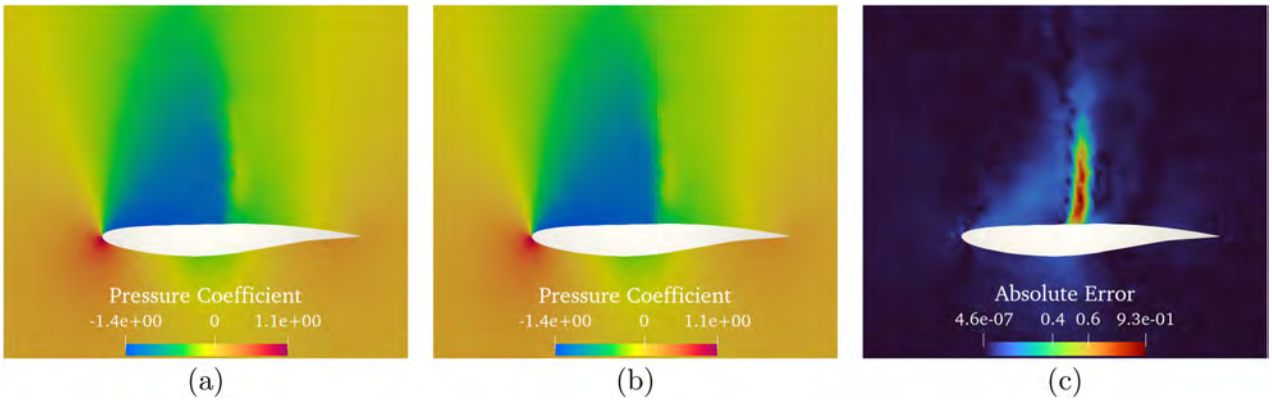

FIG. 39. c_p prediction obtained without regularization corresponding to the median error on the validation dataset. (a) True, (b) predicted, and (c) error fields.

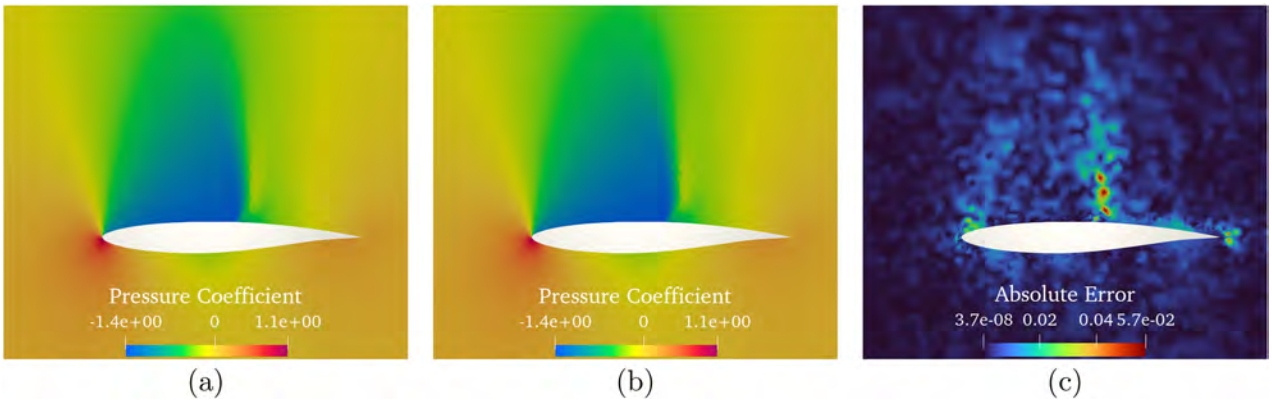

FIG. 40. Best c_p prediction obtained with PWD regularization on the validation dataset. (a) True, (b) predicted, and (c) error fields.

which allows to preserve the geometric distances between the training samples in the latent space. We tested the simplified and the complete workflow on two problems: The prediction of 1D signals (namely, the radial and thrust force) on a vertical axis turbine blade with varying thickness and curvature, and the prediction of 2D pressure coefficient fields on geometric variations of the transonic RAE2822 airfoil. The experimental results demonstrate the effectiveness of the proposed regularization methods. In particular, the PWD regularization

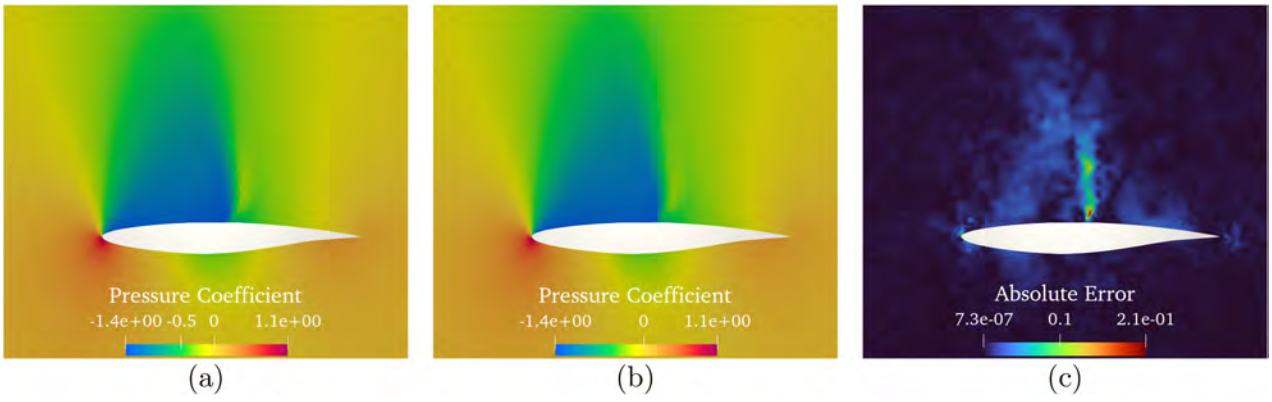

FIG. 41. c_p prediction obtained with PWD regularization corresponding to the median error on the validation dataset. (a) True, (b) predicted, and (c) error fields.

substantially enhanced the overall accuracy in the prediction of unseen configurations in both experiments, reducing the average relative error by one order of magnitude with respect to the non-regularized workflow. By performing a PCA analysis of the geometric latent codes, we observed that the regularized latent spaces can be reduced to a subspace with the exact same dimensionality of the true parameter space with negligible information loss.

B. Limitations and future work

INRs offer a flexible and powerful tool for mesh-free geometry encoding. However, this approach is sensible to the choice of the spatial point sampling. In this work, we employed the sampling strategy suggested in Ref. 34, which provides a higher density near the surface of the geometry to capture the zero level set in more detail. This approach is generally effective, but still requires careful specification of some implementation details, such as the number of points sampled on the surface, the variance of Gaussian perturbations, and the direction of the perturbation (normal to the surface or random). We observed that the representation accuracy of the encoder strongly depends on the choice of these hyper-parameters, which need to be tuned for each specific test case.

A possible direction for future improvements concerns problems with non-geometrical (PDE-related) parameters. Although this work is focused on shape variation, it can be easily extended also to such cases. Thanks to the modularity of this workflow, it is sufficient to stack the additional parameters to the input layer of the output encoder and the mapping model. However, further tests are necessary to verify the effectiveness of this extension. Future investigations should also consider 3D problems. Although our workflow is designed to handle 3D cases, dedicated experiments are needed to assess the performance in this scenario.

Beyond these methodological aspects, the proposed framework is well suited for applications, such as shape optimization and design space exploration, where fast and accurate predictions of PDEs solutions are required. For canonical configurations, such as the RAE2822 airfoil, accurate reconstruction of unsteady pressure fields is an important prerequisite for complex tasks, including data-driven modeling of transonic buffet-induced aerodynamic noise, as demonstrated in recent studies.[53,54] Future work could apply the workflow presented here and compare its performance to the data-driven models used in state-of-the-art methods for aerodynamic noise prediction. In addition, this framework has strong potential for design of active flow control strategies, where surrogate models must capture the dynamic response of the flow to control inputs, such as blowing, suction, or surface actuation. By providing efficient reconstruction of unsteady fields or time varying signals, the proposed approach could support the design of real-time control strategies, enable rapid evaluation of control effectiveness across varying geometries, and facilitate integration with optimization loops for closed-loop control.

C. Strengths and contributions

The main strengths of the proposed workflow lie in its modularity, generality, parameter-agnostic design, and mesh-free nature. Training an end-to-end model usually provides more accurate results; however, modularity allows to reuse the blocks for different problems without re-training the entire workflow. This is especially convenient for large datasets, when training a single monolithic model would require a significant computational effort. Thanks to the encoding strategy, the workflow is totally parameter-agnostic. Our approach does not require knowledge of either the parameters or the method used to generate the geometric variations since it learns all relevant features directly from the training data. Finally, by leveraging INRs continuous learning, our workflow does not depend on a fixed discretization and can predict the output of interest with a resolution that is (in principle) unlimited.

The main contribution of this work is represented by the regularization methods. The proposed regularization, especially PWD regularization, improves the standard INR-based encoding model by preserving the geometric distances in the latent space. As demonstrated by our numerical experiments, this property leads to a significant improvement of the generalization capability of the mapping model and to a higher accuracy in the prediction for unseen geometries.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Laura Balasso: Conceptualization (equal); Data curation (equal); Formal analysis (equal); Methodology (equal); Software (equal); Visualization (equal); Writing – original draft (equal). **Alessandro Alaia:** Conceptualization (equal); Project administration (equal); Supervision (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

APPENDIX A: TRAINING AND INFERENCE TIME

All experiments were implemented in PyTorch[51] v2.1. The training of the encoding models was performed on a single NVIDIA A10 GPU. Mapping models were trained on a single CPU.

1. VAWT

The three geometric encoders, i.e., DeepSDF, FRD regularization, and PWD regularization, have four hidden layers of size 128 and were trained for 2000 epochs on a training dataset composed of 42 geometries with 35 000 spatial points for each geometry and batch size 4 (geometries). The GPU training times are reported in the first column of Table VI. Latent code optimization was performed for one validation geometry at a time for 400 epochs each. The mean elapsed time across the validation dataset is approximately 1.2 s. Since the regularization is applied only to the training, latent code optimization time is the same for the three encoders. In the mapping block, a fully connected NN with four layers of size 256 was trained for 2000 epochs on a training dataset of 22 geometries (with 35 000 points each) and batch size 2. Training time is around 7 min on a single CPU.

TABLE VI. GPU training time of encoding models.

	VAWT geom. encoder	RAE2822 geom. encoder	RAE c_p encoder
DeepSDF	8.2 min	68.7 min	205 min
FRD reg.	9.7 min	74.5 min	204 min
PWD reg.	11.6 min	73 min	200 min

2. RAE2822

The geometric encoder architecture is composed of four hidden layers of size 128. The models were trained for 2000 epochs on 1000 geometry with 20 000 spatial points each and batch size 16 (geometries). The training time for DeepSDF and the two regularized encoders is reported in the second column of Table VI. Latent code optimization took on average 1.5 s for 300 epochs for each validation geometry. The output encoders have four layers of size 256 and are trained for 2000 epochs on 1000 c_p fields with 25 900 points per field and batch size 16. The training time for the non-regularized and the two regularized encoders is reported in the third column of Table VI. Latent code optimization took on average 2.3 s for 400 epochs for each validation field. The mapping block, with four layers of size 64, was trained for 1000 epochs on 1000 samples (800 training and 200 test) with batch size 32. Training time on a single CPU was approximately 6 min.

TABLE VII. Elapsed time for each step of the prediction of an unseen sample.

Step	Elapsed time (s)	
	RAE	VAWT
SDF computation	0.2971	0.2496
Latent code optimization	0.97	1.20
Latent code to field reconstruction	0.13	0.003
Total	**1.3971**	**1.4526**

3. Prediction of unseen samples

Table VII reports the elapsed time for each step during inference. Latent code optimization and field prediction were performed on an NVIDIA A10 GPU, while SDF computation was executed serially on a single CPU core. The time required for latent code to field reconstruction is significantly lower in the VAWT case. In this case, the output value is predicted directly from the geometric latent code and the simplified workflow is employed. The total prediction time is approximately 1.4 s for both methods, suggesting that the proposed approach is suitable for near-real-time applications such as shape optimization and design exploration.

APPENDIX B: VAWT RADIAL FORCE PREDICTION

In this appendix, the results reported in Sec. V B for F_T signal are reproduced for the radial force F_R. The bar plots in Fig. 42 show the relative prediction errors on training and validation datasets

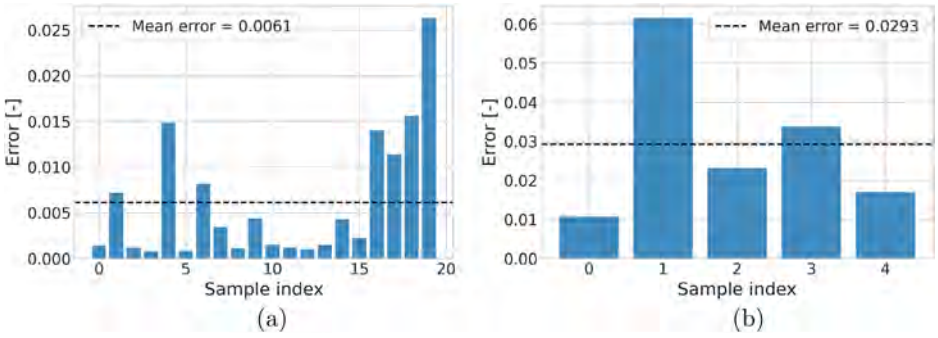

FIG. 42. F_R relative prediction error on (a) training and (b) validation samples.

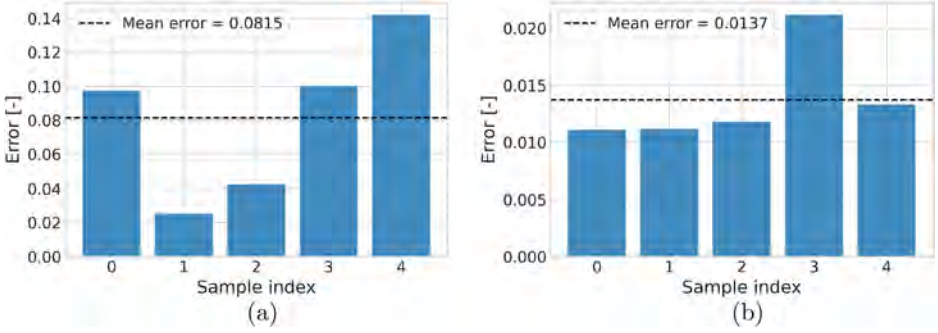

FIG. 43. Relative error on the prediction of the thrust force F_R on the validation dataset obtained with the (a) FRD regularization and (b) with PWD regularization.

TABLE VIII. F_R prediction errors. Boldface denotes the best result both in terms of mean validation error and standard deviation.

Method	Dataset	Mean error	Standard deviation
POD+GP single-fidelity	Training	0.0006	0.0005
	Validation	0.0202	0.0106
POD+GP multi-fidelity	Training	0.0009	0.0004
	Validation	0.0184	0.0050
DeepSDF + NN	Training	0.0061	0.0067
	Validation	0.0293	0.0178
FRD reg. + NN	Training	0.0024	0.0025
	Validation	0.0815	0.0423
PWD reg. + NN	Training	0.0061	0.0043
	Validation	**0.0137**	**0.0038**

obtained with the non-regularized SWF. Figure 43 displays the prediction errors on validation samples obtained with FRD and PWD regularization methods. As observed for thrust force, PWD regularization improves the prediction accuracy on unseen sample. Table VIII reports the error statistics obtained with POD+GP models (single and multi-fidelity) and the errors obtained with our methods. Figure 44 compares the predictions obtained with PWD regularization and two models based on POD and Gaussian Processes. Consistently with F_T results, PWD regularization outperforms both POD+GP methods and the standard workflow based on DeepSDF and feed-forward NN.

APPENDIX C: RAE2822 PRESSURE COEFFICIENT PREDICTIONS

In this appendix, two examples of c_p prediction are presented. In Fig. 45, the same validation case is predicted using the baseline

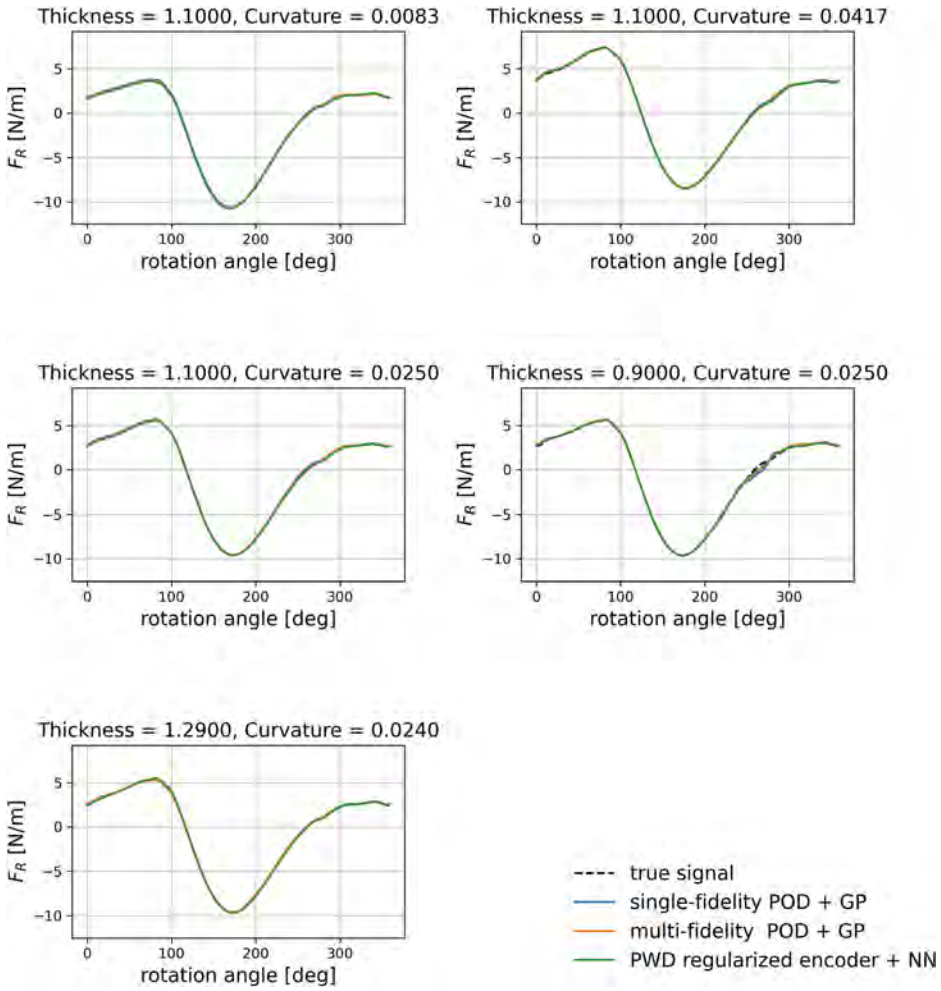

FIG. 44. Radial force prediction obtained with regularized DeepSDF + NN, compared with single-fidelity and multi-fidelity POD + GP.

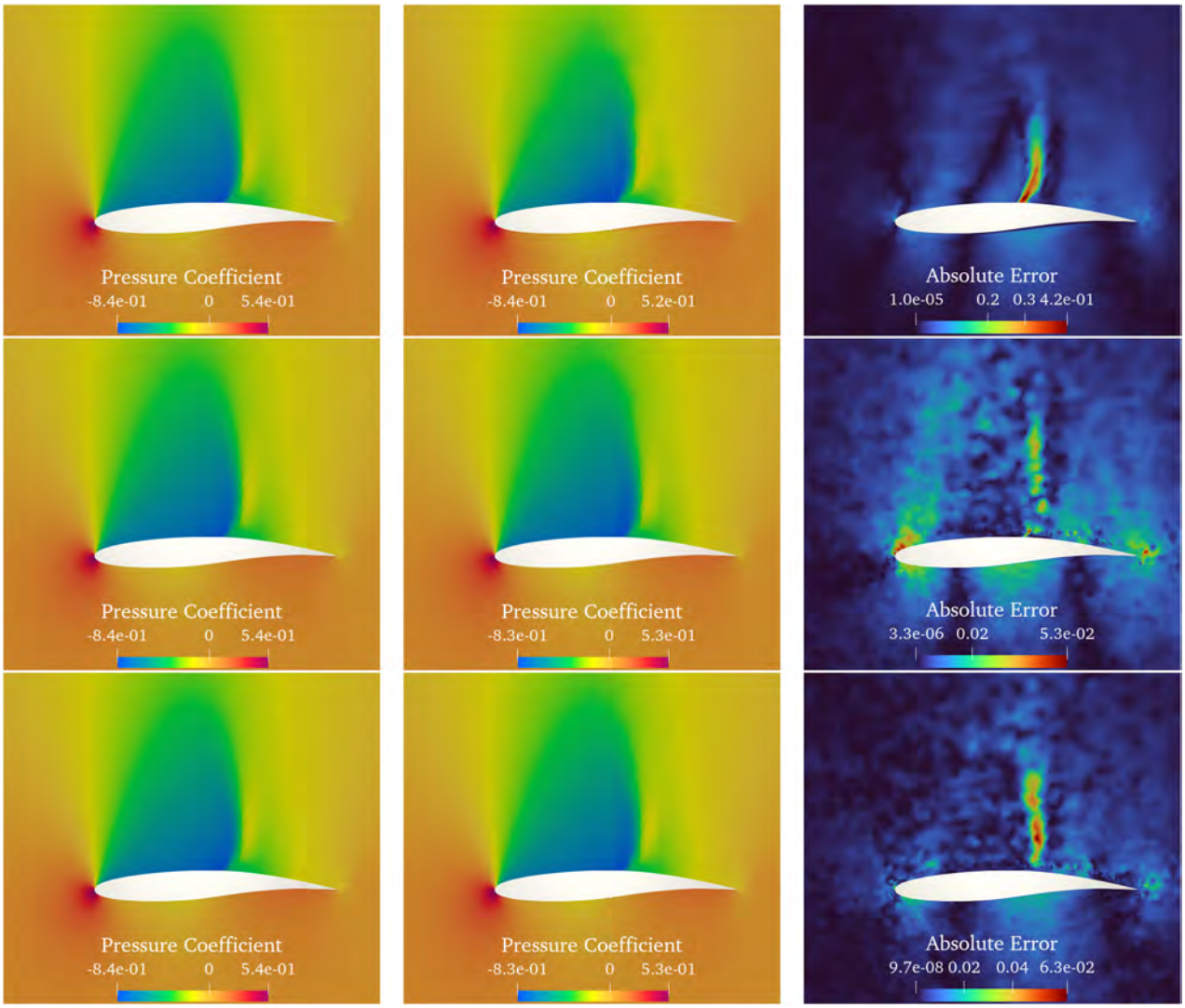

FIG. 45. Pressure coefficient prediction for an unseen geometry obtained with the complete workflow without regularization (upper row), with FRD regularization (central row), and with PWD regularization (bottom row). For each method, true c_p field (left column), predicted c_p (central column), and absolute error field (right column).

workflow and the workflow with FRD and PWD regularization. The same visualization is shown in Fig. 46 for another validation case. In the first case, both regularization methods reduced the maximum absolute error by one order of magnitude. In the second case, the maximum was reduced by approximately 40% and the extent of the error region decreased substantially.

APPENDIX D: TUNING OF REGULARIZATION WEIGHT

The regularization weight λ in Eq. (15) was tuned for each experiment presented in Sec. V. In the VAWT experiment, the regularized geometric encoder was trained using six different values of λ ranging from 0.01 to 0.5. For each value, the mean SDF prediction error on the validation samples was computed and is shown in Fig. 47. The same figure also reports the corresponding F_T prediction error for each λ, illustrating how the regularization weight affects the final prediction. To obtain this error, the mapping block was re-trained for each trained geometric encoder. The encoding model corresponding to $\lambda = 0.1$ exhibits the best results in terms of both SDF reconstruction and F_T prediction.

In the RAE experiment, the regularization weight was tuned for both the geometric and the output encoders. Figure 48 shows that the best geometric encoding results were obtained with $\lambda = 0.1$. In this case, the corresponding output prediction error is

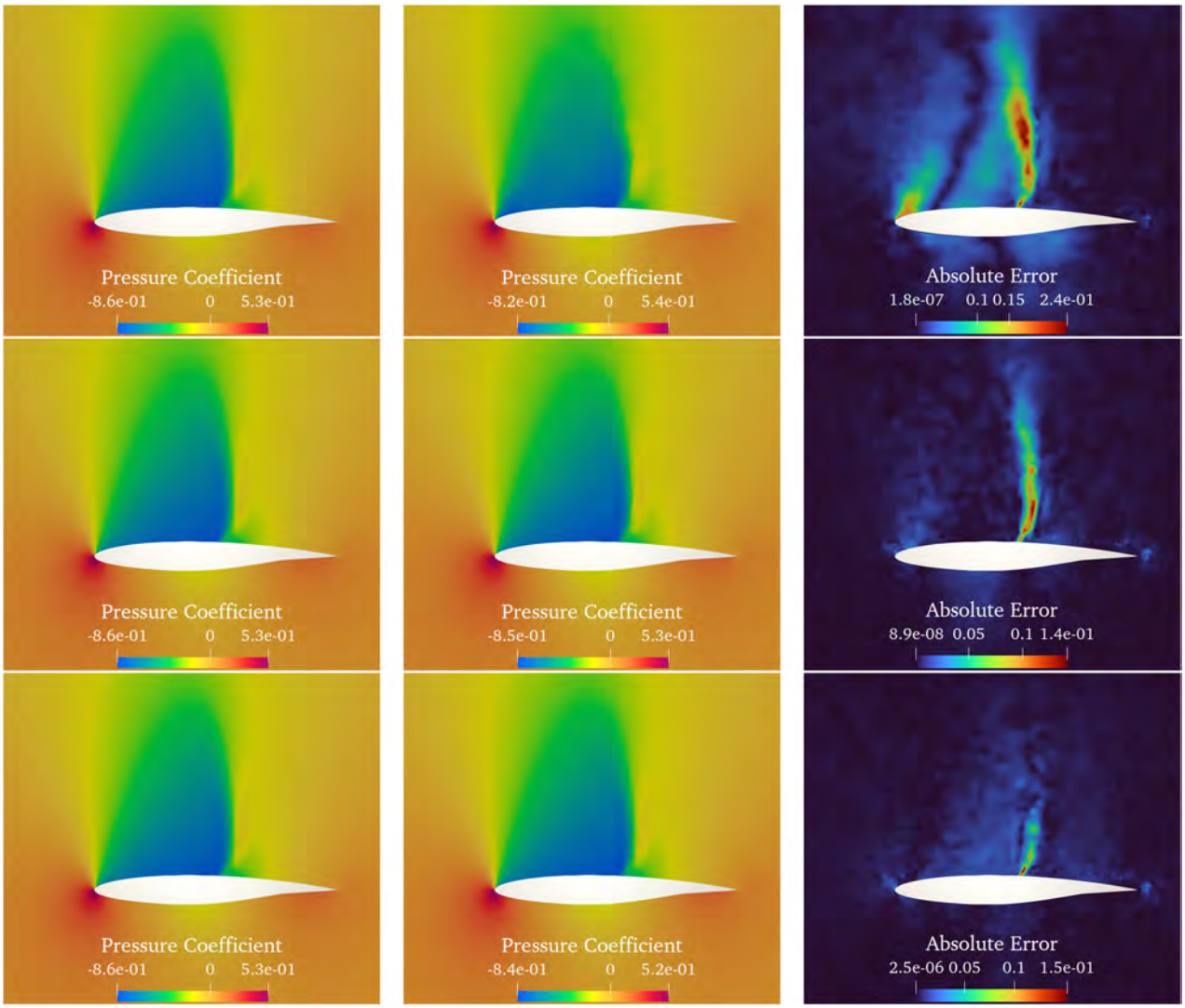

FIG. 46. Pressure coefficient prediction for an unseen geometry obtained with the complete workflow without regularization (upper row), with FRD regularization (central row), and with PWD regularization (bottom row). For each method, true c_p field (left column), predicted c_p (central column), and absolute error field (right column).

not reported since it depends also on the value of λ used in the output encoding.

Based on the tuning results presented above for the geometric encoders, where a regularization weight $\lambda = 0.1$ consistently provided good performance, we adopted this value for the output encoder as well. Subsequent sensitivity analysis (illustrated in Fig. 49) reveals that although alternative values of λ can yield marginally improved error metrics, the model's performance is not highly sensitive to λ, demonstrating robustness to the choice of the regularization weight. However, Fig. 49 also shows significantly reducing λ leads to deteriorated results, confirming that the regularization term plays an important role in improving the model's accuracy.

APPENDIX E: PWD SCALABILITY ANALYSIS

PWD regularization requires the computation of SDF distances between all possible pairs of training geometries. Although this computation is performed only once, prior to training, it can still be time-consuming for very large datasets, i.e., datasets with a large number of training samples (geometries) or a high number of spatial points per sample. While the number of spatial points per sample is not expected to vary significantly, the number of samples is test-case dependent. Moreover, for large datasets, memory usage can become a limiting factor, as all samples must be loaded into RAM in order to compute the pair-wise distances. Figure 50 shows the elapsed time on 16-cores CPU and the memory usage for the

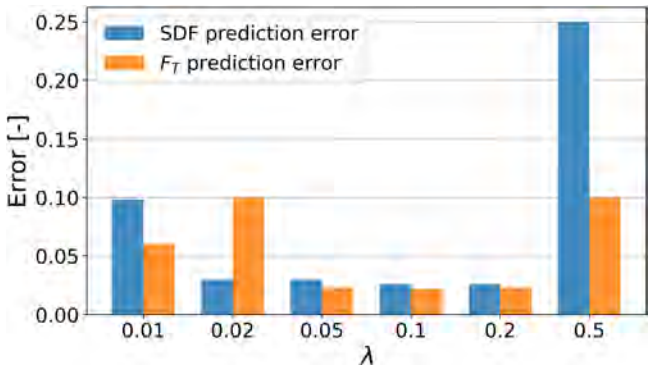

FIG. 47. SDF prediction error and F_T prediction error for six values of regularization weight λ in a VAWT regularized geometric encoder.

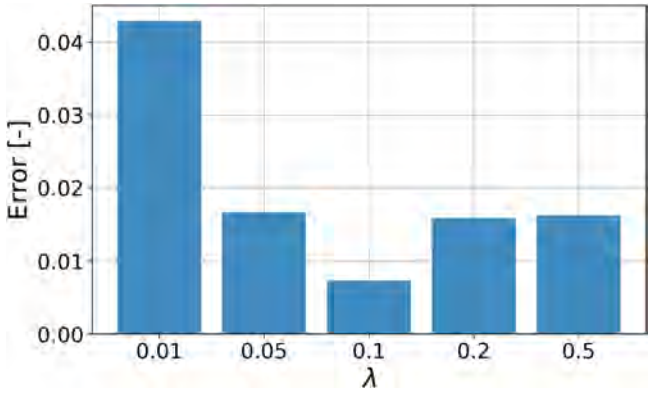

FIG. 48. SDF prediction error for different values of regularization weight λ.

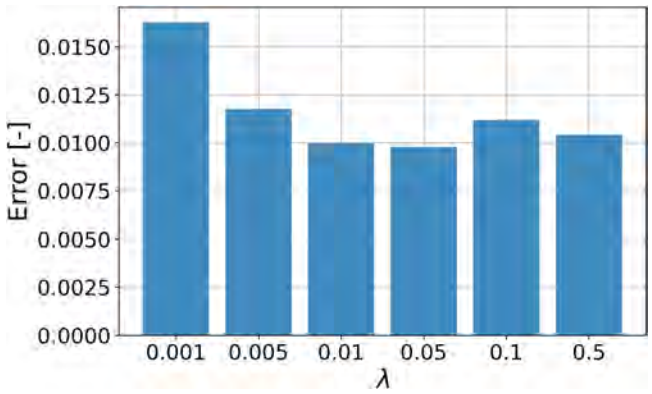

FIG. 49. c_p reconstruction error for different values of regularization weight λ.

computation of pair-wise distances on an increasing number of training geometries. For this analysis, the number of spatial samples is $128^2 = 16\,384$, which corresponds to the case of computing the SDF on a cartesian grid of size 128×128. Both time and RAM

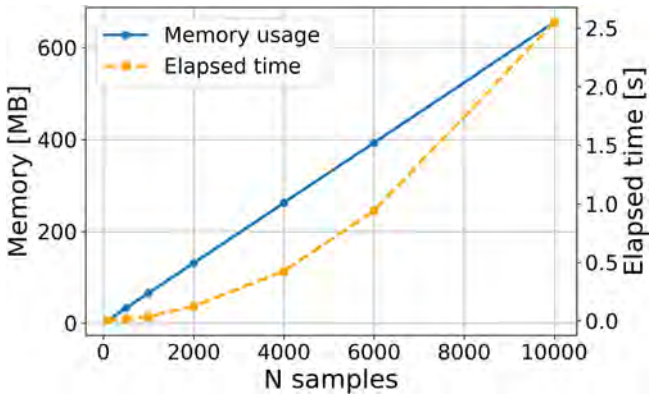

FIG. 50. Distances computation time and memory usage for an increasing number of samples, each one of size 16 384.

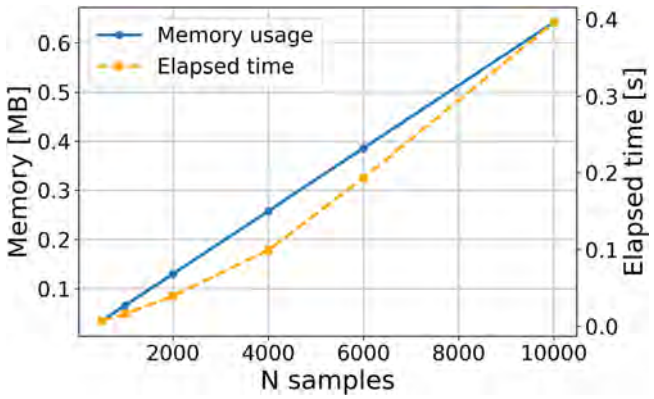

FIG. 51. Latent distances computation time and memory usage for an increasing number of samples.

consumption remain affordable with 10 000 samples, and larger datasets are uncommon in real-world industrial applications.

Another extra cost due to PWD regularization is the computation of distances between batch latent codes and all the other training codes during training. In order to analyze this cost, the latent distance is computed for a fixed latent size $d = 16$ and batch size $B = 32$, for an increasing number of training samples. Figure 51 plots the memory usage and the time consumption due to this computation for a single training epoch.

REFERENCES

[1]L. Sirovich, "Turbulence and the dynamics of coherent structures. I. Coherent structures," Q. Appl. Math. **45**, 561–571 (1987).
[2]P. Benner, S. Gugercin, and K. Willcox, "A survey of projection-based model reduction methods for parametric dynamical systems," SIAM Rev. **57**, 483–531 (2015).
[3]E. Iuliano, "Towards a POD-based surrogate model for CFD optimization," in Proceedings of the Eccomas CFD & Optimization Conference, Antalya, Turkey, 2011.
[4]A. E. Boumesbah and T. Henneron, "Parametric geometric metamodel of nonlinear magnetostatic problem based on POD and RBF approaches," IEEE Trans. Magn. **58**, 6000306 (2021).

[5]C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning, Adaptive Computation and Machine Learning* (MIT Press, 2006).
[6]F. Casenave, B. Staber, and X. Roynard, "MMGP: A mesh morphing Gaussian process-based machine learning method for regression of physical problems under nonparametrized geometrical variability," in *Advances in Neural Information Processing Systems* (Curran Associates, 2023), Vol. 36, pp. 43972–43999.
[7]D. Toal, N. Bressloff, and A. Keane, "Geometric filtration using POD for aerodynamic design optimization," in *26th AIAA Applied Aerodynamics Conference* (AIAA, 2008).
[8]F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," IEEE Trans. Neural Networks **20**, 61–80 (2008).
[9]X. Hui, J. Bai, H. Wang, and Y. Zhang, "Fast pressure distribution prediction of airfoils using deep learning," Aerosp. Sci. Technol. **105**, 105949 (2020).
[10]C. Duru, H. Alemdar, and O. U. Baran, "A deep learning approach for the transonic flow field predictions around airfoils," Comput. Fluids **236**, 105312 (2022).
[11]Z. Nie, H. Jiang, and L. B. Kara, "Stress field prediction in cantilevered structures using convolutional neural networks," J. Comput. Inf. Sci. Eng. **20**, 011002 (2020).
[12]E. Hoq, O. Aljarrah, J. Li, J. Bi, A. Heryudono, and W. Huang, "Data-driven methods for stress field predictions in random heterogeneous materials," Eng. Appl. Artif. Intell. **123**, 106267 (2023).
[13]J.-Z. Peng, X. Liu, Z.-D. Xia, N. Aubry, Z. Chen, and W.-T. Wu, "Data-driven modeling of geometry-adaptive steady heat convection based on convolutional neural networks," Fluids **6**, 436 (2021).
[14]O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (Springer, 2015), pp. 234–241.
[15]M. S. Khorrami, J. R. Mianroodi, N. H. Siboni, P. Goyal, B. Svendsen, P. Benner, and D. Raabe, "An artificial neural network for surrogate modeling of stress fields in viscoplastic polycrystalline materials," npj Comput. Mater. **9**, 37 (2023).
[16]N. Thuerey, K. Weißenow, L. Prantl, and X. Hu, "Deep learning methods for Reynolds-averaged Navier–Stokes simulations of airfoil flows," AIAA J. **58**, 25–36 (2020).
[17]S. J. Jacob, M. Mrosek, C. Othmer, and H. Köstler, "Deep learning for real-time aerodynamic evaluations of arbitrary vehicle shapes," arXiv:2108.05798 (2021).
[18]I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," Commun. ACM **63**, 139–144 (2020).
[19]H. Jiang, Z. Nie, R. Yeo, A. B. Farimani, and L. B. Kara, "Stressgan: A generative deep learning model for two-dimensional stress distribution prediction," J. Appl. Mech. **88**, 051005 (2021).
[20]J. Hu, Z. Lu, and Y. Yang, "Generative prediction of flow field based on the diffusion model," arXiv:2407.00735 (2024).
[21]D. Chen, X. Gao, C. Xu, S. Chen, J. Fang, Z. Wang, and Z. Wang, "FlowGAN: A conditional generative adversarial network for flow prediction in various conditions," in *IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)* (IEEE, 2020), pp. 315–322.
[22]C. Drygala, F. di Mare, and H. Gottschalk, "Generalization capabilities of conditional GAN for turbulent flow under changes of geometry," arXiv:2302.09945 (2023).
[23]A. Padmaprabhan, S. Hari, N. P. Thomas, K. S. Chadha, S. Sidhardh, V. Chinthapenta, and P. Kumar, "GO-GAN: Geometry optimization generative adversarial network for achieving optimized structures with targeted physical properties," arXiv:2502.00416 (2025).
[24]M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond Euclidean data," IEEE Signal Process. Mag. **34**, 18–42 (2017).
[25]F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2017), pp. 5115–5124.
[26]F. Hadizadeh, W. Mallik, and R. K. Jaiman, "A graph neural network surrogate model for multi-objective fluid-acoustic shape optimization," arXiv:2412.16817 (2024).
[27]J. Wu, C. Du, B. Dillenburger, and M. A. Kraus, "Fast prediction of stress distribution: A GNN-based surrogate model for unstructured mesh FEA," in *Proceedings of the IASS Symposium: Redefining the Art of Structural Design* (International Association for Shell and Spatial Structures, 2024).
[28]T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. Battaglia, "Learning mesh-based simulation with graph networks," in International Conference on Learning Representations, 2020.
[29]Z. Li, N. Kovachki, C. Choy, B. Li, J. Kossaifi, S. Otta, M. A. Nabian, M. Stadler, C. Hundt, K. Azizzadenesheli *et al.*, "Geometry-informed neural operator for large-scale 3D PDEs," in *Advances in Neural Information Processing Systems* (Curran Associates, 2023), Vol. 36, pp. 35836–35854.
[30]X. Fu, F. Zhou, D. Peddireddy, Z. Kang, M. B. G. Jun, and V. Aggarwal, "An FEA surrogate model with boundary oriented graph embedding approach," arXiv:2108.13509 (2021).
[31]S. Li, M. Zhang, and M. D. Piggott, "End-to-end wind turbine wake modelling with deep graph representation learning," Appl. Energy **339**, 120928 (2023).
[32]B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," Commun. ACM **65**, 99–106 (2021).
[33]Z. Chen and H. Zhang, "Learning implicit fields for generative shape modeling," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, 2019), pp. 5939–5948.
[34]J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, 2019), pp. 165–174.
[35]X. Huang, Z. Ye, H. Liu, S. Ji, Z. Wang, K. Yang, Y. Li, M. Wang, H. Chu, F. Yu *et al.*, "Meta-auto-decoder for solving parametric partial differential equations," in *Advances in Neural Information Processing Systems* (Curran Associates, 2022), Vol. 35, pp. 23426–23438.
[36]J. Zehnder, Y. Li, S. Coros, and B. Thomaszewski, "NTopo: Mesh-free topology optimization using implicit neural representations," in *Advances in Neural Information Processing Systems* (Curran Associates, 2021), Vol. 34, pp. 10368–10381.
[37]G. Catalani, S. Agarwal, X. Bertrand, F. Tost, M. Bauerheim, and J. Morlier, "Neural fields for rapid aircraft aerodynamics simulations," Sci. Rep. **14**, 25496 (2024).
[38]M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys. **378**, 686–707 (2019).
[39]E. Ang and B. F. Ng, "Physics-informed neural networks for flow around airfoil," in *AIAA Scitech 2022 Forum* (AIAA, 2022).
[40]H. Eivazi, M. Tahani, P. Schlatter, and R. Vinuesa, "Physics-informed neural networks for solving Reynolds-averaged Navier–Stokes equations," Phys. Fluids **34**, 075117 (2022).
[41]Z. Mao, A. D. Jagtap, and G. E. Karniadakis, "Physics-informed neural networks for high-speed flows," Comput. Methods Appl. Mech. Eng. **360**, 112789 (2020).
[42]X. Jin, S. Cai, H. Li, and G. E. Karniadakis, "NSFnets (Navier-Stokes Flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations," J. Comput. Phys. **426**, 109951 (2021).
[43]A. D. Jagtap, Z. Mao, N. Adams, and G. E. Karniadakis, "Physics-informed neural networks for inverse problems in supersonic flows," J. Comput. Phys. **466**, 111402 (2022).
[44]M. Baldan, P. Di Barba, and D. A. Lowther, "Physics-informed neural networks for inverse electromagnetic problems," IEEE Trans. Magn. **59**, 7001705 (2023).
[45]N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, "On the spectral bias of neural networks," in *International Conference on Machine Learning* (PMLR, 2019), pp. 5301–5310.
[46]M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," in *Advances in Neural Information Processing Systems* (Curran Associates, 2020), Vol. 33, pp. 7537–7547.
[47]K. McLaren, "A numerical and experimental study of unsteady loading of high solidity vertical axis wind turbines," Ph.D. thesis (McMaster University, 2011), see http://hdl.handle.net/11375/11096.

[48]ANSYS, Inc., *ANSYS® Free Student Fluent, Release 25.1* (ANSYS, Inc., 2025).
[49]F. Dicech and L. Parussini, "A multi-fidelity reduced-order model to quantify aerodynamic forces on a vertical-axis wind turbine with uncertain rotational speed," in UNCECOMP Proceedings, 2025, see https://2025.uncecomp.org/proceedings/pdf/21079.pdf.
[50]SU2 Foundation, *SU2 Version 8.3.0 "Harrier"* (SU2 Foundation, 2025).
[51]A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., 2019), Vol. 32, pp. 8024–8035.
[52]F. Dicech, K. Gkaragkounis, L. Parussini, A. Spagnolo, and H. Telib, "Integration of multi-fidelity methods in parametrized non-intrusive reduced order models for industrial applications," J. Comput. Sci. **85**, 102511 (2025).
[53]Q. Zhang, X. Wang, D. Yang, and W. Zhang, "Data-driven prediction of aerodynamic noise of transonic buffeting over an airfoil," Eng. Anal. Boundary Elem. **163**, 549–561 (2024).
[54]Q. Zhang, K. Zuo, W. Zhang, and Z. Dou, "Prediction of transonic buffet aerodynamic noise using multi-expert feature fusion network," AIAA J. **63**, 1–5 (2025).