



**Politecnico
di Torino**

ScuDo

Scuola di Dottorato - Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation

Doctoral Program in Pure and Applied Mathematics (38th cycle)

A Regularized Deep Learning Framework for CFD Surrogate Modeling with Geometric Variability

By

Laura Balasso

Supervisor(s):

Dr. Alaia Alessandro

Prof. Berrone Stefano

Politecnico di Torino

2026

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Laura Balasso
2026

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

Abstract

The analysis of complex physical systems in industrial environments relies on high-fidelity numerical simulations, which are often computationally prohibitive, particularly in many-query scenarios such as design optimization, parametric studies, and uncertainty quantification. This limitation is further exacerbated by geometric variability, as each configuration typically requires the generation of a dedicated computational mesh. These challenges motivate the development of fully data-driven surrogate modeling strategies capable of leveraging available simulation data while remaining flexible with respect to both geometry representation and discretization.

This thesis investigates deep learning-based surrogate models for predicting physically relevant quantities in computational fluid dynamics (CFD) problems characterized by geometric variability. Many existing approaches exhibit practical limitations, as they often rely on a shared reference mesh or assume the availability of explicit geometric parameterizations, restricting their applicability in real-world industrial scenarios.

The objective of this work is to develop a predictive framework capable of estimating fluid dynamic quantities of interest directly from geometrically varying configurations. By avoiding reliance on task-specific parametrizations, the proposed approach enables the integration of heterogeneous datasets and facilitates the reuse of previously generated designs, even when their underlying parametrization is unavailable or inconsistent.

In addition, the framework is designed to avoid dependencies on fixed geometric discretizations, such as volumetric or surface meshes. These representations often require costly and error-prone re-meshing or registration procedures to handle geometric variability across the dataset. Moreover, fixed discretizations introduce inherent resolution limitations, which may constrain the accuracy of the learned models. To address these issues, the proposed methodology adopts an implicit

representation of the geometry based on signed distance functions (SDFs), allowing the domain to be modeled as a continuous function rather than a discrete set of values defined on mesh nodes or elements. This continuous formulation enables the model to capture geometric features with potentially arbitrary resolution and to extract information directly from the implicit description, bypassing the need for mesh alignment or interpolation steps.

The approach is implemented using Implicit Neural Representations (INRs), where geometry and solution fields are encoded through two separate models operating on point samples drawn from the underlying domain. A feed-forward neural network is subsequently trained to learn a mapping between the corresponding latent spaces, effectively linking the geometric representation to the associated physical solution. This formulation enables mesh-independent predictions and allows the evaluation of the predicted fields at arbitrary spatial locations within the computational domain. Moreover, the modularity of the framework allows to predict a wide range of outputs, including global integral quantities, such as drag coefficient, as well as local high-fidelity scalar and vector fields, such as pressure distribution, defined over the entire spatial domain.

A key contribution of this work is the introduction of a regularization strategy for the INR-based encoders aimed at preserving the geometric structure within the latent space. This is achieved by enforcing consistency between distances in the geometry space and the corresponding distances in the latent representation, thereby promoting a coherent organization of the encoded samples.

The proposed framework is validated on three CFD test cases, demonstrating that the introduced regularization improves both predictive accuracy and generalization capabilities of the model, while maintaining robustness with respect to geometric variability.

Contents

List of Figures	viii
List of Tables	xvi
1 Introduction	1
1.1 Literature Review	2
1.2 Problem Statement	6
1.3 Contributions	7
1.4 Thesis Overview	8
2 Grid-based Approach	10
2.1 U-Net Architecture	10
2.2 Prediction of Physical Fields	11
2.3 Experiment	14
2.4 Final Considerations	21
3 Deep Geometric Encoding	26
3.1 DeepSDF Model	27
3.2 Point Cloud Sampling	29
3.3 Training and Inference Procedure	30
3.4 Test Case	31

3.4.1	Wall Thickness Encoding	32
3.4.2	Gaussian Surface Deformations	34
3.5	Final Considerations	40
4	Learning Framework and Latent Space Regularization	42
4.1	Baseline Workflow	43
4.1.1	Encoding Blocks	43
4.1.2	Mapping Block	45
4.1.3	Simplified Workflow	45
4.1.4	General Considerations	50
4.2	Distance Preserving Regularization	51
4.2.1	Fixed Reference Distance Regularization	54
4.2.2	Pair-Wise Distance Regularization	56
4.2.3	Output Distance Regularization	59
5	Test Cases	61
5.1	Datasets	62
5.1.1	Vertical Axis Wind Turbine	62
5.1.2	Transonic RAE2822	67
5.1.3	DrivAerML	70
5.2	Results	74
5.2.1	Models Validation and Error Metrics	75
5.2.2	Vertical Axis Wind Turbine	76
5.2.3	Transonic RAE2822	86
5.2.4	DrivAerML	95
6	Conclusions	105
6.1	Summary of the Work	105

6.2	Key Results and Experimental validation	105
6.3	Limitations and future work	106
6.4	Strengths and Contributions	108
References		110
Appendix A PWD Scalability Analysis		116
Appendix B Tuning of Regularization Weight		118
Appendix C VAWT Radial Force Prediction		120
Appendix D RAE2822 Pressure Coefficient Predictions		123
Appendix E Training and Inference Time		126

List of Figures

2.1	Gyroid units with wall thickness (a) 0.0003, (b) 0.005 and (c) 0.01. .	15
2.2	Gyroid lattice.	15
2.3	Sampled combinations of wall thickness and pressure drop used for U-Net training (blue) and validation (red). Gray markers indicate unsteady cases excluded from the dataset.	16
2.4	(a) Velocity and (b) pressure fields inside the gyroid unit with thickness 0.0003 m and $\Delta p = 10$ Pa.	16
2.5	(a) Input and (b) target channels of a training slice.	18
2.6	U-Net training and test loss.	19
2.7	Relative masked prediction error on validation dataset for the four target fields, namely (a) U_x , (b) U_y , (c) U_z and (d) pressure.	20
2.8	Normalized U_x prediction corresponding to the median error on validation dataset.	21
2.9	Detail of normalized U_x prediction near the channel boundary. . . .	21
2.10	Normalized U_y prediction corresponding to the median error on validation dataset.	21
2.11	Normalized U_z prediction corresponding to the median error on validation dataset.	22
2.12	Normalized pressure prediction corresponding to the median error on validation dataset.	22
2.13	U-Net prediction of U_y field with a slight discontinuity due to pre-processing.	22

2.14	Predictions corresponding to the maximum error on the validation dataset for velocity components and pressure.	25
3.1	DeepSDF auto-decoder architecture.	28
3.2	Wall thickness sampling of DeepSDF training and validation dataset.	32
3.3	Relative SDF prediction error on (a) training and (b) validation dataset.	33
3.4	Predicted value of the SDF on surface for the five validation geometries.	35
3.5	Distribution of the predicted surface distance, expressed as the absolute SDF normalized by the corresponding wall thickness, for different wall thicknesses. Each box plot summarizes the distribution of the normalized $ SDF $ values over the surface for (a) training and (b) validation geometries.	36
3.6	Baseline gyroid geometry with control points for the Gaussian surface deformations highlighted in magenta.	38
3.7	SDF sampled around a deformed geometry.	38
3.8	Examples of geometries generated from the baseline gyroid by applying a Gaussian perturbation at randomly selected control points along the surface normal direction. The first row shows the resulting deformed geometries, while the second row reports the corresponding deformation fields used to generate them.	39
3.9	Training loss for the encoding model trained on gyroids with Gaussian surface deformations	39
3.10	Loss history for latent codes optimization.	39
3.11	Relative error on SDF predictions for (a) training and (b) validation dataset.	40
3.12	Predicted SDF on surface for the geometries associated to the (a) best, (b) median and (c) worst prediction error.	40

4.1	Overview of the complete workflow training procedure. The geometric encoder and the output encoder are first trained separately to learn compact latent representations of the input geometries and target fields. Once the latent spaces have been established, the mapping block is trained to predict the output latent code from the geometric latent code.	46
4.2	Overview of the complete inference workflow. An unseen SDF field is provided to the trained geometric encoder, which estimates the corresponding latent code through the optimization procedure. The optimized latent code is then provided to the trained mapping block, which predicts the latent representation of the associated field. Finally, the predicted latent code is passed to the trained output encoder, which reconstructs the target field at the desired spatial or temporal locations.	46
4.3	Architecture of simplified mapping block for (a) integral and (b) spatially / temporally distributed output.	48
4.4	Diagram of the simplified workflow (SWF) for (a) integral quantities of interest and (b) spatially or temporally distributed fields. Unlike the complete workflow, the mapping block directly predicts the target output rather than its latent representation. For distributed fields, the spatial or temporal coordinate is additionally provided as input, enabling point-wise prediction.	49
4.5	Correlation plot of geometric distance and latent codes distance. Each color represents the distance of training samples from a fixed reference sample.	52
5.1	Schematic depiction of the 2D computational domain used for the 3-bladed Darrieus VAWT.	62
5.2	Baseline NACA0015 (blue), and geometrical variations obtained by varying k_{thick} and k_{curv} (orange).	64
5.3	Thrust force (F_T) and radial force (F_R) acting on a rotating turbine blade.	65

5.4	Example of velocity field computed for a deformed airfoil geometry with thickness coefficient $k_{thick} = 0.8$ and curvature coefficient $k_{curv} = 0.016$. The field values are reported in their original physical scale, without normalization.	65
5.5	Power coefficient as a function of the deformation parameters for the (a) initial and (b) complete dataset.	66
5.6	(a) F_T and (b) F_R signals for each airfoil in	66
5.7	Training dataset for the geometric encoder.	67
5.8	Point cloud used for SDF computation of the airfoil with $k_{thick} = 1.4$ and $k_{curv} = 0.05$. The sampled points are colored according to their corresponding signed distance values, illustrating both the sampling distribution and the resulting SDF representation.	67
5.9	RAE2822 deformation modes used to generate geometric variations of the base airfoil.	68
5.10	Baseline RAE2822 and deformed airfoils in the training dataset. . .	68
5.11	Snapshots of the computational mesh for a geometric variation of the base RAE2922 in the training dataset.	69
5.12	Example of the pressure coefficient computed around a training airfoil and restricted to the inference domain.	70
5.13	Surface pressure coefficient for 40 geometrical variations of the base RAE2822 in the training dataset.	70
5.14	Parametric generation of geometry variants starting from the baseline DrivAer model. (a) Definition of morphing boxes and (b–d) representation of the 16 geometric design parameters. Reproduced from Ashton et al. [1] with author’s permission.	72
5.15	Examples of deformed geometries generated by applying the full set of 16 deformation parameters. The corresponding front views are shown in the first column, while the side views are reported in the second column.	73

5.16	Surface pressure distributions for six geometry configurations. Each configuration corresponds to a different combination of the 16 deformation parameters applied to the baseline vehicle geometry.	74
5.17	Example of point-cloud samples for a training geometry. (a) SDF sample and (b) target pressure sample, obtained as random subsample of surface mesh cell centers.	75
5.18	Relative SDF prediction error on (a) training and (b) validation samples.	77
5.19	Airfoil surface reconstructions 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. The two profiles almost completely overlap, indicating the high reconstruction accuracy even for the worst-case examples.	78
5.20	F_T relative prediction error on (a) training and (b) validation samples.	78
5.21	F_T true and predicted signal for (a) average case and (b) worst case in the training dataset.	79
5.22	F_T true and predicted signal for (a) average case and (b) worst case in the validation dataset.	79
5.23	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. . . .	80
5.24	Explained variance of the three PCA components computed on the three-dimensional latent codes obtained (a) without regularization and (b) with PWD regularization.	81
5.25	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.	82
5.26	Relative error on SDF predictions obtained with (a) FRD regularization and (b) PWD regularization.	82
5.27	Relative error on the prediction of the thrust force, F_T on the validation dataset obtained with the (a) FRD regularization, and (b) with PWD regularization.	83

5.28 Thrust force predictions obtained with regularized DeepSDF + NN, compared with single-fidelity and multi-fidelity POD + GP.	84
5.29 Relative SDF prediction error on (a) training and (b) validation samples.	86
5.30 Relative C_p representation error on (a) training and (b) validation samples.	87
5.31 Relative C_p prediction error on (a) training and (b) validation samples.	88
5.32 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.	89
5.33 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.	90
5.34 Distribution of the relative error on SDF reconstruction on validation dataset obtained with (a) FRD and (b) PWD regularization.	92
5.35 Distribution of the relative C_p error on the validation dataset obtained with (a) FRD and (b) PWD regularization.	92
5.36 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.	92
5.37 Distribution of the relative prediction error on C_p for the validation dataset obtained with (a) FRD and (b) PWD regularization.	93
5.38 Comparison of true and predicted C_p fields obtained with the non-regularized workflow. From top to bottom: best and median prediction cases. From left to right: ground-truth field, predicted field and absolute error.	93

5.39	Comparison of true and predicted C_p fields obtained with the PWD-regularized workflow. From top to bottom: best and median prediction cases. From left to right: ground-truth field, predicted field and absolute error.	94
5.40	Cumulative explained variance resulting from PCA on (a) non-regularized and (b) PWD-regularized latent space.	96
5.41	Cumulative explained variance resulting from PCA on PWD-regularized latent space after 100 training epochs.	97
5.42	Relative SDF prediction error on (a) training and (b) validation dataset.	98
5.43	Absolute value of the predicted SDF on surface for the geometries associated to the (a) median and (b) worst prediction error.	99
5.44	Training and validation loss of the C_D prediction model based on (a) non-regularized and (b) PWD-regularized geometric encoding. The shaded regions indicate the range of loss values across batches within each epoch. Their lower extent is clipped for visualization purposes to improve the visibility of the mean loss curves.	100
5.45	C_D relative prediction error on unseen geometries obtained with (a) non-regularized and (b) PWD-regularized geometric encoding.	101
5.46	True vs predicted C_D for unseen geometries obtained with (a) non-regularized and (b) PWD-regularized geometric encoding.	101
5.47	Training and validation loss of the surface pressure prediction model. The shaded regions indicate the range of loss values across batches within each epoch.	102
5.48	Relative pressure prediction error on unseen geometries.	102
5.49	Comparison of true and predicted surface pressure fields for three representative validation geometries. From top to bottom: best, median and worst prediction cases. From left to right: ground-truth field, predicted field and absolute error.	103
5.50	Relative $c_{D,p}$ computed from true and predicted surface pressure. Distribution for (a) training and (b) validation dataset.	104

5.51	$c_{D,p}$ computed from true vs predicted surface pressure for (a) training and (b) validation dataset.	104
A.1	Distances computation time and memory usage for an increasing number of samples, each one of size 16,384.	117
A.2	Latent distances computation time and memory usage for an increasing number of samples.	117
B.1	SDF prediction error and F_T prediction error for six values of regularization weight λ in VAWT regularized geometric encoder.	118
B.2	SDF prediction error for different values of regularization weight λ	119
B.3	C_p reconstruction error for different values of regularization weight λ	119
C.1	F_R relative prediction error on (a) training and (b) validation samples.	120
C.2	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.	121
C.3	Radial force predictions obtained with regularized DeepSDF + NN, compared with single-fidelity and multi-fidelity POD + GP.	122
D.1	Pressure coefficient prediction for an unseen geometry obtained with the complete workflow without regularization (upper row), with FRD regularization (middle 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).	124
D.2	Pressure coefficient prediction for an unseen geometry obtained with the complete workflow without regularization (upper row), with FRD regularization (middle 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).	125

List of Tables

2.1	Statistics of the relative masked L_2 error on the validation dataset. . .	19
5.1	SDF reconstruction errors. Boldface denotes the best result both in terms of mean validation error.	82
5.2	F_T prediction errors. Boldface denotes the best result both in terms of mean validation error and standard deviation.	85
5.3	SDF reconstruction errors. Boldface denotes the best result both in terms of mean validation error.	91
5.4	C_p reconstruction errors. Boldface denotes the best result in terms of mean validation error.	91
5.5	C_p prediction errors. Boldface denotes the best result in terms of mean validation error.	91
5.6	SDF prediction error statistics on training and validation dataset. . .	98
5.7	C_D prediction errors.	100
5.8	Surface pressure prediction error statistics on training and validation dataset.	102
C.1	F_R prediction errors. Boldface denotes the best result both in terms of mean validation error and standard deviation.	121
E.1	GPU training time of encoding models.	127
E.2	Elapsed time for each step of the prediction of an unseen sample. . .	127

Chapter 1

Introduction

The analysis of complex physical systems in industrial environments heavily relies on high-fidelity numerical simulations. In engineering fields such as aerospace, automotive and manufacturing, the underlying physical phenomena are typically described by systems of partial differential equations (PDEs), such as the Navier–Stokes equations in fluid dynamics. Since closed-form analytical solutions are generally unavailable, these governing equations are solved numerically on discretized spatial and temporal domains. Although numerical methods can provide highly accurate approximations of the physical quantities of interest, they are often computationally expensive. Single evaluations may require hours or days of computation, particularly when a fine spatial or temporal discretization is needed. In industrial workflows, simulations are rarely performed as single, standalone runs. They are usually embedded within iterative procedures including parametric design exploration, constrained optimization, sensitivity analysis and uncertainty quantification.

These iterative workflows requiring repeated simulations become even more demanding when the problem involves geometric variations of the computational domain. This is the case for problems like shape optimization, in which the computational mesh must be regenerated for each new configuration, further increasing the overall computational cost. Furthermore, automatic re-meshing is a non-trivial task that may introduce additional numerical errors, especially for complex, real-world geometries. Under these circumstances, managing large-scale high-fidelity simulation pipelines becomes impractical. Reducing the computational cost of many-query processes to sustainable levels is essential to enable their practical adoption in in-

dustry. This challenge motivates the study and development of efficient surrogate models that accelerate or completely substitute the numerical solver.

1.1 Literature Review

In industrial practice, one class of methods aims at accelerating solvers while preserving their numerical structure. Classical Reduced Order Models (ROMs), such as Proper Orthogonal Decomposition (POD) [2, 3] combined with Galerkin projection [4], construct a low-dimensional representation of the solution manifold from high-fidelity snapshots and solve a reduced system instead of the original large-scale discretization. To further decrease the computational burden in nonlinear problems, hyper-reduction strategies, such as the Discrete Empirical Interpolation Method (DEIM) [5], limit the evaluation of nonlinear terms to carefully selected degrees of freedom, significantly reducing the online cost. Projection-based ROMs provide approximations with rigorous theoretical foundations and well-established error bounds.

Alongside, recent industrial research has employed machine learning (ML) within existing solvers, significantly accelerating specific components of the numerical pipeline. In turbulence modeling for Reynolds-Averaged Navier–Stokes (RANS) simulations, neural networks have been used to replace or correct specific closures, such as Reynolds stress models [6], while the rest of the solver remains unchanged. Similarly, data-driven preconditioners [7, 8] and learned initial guesses [9–11] have been proposed to accelerate the convergence of large linear or nonlinear systems. Both projection-based ROMs and solver-integrated ML strategies can significantly reduce the computational cost of high-fidelity simulations while preserving a strong connection with the underlying governing equations. However, many of these approaches are strongly problem-dependent or rely on intrusive methods, i.e., methods requiring modifications to the underlying numerical solver. Their implementation often requires substantial modifications of the solver, which is often incompatible with industrial workflows based on commercial black-box software. Moreover, although these intrusive approaches can considerably reduce the execution time of simulations, they still rely on the iterative solution of a (reduced) numerical system, which limits their applicability in real-time scenarios.

In settings where modifying the solver is impractical, non-intrusive data-driven surrogate models provide a viable alternative. These models learn directly from high-fidelity solutions (obtained through numerical simulations or experimental measurements) and approximate the mapping between the parameters of the problem and the corresponding quantities of interest. Once trained, these models provide rapid predictions without the need to solve the governing equations during inference. Unlike intrusive approaches, fully data-driven models do not explicitly embed prior knowledge of the underlying physical laws. As a consequence, their predictions may not strictly satisfy the governing equations and their accuracy strongly depends on the quality and representativeness of the training data. Nevertheless, in many industrial settings they are often preferred due to their flexibility and ease of deployment. In fact, they can be developed independently of the specific application and completely replace the solver at inference time, enabling near-real-time evaluations.

For these reasons, there is a growing need to investigate non-intrusive strategies that combine accuracy, efficiency and robustness, while meeting the practical constraints of industrial applications. Among the classical non-intrusive techniques, POD combined with interpolation schemes such as Radial Basis Functions (RBF) or Gaussian Processes, remains widely used due to its interpretability and low computational cost. It is worth noting that these methods differ in their formulation, as POD-based interpolation approaches are typically deterministic, while Gaussian Process regression provides a probabilistic surrogate with uncertainty quantification capabilities. When dealing with geometric variations, POD-based approaches usually perform a modal decomposition of the solution and predict the coefficients from the geometric parameters of the problem by training an interpolation or regression model.

The implementation of these of approaches relies on two fundamental requirements, which significantly limit their applicability to industrial contexts. First, the assumption that the geometric parameters defining each configuration are known. In real-world settings, such information is often unavailable, which complicates the construction of a user-defined parametrization. Ideally, surrogate models should be able to leverage all available simulation data, including geometries generated at different times with heterogeneous methods or parameterizations, which may no longer be accessible or properly documented. Second, POD is inherently discretization-dependent and requires a reference mesh shared across all training geometries to perform the modal decomposition. The snapshot matrix construction relies on

meshes having the same number of nodes and consistent numbering. In practice, this condition can be difficult to satisfy in automated workflows, where datasets involving geometrical variations are often associated with independently generated meshes, which may differ in size and node numbering. When mesh coherence is not satisfied, registration techniques can be employed to map all geometries onto a common template. However, these procedures are difficult to fully automate and can introduce artifacts or localized errors, injecting noise into the training data and undermining the overall accuracy of the surrogate. Moreover, even in a setting in which a common discretization is available and the geometric parameters are known, the inherent linearity of POD constrains its applicability to a limited range of problems.

These limitations highlight the need for surrogate modeling strategies that go beyond classical POD–interpolation frameworks. In industrial scenarios, datasets often arise from heterogeneous design and simulation pipelines, where geometric variations are generated using different parametrizations and mesh generation procedures, leading to inconsistencies in both discretizations and geometric representations. In this context, there is a growing demand for approaches that are independent of a shared reference discretization and do not rely on the explicit availability of geometric parameters. Such methods would enable the exploitation of available simulation data, regardless of how geometries were generated or represented, while maintaining high predictive accuracy across complex and nonlinear regimes. Recent advances in Deep Learning (DL) have enabled the development of sophisticated surrogate models capable of handling geometric variations and capturing highly non-linear relationships.

Among these, Convolutional Neural Networks (CNNs) [12] have been widely used to predict solutions of PDEs on 2D domains with varying shapes, treating both the input domain and the target solution as image-like data. CNNs have proven their effectiveness across a variety of applications, including the prediction of pressure fields around airfoils [13, 14], stress fields [15, 16] and heat transfer [17]. Within the class of convolutional architectures, the U-Net model [18], originally developed for biomedical image segmentation, has been successfully adapted for predicting stress fields in solid mechanics [19] as well as 2D [20] and 3D [21] flow fields in fluid dynamics.

Generative Adversarial Networks (GANs) [22] have also been applied to generate realistic 2D fields [16, 23–26] and to structural optimization problems [27]. While these grid-based architectures are very effective in many scenarios, practical considerations arise when dealing with complex geometries or multi-scale phenomena. To accurately capture fine-grained details, the resolution of the Cartesian input grid should match that of the smallest features in the computational mesh, which increases the grid size. Higher input resolutions may require deeper network architectures, increasing the model complexity. On the other hand, coarser grids can introduce interpolation errors that become more pronounced in the presence of sharp gradients, curved boundaries or strongly varying scales.

In order to address these challenges, geometric deep learning techniques [28, 29] have been employed to work directly with unstructured data. Graph Neural Networks (GNNs) have been widely used to predict 2D [30, 31] and 3D [32, 33] PDE solutions on both structured [34] and unstructured [35] grids. While GNNs offer considerable flexibility in handling complex geometries, their performance depends strongly on the underlying graph topology and their computational cost can be high. Additionally, the irregular data structure poses challenges for scalability and parallelization of GNN-based models.

Implicit Neural Representations (INRs) offer a compelling alternative to traditional mesh-based approaches. Rather than relying on discrete grids, INRs operate on point-based data and represent continuous signals through neural networks that map spatial or temporal coordinates directly to the corresponding field values. These models can also encode 2D or 3D scenes through radiance fields [36], by learning occupancy functions [37] or Signed Distance Functions (SDFs) [38]. Unlike discrete representations, INRs provide a continuous and resolution-independent description of the data while maintaining a relatively compact memory footprint.

INRs can also be extended to represent entire families of functions by conditioning the network on a latent code associated with each instance. This latent vector, concatenated with the spatial coordinates, identifies the specific object or signal to which a point belongs, enabling the network to jointly learn the spatial mapping and a compact representation of the instance. For example, the DeepSDF model [38] learns latent representations of 3D shapes through their SDF, while [37] learns a distribution of occupancy functions. More recently, INRs have been explored for the prediction of PDE solutions. In [39], an unsupervised INR is trained using a

physics-informed loss to approximate parametric PDE solutions in 1D and 2D. In [40], INRs are employed in a mesh-free topology optimization framework to determine optimal material distributions. An end-to-end framework for predicting 2D and 3D flow fields using INRs is presented in [41], where implicit representations are learned for both the geometry—through its SDF—and the flow field, which are then mapped by a Multi-Layer Perceptron (MLP), highlighting the potential of INRs to jointly encode geometry and physical solutions in a continuous latent representation.

Similarly to INRs, Physics-Informed Neural Networks (PINNs) [42] are mesh-free models that predict continuous signals directly from spatial or temporal coordinates. They are designed to approximate solutions of PDEs [43, 44] and to address inverse problems [45–48] such as the identification of model parameters from limited observations. Rather than relying solely on data supervision, PINNs incorporate the governing physical laws into the training process by penalizing deviations from the differential equations and boundary conditions within the loss function. Despite their conceptual appeal, training PINNs over complex domains often remains challenging. In addition, the physics-based loss formulation must typically be tailored to the specific PDE system under consideration or explicitly coupled with the numerical solver. As a consequence, PINNs are strongly problem-dependent and are less suitable in purely data-driven settings or when only black-box solvers are available, which limits their applicability in many practical industrial scenarios.

1.2 Problem Statement

Overall, the existing literature provides a broad range of surrogate modeling strategies, each addressing different aspects of the problem. POD-based methods offer computational efficiency but rely on shared discretizations and explicit parametrizations. Grid-based deep learning models overcome the need for explicit geometric parametrizations by learning directly from geometry representations such as occupancy fields or signed distance functions, although they remain tied to structured grid discretizations. INRs alleviate this restriction by operating on point-wise representations of physical fields and geometries. However, these models lack an inherent notion of geometric distance or shape similarity, resulting in irregular representations and highly nonlinear mappings that undermine their generalization capabilities. As a result, there remains a need for surrogate modeling frameworks that simultaneously

avoid explicit geometric parametrization, operate independently of a shared reference discretization, and incorporate appropriate geometric regularization to ensure robust and consistent representations in realistic industrial workflows involving complex geometries.

This thesis aims to demonstrate that appropriately regularized INR-based surrogate models can provide a flexible and accurate framework for industrial CFD applications, addressing key practical challenges associated with geometric variability while enabling accurate predictions of unseen configurations for repeated simulation workflows.

1.3 Contributions

In this work, two complementary approaches are investigated on industrial test cases: a grid-based convolutional architecture and an INR framework. Furthermore, a novel regularization strategy for INRs is introduced, incorporating a notion of distance in the learned latent space to promote structure and consistency across geometrically varying instances. Despite their known limitations, grid-based architectures remain widely adopted in industrial research. However, most existing studies focus on two-dimensional benchmarks, while investigations on complex three-dimensional geometries remain relatively limited.

For this reason, the first part of this work is devoted to the evaluation of a 3D U-Net-based approach on an industrial test case, with the objective of evaluating its accuracy and practical constraints in realistic settings. The second part of this thesis builds upon the framework introduced in [41] for predicting the solution of a target PDE defined over a parametrized domain. In this approach, INRs are employed to encode both the geometry (through its SDF) and the associated PDE solution into latent spaces, which are subsequently mapped through a feed-forward neural network. In both approaches, no explicit knowledge of the underlying geometric parametrization is required, as the geometry is implicitly represented through its signed distance function. This representation allows both architectures to process the training geometries independently of the parametrization method used to generate them. Nevertheless, the grid-based model operates on a fixed Cartesian discretization of the SDF and of the associated solution fields, whereas the INR-based workflow relies on point-wise coordinate input and continuous function representation. As a

consequence, the latter is not tied to a predefined grid resolution and can evaluate the predicted solution at arbitrary spatial (or temporal) locations.

Building upon the INR-based workflow, we introduce a novel regularization strategy aimed at structuring the learned latent representations by enforcing a geometry-aware consistency in the latent space. Standard INR-based encoders learn a latent space that lacks a meaningful notion of distance. Namely, distances between latent codes are not constrained to reflect distances in the original geometric or solution space. Such lack of structure may result in unnecessarily complex mappings between the geometry and solution latent spaces, affecting the smoothness and generalization capabilities of the learned operator. In order to address this issue, we present two regularization methods that enforce a geometrically consistent latent structure by introducing a modified loss function. The first method forces the latent codes to maintain a distance from a given reference which is proportional to the geometric distance between the associated shapes. The second method strengthens and refines the first by preserving the pair-wise distances between all the couples of training samples.

The INR-based workflow, both with and without regularization, is tested on three fluid dynamics problems. The first problem concerns the prediction of forces acting on a blade of a vertical axis wind turbine. The second, pertains to the prediction of the 2D pressure coefficient field around an airfoil. The third problem involves the prediction of drag coefficient and 3D pressure field on the surface of a road-car with 16 varying geometric features. All experiments demonstrate that the proposed regularization induces a geometrically coherent structure in the learned latent space and substantially enhance the accuracy and generalization of the mapping between the geometric and solution encoding.

1.4 Thesis Overview

The remainder of this thesis is structured as follows:

Chapter 2 reviews the U-Net architecture and evaluates its performance on a 3D internal fluid dynamics problem. The quantities of interest are the velocity and pressure fields inside a gyroid lattice heat exchanger with varying wall thickness.

The 3D fields, interpolated on a Cartesian grid, are divided into 2D slices, which are predicted starting from the corresponding 2D SDF field.

Chapter 3 provides a review of the DeepSDF model and presents the outcomes of the geometric encoding applied to gyroid structures with Gaussian surface deformations.

Chapter 4 describes the structure and training strategy of the baseline INR-based workflow and introduces the two regularization methods.

Chapter 5 presents the test cases and datasets used to evaluate the proposed framework and reports the corresponding experimental results.

Chapter 6 presents the conclusions of the thesis, including a summary of the proposed approach, the key results obtained, as well as a discussion of its limitations and possible directions for future work.

Chapter 2

Grid-based Approach

Grid-based convolutional architectures have been extensively adopted for surrogate modeling of parametrized PDE solutions. Their ability to process spatially distributed data on structured grids has led to successful applications in the prediction of physical fields over parametrized geometries [13, 15, 17, 14, 20, 21, 19]. At the same time, these methods rely on representing both the geometry and the solution on a common Cartesian discretization, making the framework dependent on a fixed spatial resolution and requiring interpolation of the original simulation data.

This chapter presents a grid-based surrogate modeling framework based on the U-Net architecture [18] and evaluates its performance on a three-dimensional internal flow test case involving gyroid lattice structures with varying wall thickness. The proposed workflow employs the Signed Distance Function (SDF) as an implicit representation of the geometry and predicts the corresponding velocity and pressure fields on two-dimensional slices extracted from Cartesian volumetric grids. Besides assessing the predictive capabilities of the approach, the chapter discusses the practical limitations associated with fixed-grid representations, which motivate the development of the point-wise framework introduced in the following chapters.

2.1 U-Net Architecture

Initially developed for the segmentation of biomedical images, U-Net [18] is an image-to-image fully convolutional architecture designed to process data defined on Cartesian (pixel) grids. The network operates on multi-channel tensors, where the

spatial dimensions represent the discretized domain and the channels correspond to different variables or features. In contrast to standard CNNs, U-Net learns a mapping between input and output fields while preserving their spatial structure through skip connections, i.e., direct links between corresponding encoder and decoder layers. This makes it particularly suitable for tasks in which both the input and the target are spatially distributed quantities, such as image segmentation, super-resolution (i.e., the reconstruction of high-resolution fields from low-resolution inputs) or the prediction of physical fields. The U-Net architecture is composed of a contracting path, in which the grid size is progressively reduced while the number of feature channels is increased, and an expansive path, in which the spatial resolution is progressively recovered and the number of channels is reduced until matching the original input size. The contracting path is composed of a sequence of down-sampling blocks, each consisting of two unpadded convolutional layers followed by a rectified linear unit (ReLU) activation and a max-pooling operation. At each subsequent block, the number of feature channels is doubled while the spatial resolution is progressively reduced, until the bottleneck layer with low spatial resolution and a high number of channels is reached. The expansive path mirrors the contracting one and is composed of a sequence of up-sampling blocks that progressively increase spatial resolution while halving the number of feature channels. To recover fine-grained spatial information lost in the down-sampling process, U-Net introduces skip connections between corresponding blocks of the two paths. Specifically, the feature maps produced in the contracting path are concatenated with those of the expansive path at the same resolution level, providing high-resolution information that improves the localization of spatial structures in the output. Each block in the expansive path performs an up-convolution, the concatenation with the corresponding feature map from the contracting path and two convolutions.

2.2 Prediction of Physical Fields

The U-Net architecture is particularly suitable for the prediction of spatially distributed physical quantities in presence of geometric variations, as its encoder–decoder structure preserves spatial information while enabling dense output predictions over regular grids. The input geometry can be provided to the model through its occupancy field or signed distance function (SDF), without requiring explicit knowledge

of the underlying geometric parametrization. Furthermore, the multi-channel data structure allows for the prediction of multiple fields for each configuration. Although the original U-Net architecture is designed for data defined on two-dimensional Cartesian grids, several works, including Chen et al. 2019 [49] and Jacob et al. 2021 [21], have extended this approach to three-dimensional problems, either by introducing fully three-dimensional convolutional operations or by processing the 3D grid as a sequence of 2D slices. In this work, the second strategy is adopted, and the volumetric data is decomposed into slices which are processed independently by a 2D network. While fully 3D convolutions could capture volumetric correlations more effectively, they typically entail a significantly higher computational and memory cost, which may limit their practical applicability for high-resolution grids.

In this work, the input geometry is provided to the model through its SDF. The SDF associated to a given closed manifold $\Gamma = \partial\Omega \subset \mathbb{R}^n$ ($n = 2, 3$) is a continuous function $SDF : \mathbb{R}^n \rightarrow \mathbb{R}$ 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:

$$SDF(\mathbf{x}) = \begin{cases} \text{dist}(\mathbf{x}, \Gamma), & \mathbf{x} \in \Omega, \\ -\text{dist}(\mathbf{x}, \Gamma), & \mathbf{x} \notin \Omega, \\ 0 & \mathbf{x} \in \partial\Omega, \end{cases} \quad (2.1)$$

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

Given a training dataset consisting of N_{train} 3D geometries with associated volumetric fields, a common rectangular bounding box is selected and overlaid with a Cartesian grid of size $N_x \times N_y \times N_z$, on which the fields of interest are interpolated. A dummy constant value is assigned to grid cells where the physical fields are not defined. On the same grid, the SDF is computed for each configuration. This results in N_{train} SDF fields (representing the training geometries) and the corresponding physical fields, all defined on the same Cartesian grid. At this stage, the 3D grids are split along the z -direction into N_z two-dimensional slices, resulting in $N_{\text{train}} \cdot N_z$ training samples. Computing the SDF and the physical fields on the full 3D domain before performing this splitting allows the 2D model to partially retain the three-

dimensional effects. Nevertheless, each slice is treated as an independent training sample within the optimization process.

During the training, a map is learned between the SDF slices and the associated fields. If the problem involves non-geometric parameters, such as varying operating conditions, additional parameters are passed as input values to the model through an additional constant channel. Most existing approaches that employ U-Net-like architectures for the prediction of spatial fields, train the model by minimizing the L_1 loss,

$$\mathcal{L}_{L_1} = \frac{1}{N} \sum_{i=1}^N |\hat{y}_i - y_i|, \quad (2.2)$$

where y_i and $\hat{y}_i$ denote the reference and predicted values, respectively, and N is the total number of training samples over which the loss is evaluated.

However, computing the L_1 loss over the entire Cartesian grid is not appropriate when the target field is only defined over a subset of the spatial domain. In particular, in fluid dynamics applications involving internal or external flows, the region where the solution is defined depends on the geometry and may occupy only a fraction of the computational grid. When the training geometries vary significantly along the slicing direction, the corresponding 2D slices exhibit strong variability in the region where the target field is defined. In some slices, the field is available over most of the grid cells, whereas in others it is only defined over a small portion of the domain. In these cases, the regions where the field is not defined disproportionately contribute to the overall loss. As a result, the model tends to prioritize error minimization in these regions, at the expense of accurately learning the field structure in the smaller regions where the target field is actually defined. Since a constant arbitrary value is assigned to the target field in those regions where the field is undefined, a bias is introduced toward smoothed or constant predictions also in the regions where the field is actually defined, thus compromising the accuracy. In this work, a weighted loss is introduced to address this issue. Without losing any generality, in the following, we assume that the physical fields are defined only inside the geometry (i.e. internal flows). However, the proposed approach can be easily extended to the opposite case (i.e. external flows). A mask can be defined to filter out the cells outside the geometry as follows:

$$w(x) = \begin{cases} 1 & \text{if SDF}(x) \leq 0 \\ 0 & \text{if SDF}(x) > 0. \end{cases} \quad (2.3)$$

Then, the modified loss function is defined as:

$$\mathcal{L}_{L_1} = \frac{1}{\sum_{x \in \Omega} w(x)} \sum_{x \in \Omega} w(x) |\hat{y}(x) - y(x)|, \quad (2.4)$$

where Ω is the Cartesian domain, $y(x)$ is the target quantity and $\hat{y}(x)$ its the predicted value. Note that, with the particular choice of the weighting function in 2.3, the value at the denominator simply gives the number of cells inside the geometry (where the field is defined). This modified loss corresponds to a masked mean absolute error (MAE), where the error is computed only over the physically valid region of the domain, as defined by the geometry. This enables the training to focus on the physically meaningful region, while ignoring the artificially extended parts. Furthermore, using the same error metric during inference yields a more accurate error estimation.

2.3 Experiment

The approach described in Section 2.2 has been employed for the prediction of volumetric fields in an internal thermo-fluid problem. The test case considers fluid flow through a Triply Periodic Minimal Surface (TPMS) lattice structure with a gyroid topology. Gyroid geometries are characterized by smooth curvature and a high surface-to-volume ratio, properties that are beneficial for convective heat transfer. In addition, the absence of sharp corners reduces stress concentrations, making gyroid TPMS structures attractive for lightweight and efficient heat exchangers. The analysis of the flow within a single gyroid unit cell is therefore of particular interest from an engineering perspective.

The gyroid geometry is first defined implicitly through the periodic scalar function

$$f(x, y, z) = \sin(x) \cos(y) + \sin(y) \cos(z) + \sin(z) \cos(x). \quad (2.5)$$

The zero level set of this function defines the reference gyroid surface. The SDF is then computed with respect to this surface, and geometries with different wall thicknesses are generated by extracting iso-surfaces of the SDF at prescribed distance values. In this work, the wall thickness therefore refers to the prescribed SDF offset used to generate the geometry, expressed in meters. It should not be interpreted as the local distance between opposite solid walls or between adjacent fluid channels.

Figure 2.1 illustrates the gyroid unit cell for different wall thickness values, while Figure 2.2 shows the extended lattice obtained by periodic repetition of the unit cell. The simulation of the flow through the gyroid unit cell is performed using the `buoyantPimpleFoam` solver from the open-source CFD software `OpenFOAM` [50]. to solve the laminar and incompressible Navier-Stokes equations over an unstructured mesh. Periodic boundary conditions are imposed in all directions except for the streamwise direction x , where a pressure difference Δp is applied.

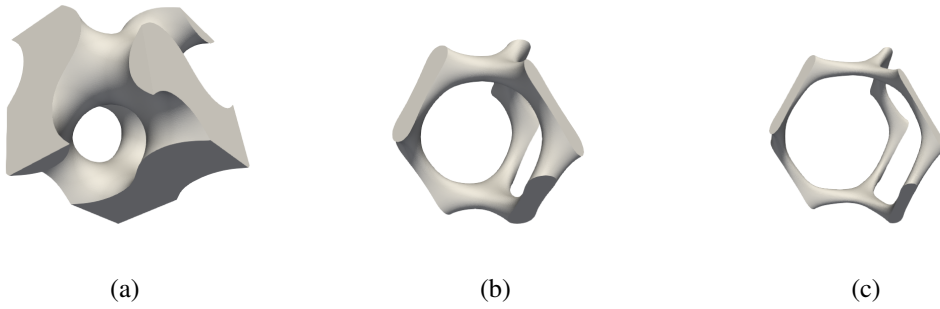


Fig. 2.1 Gyroid units with wall thickness (a) 0.0003, (b) 0.005 and (c) 0.01.

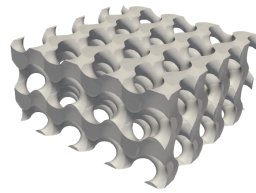


Fig. 2.2 Gyroid lattice.

The problem depends on two parameters: a physical parameter, namely the imposed pressure drop Δp , and a geometric parameter, namely the wall thickness of the gyroid structure. The simulation described above is therefore repeated for different combinations of these parameters. From each solution, the velocity field and

the pressure field are extracted. Figure 2.4 shows an example of the two volumetric fields for a gyroid unit with wall thickness 0.0003 m, obtained by imposing $\Delta p = 10$ Pa. The parameter values are sampled on a regular grid in the two-dimensional parameter space, as shown in Figure 2.3. Simulations characterized by small wall thickness values and large Δp , corresponding to transitional Reynolds numbers, failed to reach a steady state and were therefore excluded from the dataset. The 115 steady solutions are split into 80% training and 20% validation. In this chapter, the term *validation dataset* denotes the subset of unseen geometries used to evaluate the reconstruction accuracy of the learned geometric representation after training.

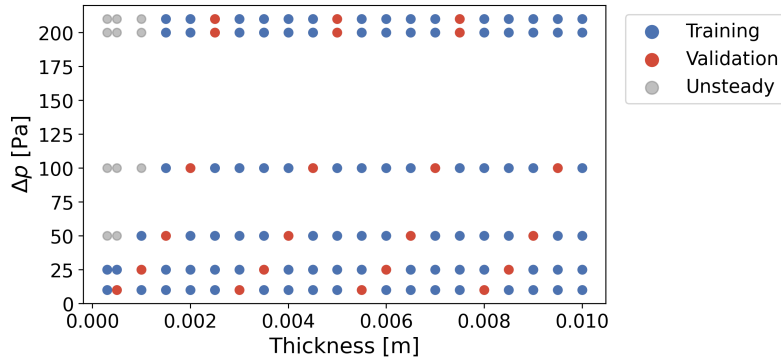


Fig. 2.3 Sampled combinations of wall thickness and pressure drop used for U-Net training (blue) and validation (red). Gray markers indicate unsteady cases excluded from the dataset.

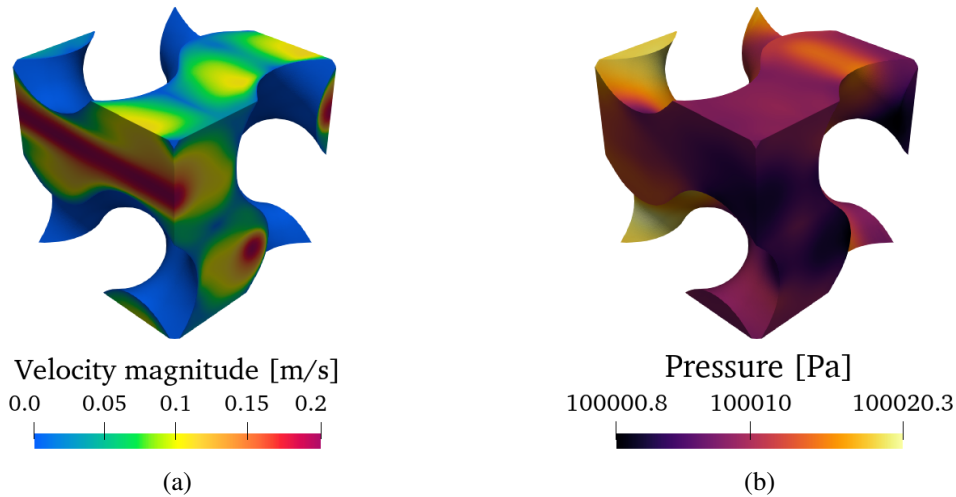


Fig. 2.4 (a) Velocity and (b) pressure fields inside the gyroid unit with thickness 0.0003 m and $\Delta p = 10$ Pa.

The objective is to train the U-Net model described in Section 2.2 to predict the velocity and pressure fields inside the gyroid channels for different values of wall thickness and Δp . To this end, each volumetric field in the dataset is interpolated on a 3D Cartesian grid of size $128 \times 128 \times 128$. A constant value of zero is assigned to the grid cells outside the channels, where the fields are not defined. On the same grid, the SDF is computed for each wall thickness configuration. The resulting discrete 3D tensor is then reshaped into a sequence of two-dimensional slices along one axis, which are used as input samples for the model. This slicing operation does not introduce any additional interpolation error, as it corresponds to a reorganization of the same discretized representation. Before training, the pressure fields were centered by subtracting their mean value. Subsequently, pressure and each velocity component were independently normalized by the maximum absolute value of the corresponding quantity computed over the dataset. The U-Net architecture described in Section 2.1 is employed with 32 feature channels in the first convolutional layer, which are progressively increased along the contracting path. Each down-sampling block consists of a convolution with stride two followed by batch normalization and a leaky ReLU activation. In the expansive path, the spatial resolution is restored through bilinear upsampling followed by convolution, batch normalization and ReLU activation. The model is trained with two input channels and four target channels. The input channels correspond to the SDF and a constant field containing the value of Δp , while the target channels correspond to the three velocity components and the pressure. Figure 2.5 shows an example of input and target channels for a training slice.

The model was trained on 93 simulations, each split into 128 slices. During training, 20% of the 11,904 slices were randomly selected and excluded from the optimization process in order to monitor the test loss. The model was trained for 1200 epochs with an initial learning rate of $1 \cdot 10^{-6}$ and a stepwise constant schedule that multiplies the learning rate by 0.5 every 400 epochs, to progressively stabilize optimization in later stages of training. The evolution of the training and test losses is reported in Figure 2.6. The curves exhibit a clear stepwise behavior consistent with the learning rate schedule. At 400 epochs, a noticeable decrease in both training and test losses is observed, corresponding to the first reduction of the learning rate. A further change in trend occurs around 800 epochs, where the training loss continues to decrease more significantly, while the test loss stabilizes in terms of its mean value, with a reduction in variability but no further significant improvement in its

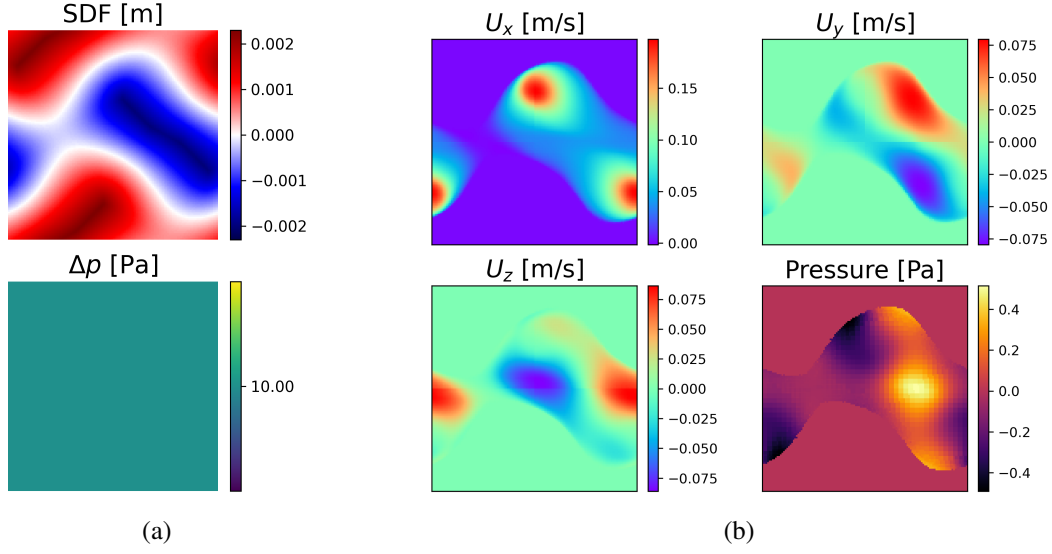


Fig. 2.5 (a) Input and (b) target channels of a training slice.

average level. Overall, the training process converges by 1200 epochs, where no further meaningful improvements are observed.

The trained model was then used for the prediction of unseen slices. Recall that the validation set consists of slices belonging to entire simulations that were not seen during training. Given the weight defined in Equation 2.3, the relative prediction error for a given target field is computed for each slice as follows:

$$\text{Relative masked } L_2 \text{ error} = \frac{\|(\text{true} - \text{pred}) \odot w\|_2}{\|\text{true} \odot w\|_2} = \frac{\sqrt{\sum_i w_i (\text{true}_i - \text{pred}_i)^2}}{\sqrt{\sum_i w_i (\text{true}_i)^2}}, \quad (2.6)$$

where true and pred denote the normalized ground-truth and predicted fields, respectively. This computation only considers the cells where the target fields are defined, i.e., the cells inside the gyroid channels. Figure 2.7 shows the distribution of the relative masked error in the validation dataset for each target field, while Table 2.1 reports the error statistics.

The model achieves consistent prediction accuracy across the velocity components, while slightly larger errors are observed for the pressure field. Figures 2.8, 2.10, and 2.11 show that, in the average-case scenario, the predicted velocity components capture the main flow structures with reasonable accuracy. Most of the

Table 2.1 Statistics of the relative masked L_2 error on the validation dataset.

Target field	Min [-]	Max [-]	Median [-]	Mean [-]
U_x	0.0133	0.2533	0.0319	0.0403
U_y	0.0146	0.5627	0.0517	0.0806
U_z	0.0151	0.4815	0.0524	0.0771
p	0.0491	0.3774	0.0902	0.1020

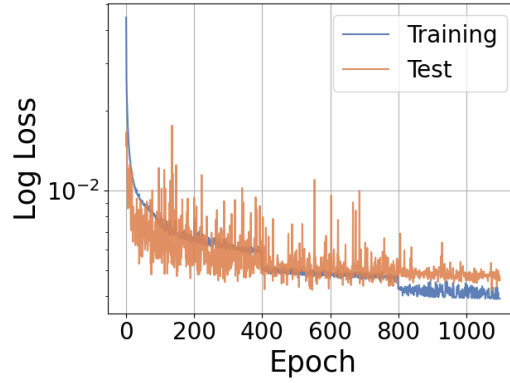


Fig. 2.6 U-Net training and test loss.

discrepancies are concentrated near the boundaries of the channels. This behavior is consistent with the presence of strong gradients near the walls. Additionally, in these regions the original computational mesh is significantly finer than the Cartesian grid used for interpolation, which may further contribute to the observed discrepancies. This phenomenon is clearly visible in Figure 2.9, which shows a zoomed view of the velocity field near the channel boundary, highlighting the limited resolution of the Cartesian grid and the artifacts introduced by the interpolation from the original CFD mesh. Figure 2.12 shows that the average pressure prediction is reasonably accurate. However, larger discrepancies are observed near the channel boundaries compared to the velocity field. In these regions, the reference pressure values are likely affected by interpolation errors introduced when projecting the solution onto the Cartesian grid, as can be observed in the ground-truth field shown in Figure 2.12. As a result, part of the observed error may be attributed to inaccuracies in the target data rather than to the model itself. This effect is less pronounced for the velocity field, probably because the velocity is always zero on the wall surface, which leads to a more robust interpolation in boundary regions. The maximum errors are relatively high for all target variables, indicating the presence of outliers. Figure 2.14 shows the worst-case predictions for each target field. For the first velocity component,

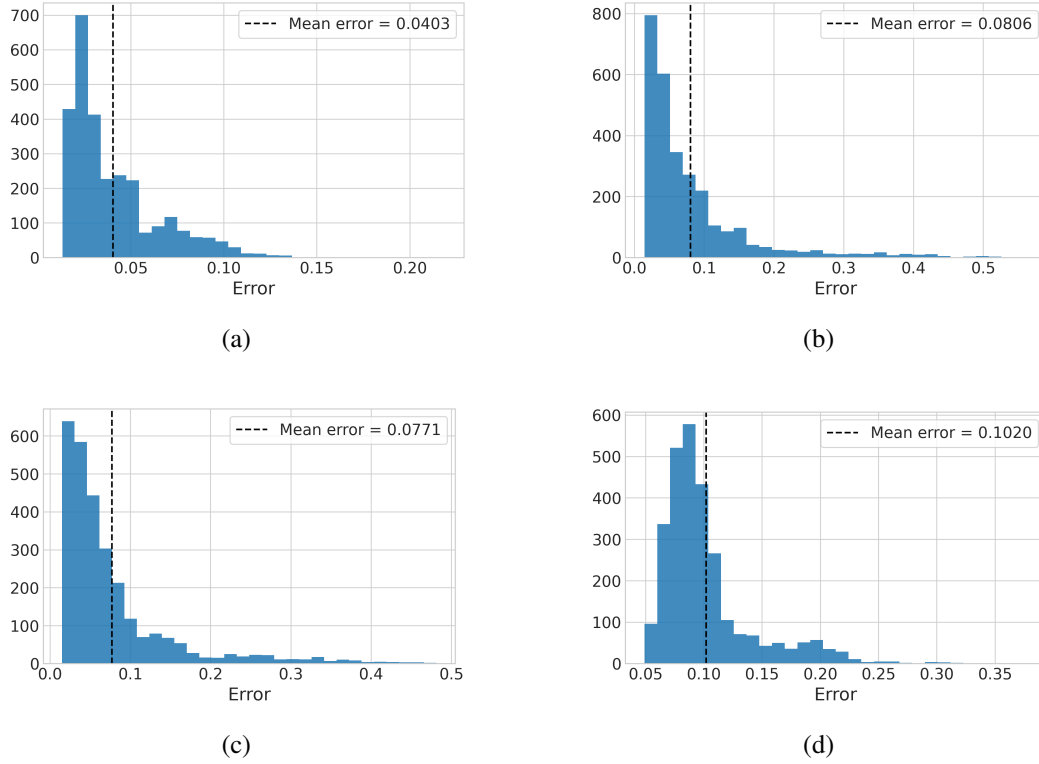


Fig. 2.7 Relative masked prediction error on validation dataset for the four target fields, namely (a) U_x , (b) U_y , (c) U_z and (d) pressure.

most of the error is again concentrated near the boundaries. For the remaining fields, the model shows reduced accuracy in predicting small channels characterized by small-scale flow structures and high gradients.

It is worth noting that the reference fields may exhibit slight discontinuities, as visible in Figure 2.13. These artifacts arise from the data preprocessing procedure: the simulations are periodic, but correspond to different phase shifts of the gyroid geometry for different wall thickness values. In order to map all configurations onto a common Cartesian domain, each solution is spatially repeated, which may introduce small mismatches at the interfaces between adjacent periodic copies. In contrast, the predicted fields appear smoother, as the convolutional operations promote local spatial smoothing and attenuates high-frequency artifacts.

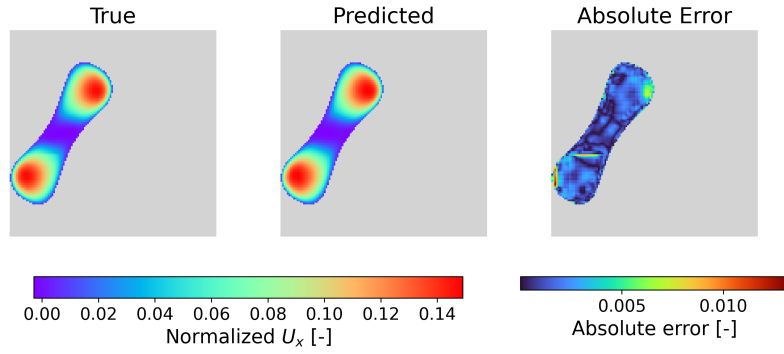


Fig. 2.8 Normalized U_x prediction corresponding to the median error on validation dataset.

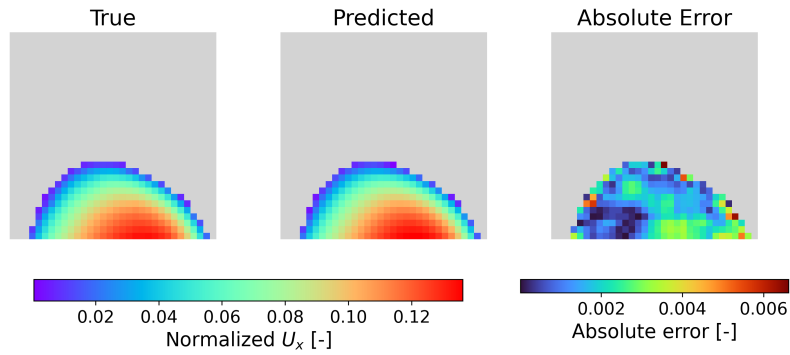


Fig. 2.9 Detail of normalized U_x prediction near the channel boundary.

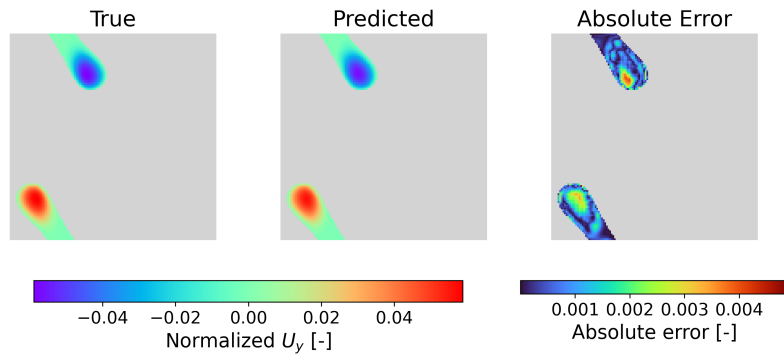


Fig. 2.10 Normalized U_y prediction corresponding to the median error on validation dataset.

2.4 Final Considerations

The results presented in this chapter show that the U-Net-based architecture provides a viable approach for the prediction of spatially distributed physical fields defined on parametrized geometries. The model is able to capture the main flow features,

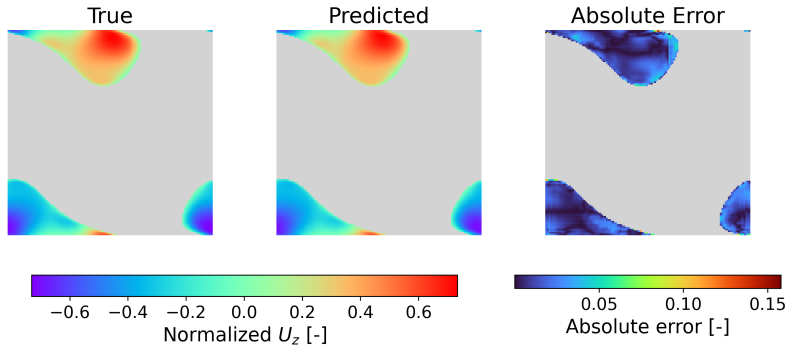


Fig. 2.11 Normalized U_z prediction corresponding to the median error on validation dataset.

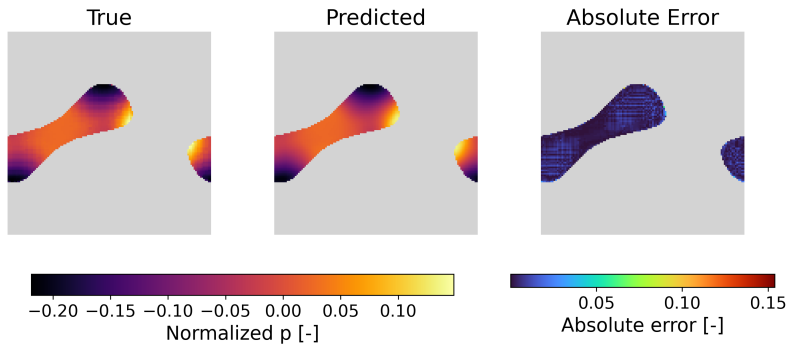


Fig. 2.12 Normalized pressure prediction corresponding to the median error on validation dataset.

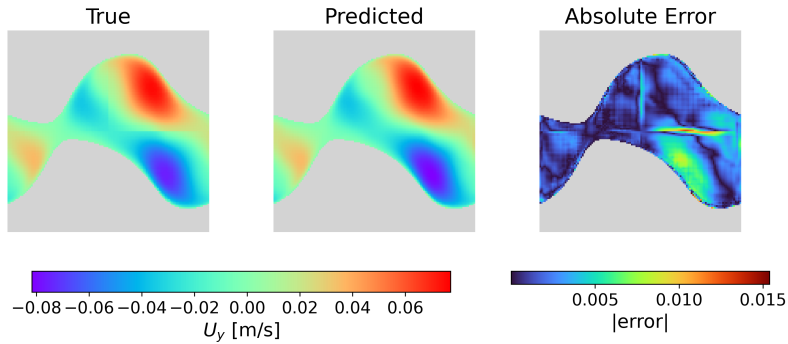


Fig. 2.13 U-Net prediction of U_y field with a slight discontinuity due to pre-processing.

achieving a moderate level of accuracy on average, particularly for the velocity field. However, the analysis of the error highlights the presence of large localized errors, especially near solid boundaries and in regions characterized by strong gradients and small-scale structures.

From a modeling perspective, the proposed framework naturally incorporates both geometric and non-geometric parameters. In particular, the geometry is implicitly represented by the SDF, which renders the model independent from a user-defined geometry parametrization. Additional physical parameters can be effectively introduced as constant input channels. Moreover, the use of a weighted loss function based on the SDF allows the model to focus the learning process on the physically relevant regions of the domain.

These observations are closely related to the grid-based nature of the adopted approach. Representing both geometry and physical fields on a common Cartesian grid enables the use of convolutional architectures and leads to an efficient learning framework. On the other hand, this representation introduces several limitations, including interpolation errors and a limited ability to capture small-scale features, particularly in the proximity of complex boundaries. Furthermore, the slice-wise representation of three-dimensional data may undermine the accurate reconstruction of 3D phenomena. An additional drawback of the grid-based approaches concerns the lack of flexibility in the spatial resolution. In order to better resolve localized regions of interest, such as boundary layers or areas with strong gradients, the resolution of the entire Cartesian grid must be increased. This not only leads to a higher memory footprint, but also to a significant increase in the complexity of the neural network. Since CNNs generally perform poorly with aggressive (transpose) convolutions, a large resolution would require deeper encoder–decoder structures and consequently a larger number of feature channels and skip connections. As a result, the computational cost grows more than linearly with the grid resolution. For these reasons, increasing the grid resolution was not considered a primary direction in this work, as the associated computational costs would outweigh the expected benefits within the scope of the proposed framework.

An alternative to uniformly increasing the grid resolution is to adopt non-uniform or adaptive grid constructions, which allow the computational effort to be concentrated in regions of interest. Such approaches can effectively capture fine-scale features in regions characterized by strong gradients, such as near-wall regions or boundary layers, while limiting the overall number of grid points. However, the generation of suitable adaptive discretizations remains a challenging task, particularly for complex industrial geometries, where mesh construction may require problem-dependent strategies and, in some cases, manual intervention. Moreover, adaptive or multi-resolution grids introduce additional challenges for surrogate modeling, as

the learning framework must handle spatial features occurring at different scales within the same domain. Developing models capable of consistently exploiting such heterogeneous spatial resolutions remains an open challenge. For these reasons, this work explores alternative representations, such as point cloud-based formulations, which provide a more flexible description of complex geometries and avoid the need for a predefined spatial discretization.

Non-Euclidean approaches, such as graph neural networks (GNNs), offer powerful alternatives for learning directly on unstructured domain discretizations and have demonstrated strong performance in various PDE-related tasks [30, 31, 33]. By explicitly leveraging the connectivity of the computational mesh, GNNs can naturally incorporate geometric and topological information and capture local physical interactions. However, their application to surrogate modeling across highly variable geometries introduces additional challenges. Since each geometry is associated with a different graph structure, achieving generalization across diverse topologies may require large and highly representative training datasets. Moreover, the message-passing operations underlying GNNs introduce additional computational and memory costs, as information propagation through neighboring nodes typically requires multiple layers and increases the size of the model. These aspects can become particularly relevant for large-scale CFD simulations involving high-resolution discretizations. Therefore, although GNNs represent a valid approach for physics-based learning, in this work we adopt a mesh-free learning strategy based on point-cloud representations, which provides a direct interface with CFD data and enables a unified treatment of diverse geometries without requiring explicit mesh connectivity.

In the next chapter, a point-wise approach based on implicit neural representations is introduced, in which both geometry and physical fields are modeled in a continuous way. This formulation aims at overcoming some of the aforementioned limitations of grid-based methods, providing a more flexible framework for handling complex geometries and high-resolution features.

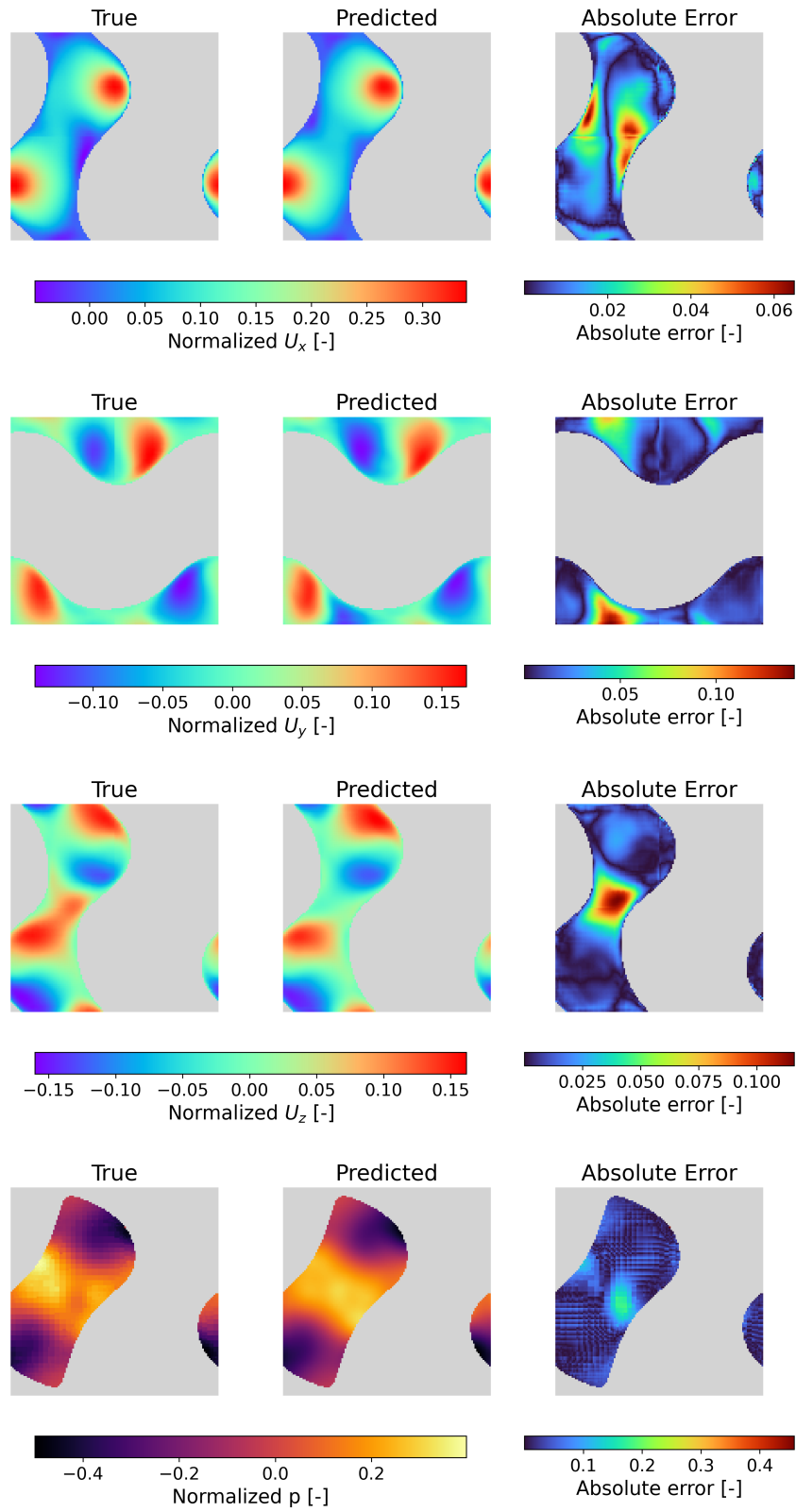


Fig. 2.14 Predictions corresponding to the maximum error on the validation dataset for velocity components and pressure.

Chapter 3

Deep Geometric Encoding

The limitations discussed in Chapter 2 highlight some intrinsic drawbacks of grid-based representations. In particular, the dependence on a fixed Cartesian discretization constrains the achievable resolution, introduces interpolation artifacts and makes the accurate reconstruction of small-scale features particularly challenging. Moreover, the reliance on a structured grid ties the model complexity directly to the chosen spatial resolution, leading to significant computational and memory costs when refinement is required. This lack of flexibility motivates the search for alternative representations that allow accuracy to be improved through flexible point sampling, rather than by uniformly increasing the resolution of a fixed grid.

In this context, point-wise approaches based on implicit neural representations (INRs) offer a viable alternative. These models learn a continuous mapping defined over the spatial domain, enabling the evaluation of the solution at arbitrary locations. Among these approaches, DeepSDF provides an effective framework for encoding complex geometries through a continuous signed distance function (SDF). In this work, the DeepSDF architecture is employed to encode both the geometry and the physical fields of interest into separate latent spaces. These latent representations are then mapped through a feed-forward neural network to approximate the relationship between geometry and the corresponding physical fields.

This chapter first reviews the DeepSDF model and then investigates its capability to encode geometric variations through dedicated experiments.

3.1 DeepSDF Model

DeepSDF, originally introduced by Park et al. [38], is an INR-based model that learns a latent representation for 2D and 3D shapes by encoding their SDF. Specifically, it represents a shape as a continuous function that maps spatial coordinates to signed distance values. This implicit representation of the geometry is not tied to a particular discretization or to a fixed resolution level as the SDF can be queried at arbitrary locations. The model is conditioned on a low-dimensional learnable latent code, which captures the underlying geometric variability across different instances. Importantly, this approach does not rely on a pre-defined parametrization of the geometry, as the latent codes are directly learned from data and provide a data-driven shape parametrization which can be employed in prediction tasks.

In order to train DeepSDF model for encoding a set of N shapes, the SDF defined in Equation 2.1 is computed on M points sampled in the physical space, both inside and outside the geometry:

$$X_i = \{\mathbf{x}_j, s_j\} : s_j = \text{SDF}_i(\mathbf{x}_j), \quad j = 1, \dots, M, \quad i = 1, \dots, N. \quad (3.1)$$

The architecture, represented in Figure 3.1, is a fully connected neural network $f_\theta(\mathbf{z}, \mathbf{x}) \approx \text{SDF}(\mathbf{x})$ 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. Each latent code $\mathbf{z}_i$ is associated with the i -th training shape. Given a query point $\mathbf{x}$ and the latent code $\mathbf{z}_i$, the network outputs the SDF value at $\mathbf{x}$ for the corresponding shape.

It is worth noting that, somewhat counterintuitively, the unknown latent code of each shape is provided as an input to the model. For this reason, the architecture is commonly referred to as an *auto-decoder*, to distinguish it from the standard encoder-decoder paradigm. For simplicity and consistency in terminology, in this work we will refer to this model as an *encoder* or encoding model.

A probabilistic approach is used to derive DeepSDF formulation. The posterior probability distribution over the latent codes $\mathbf{z}_i$, given the SDF sample X_i , can be written as

$$p_\theta(\mathbf{z}_i | X_i) = p(\mathbf{z}_i) \prod_{(\mathbf{x}_j, s_j) \in X_i} p_\theta(s_j | \mathbf{z}_i; \mathbf{x}_j), \quad (3.2)$$

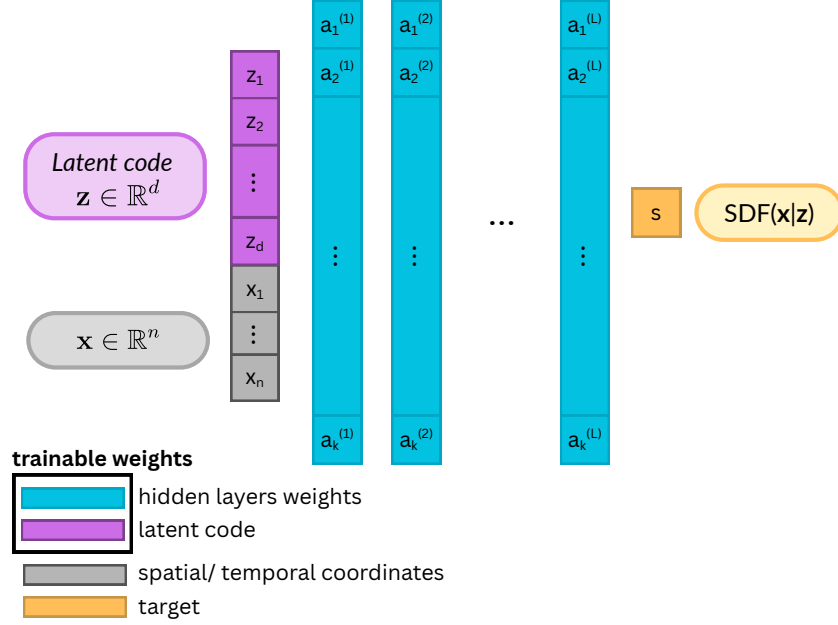


Fig. 3.1 DeepSDF auto-decoder architecture.

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

$$p_\theta(s_j|\mathbf{z}_i; \mathbf{x}_j) = \exp(-\mathcal{L}(f_\theta(\mathbf{z}_i, \mathbf{x}_j), s_j)), \quad (3.3)$$

where $\mathcal{L}(\cdot, \cdot)$ denotes a loss function that measures the discrepancy between the predicted SDF value and the ground truth. During training, the joint log-posterior over the dataset is maximized with respect to both the latent shape representations and the neural network parameters i.e.

$$\underset{\theta, \{\mathbf{z}_i\}_{i=1}^N}{\operatorname{argmin}} \sum_{i=1}^N \left(\sum_{j=1}^M \mathcal{L}(f_\theta(\mathbf{z}_i, \mathbf{x}_j), s_j) + \frac{1}{\sigma^2} \|\mathbf{z}_i\|_2^2 \right). \quad (3.4)$$

At inference time, the latent code associated with a new shape is estimated via maximum-a-posteriori (MAP) optimization, while keeping the learned network parameters θ fixed:

$$\hat{\mathbf{z}} = \underset{\mathbf{z}}{\operatorname{argmin}} \sum_{(\mathbf{x}_j, s_j) \in X} \mathcal{L}(f_{\theta}(\mathbf{z}, \mathbf{x}_j), s_j) + \frac{1}{\sigma^2} \|\mathbf{z}\|_2^2. \quad (3.5)$$

3.2 Point Cloud Sampling

It is worth noting that this encoding framework does not impose any constraint on the spatial distribution of the points $\mathbf{x}_j$ in the continuous 3D space, nor does it require a consistent sampling strategy across different training shapes, as each shape can be sampled independently. In this work, a non-uniform sampling strategy is adopted following the procedure proposed in the original DeepSDF formulation, with a higher density of points in the vicinity of the surface in order to better capture the zero level set, where fine-scale geometric details and local surface features are concentrated. More specifically, the total number of sampled points M is partitioned into two subsets, M_{volume} and M_{surface} , with $M_{\text{volume}} \ll M_{\text{surface}}$. The M_{volume} points are uniformly sampled within a simple bounding volume (either a box or a sphere) enclosing the geometry. The choice of this bounding region is not required to be conformal to the underlying shape, as the volumetric samples are sparsely distributed and mainly serve to complement the higher-density sampling near the surface. The M_{surface} points are instead sampled on the surface and subsequently perturbed by adding Gaussian noise, following the standard DeepSDF formulation. This results in a local sampling neighborhood around the zero level set, providing a richer learning signal in the vicinity of the surface. To account for different spatial scales, the surface samples are divided into three subsets, each perturbed with Gaussian noise characterized by a different standard deviation, following the official DeepSDF implementation. This multi-scale perturbation strategy is used as a simple heuristic to improve robustness to local geometric variability, rather than as a hyperparameter tuned for a specific application.

3.3 Training and Inference Procedure

From a practical standpoint, training the DeepSDF model involves the joint optimization of the network parameters and the latent codes associated with each training shape. Given a set of sampled points with corresponding SDF values, the network learns to map spatial coordinates and latent codes to their associated signed distance values. The network weights and the latent vectors are initialized randomly and during training they are both updated in order to minimize the SDF reconstruction error over the training dataset. The training objective defined in Equation 3.4 is implemented using an ℓ_1 loss on the SDF reconstruction, combined with a regularization term on the latent codes that penalizes their ℓ_2 norm. The resulting training loss can therefore be written as:

$$\mathcal{L}_{\text{DeepSDF}} = \sum_{i=1}^N \left(\sum_{j=1}^M |s_{i,j} - \hat{s}_{i,j}| + \lambda_{\text{norm}} \|\mathbf{z}_i\|_2^2 \right), \quad (3.6)$$

where $s_{i,j}$ and $\hat{s}_{i,j}$ are the ground-truth and predicted SDF values at point $\mathbf{x}_j$ for the i -th shape respectively, $\mathbf{z}_i$ is the associated latent code and λ_{norm} is the norm regularization weight.

At inference time, the goal is to estimate the latent code associated with a new shape from its SDF sample. Since the latent code is part of the input of the model, it cannot be directly inferred through a standard forward pass. Instead, it is treated as an unknown variable and estimated by solving an optimization problem. During this stage (latent code optimization), the weights previously learned during the training are kept fixed, and only the latent code is updated. The network takes as input the spatial coordinates of the point cloud and the (randomly initialized) latent code, and predicts the corresponding SDF values. The optimal latent representation is obtained by minimizing the loss function in Equation 3.6 only with respect to the latent code $\mathbf{z}$. This inference procedure requires solving an additional optimization problem for each new shape and therefore does not allow for strictly real-time predictions. However, the optimization of the latent code is significantly less computationally demanding than training the neural network, as only the latent codes are updated. Convergence is typically achieved in a very few iterations (see Appendix E). Note that a standard encoder-based architecture would provide a significantly faster inference procedure, as the latent representation would be obtained through a single

forward pass of the encoder. In contrast, the DeepSDF auto-decoder formulation requires iterative optimization of the latent code for each new shape, resulting in a more expensive inference procedure. However, the auto-decoder formulation does not require the latent codes to be generated by a shared encoder network. Instead, each latent vector is directly optimized together with the decoder parameters during training, providing additional freedom in organizing the latent representation. This additional degree of freedom can facilitate the learning of a compact representation of the shape distribution. Moreover, the implicit formulation of the decoder provides an additional advantage compared to architectures relying on fixed geometric discretizations. Since the SDF can be queried at arbitrary spatial locations, the representation is not tied to a predefined grid or point sampling strategy. This allows the spatial resolution to be increased locally, for example near the surface, without requiring a global refinement of the representation.

3.4 Test Case

The model described in Section 3.1 is here employed to encode the gyroid geometries introduced in Section 2.3. Although a publicly available implementation of DeepSDF served as an initial reference [51], the final implementation used in this work was extensively reworked and is not directly aligned with the original software in terms of experimental reproducibility.

In a first experiment, the DeepSDF model is trained on the gyroids geometries with varying wall thickness. The objective is to learn a latent representation of the gyroid shapes without access to the underlying geometric parameter, thereby evaluating the model’s ability to learn a meaningful data-driven parametrization. A second experiment is conducted on gyroid geometries with fixed wall thickness and Gaussian surface perturbations. The aim is to assess the ability of the model to encode this different type of geometric variability. This setting can be seen as a simplified approximation of practical engineering scenarios in which geometries are affected by localized imperfections, such as manufacturing tolerances or surface defects, which may influence the physical response of the system.

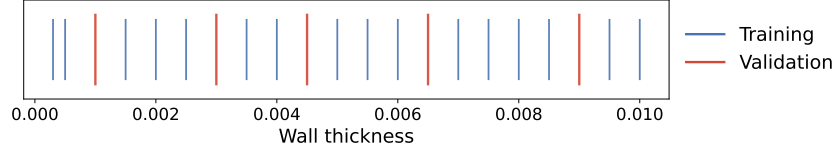


Fig. 3.2 Wall thickness sampling of DeepSDF training and validation dataset.

3.4.1 Wall Thickness Encoding

In this experiment, DeepSDF model is trained to encode the geometries described in Section 2.3, namely gyroid units with varying wall thickness. The resulting dataset of 21 geometries (three examples are shown in Figure 2.1) is divided into training and validation datasets as shown in Figure 3.2. For each train and validation geometry, 30,000 points are sampled in the 3D space following the sampling strategy described in Section 3.2. For each sampled point $\mathbf{x}_j$, the signed distance function is evaluated with respect to the corresponding geometry, yielding $s_j = \text{SDF}_i(\mathbf{x}_j)$, where the subscript i denotes the associated geometry. The point clouds corresponding to the selected validation geometries are not included in the training process in order to assess the model accuracy on unseen gyroid configurations.

A DeepSDF model with 6 hidden layers of size 512, ReLU activations and a final tanh activation was trained using the Adam optimizer for 3,000 epochs. The learning rate was initialized at $5 \cdot 10^{-4}$ and halved every 500 epochs using a stepwise constant schedule. The same scheduling strategy was adopted for the latent codes, with an initial learning rate of 10^{-3} . After training, the latent code optimization procedure described in Section 3.3 was applied independently to each validation geometry. During this inference step, the network parameters were kept fixed and only a shape-specific latent code was updated by minimizing the reconstruction error on the available SDF samples. This procedure resulted in a set of latent codes, each associated with a previously unseen validation shape.

To assess both the fitting capability of the model and its generalization performance to unseen configurations, the reconstruction error is evaluated on both the training and validation datasets. For the training set, the objective is to quantify how accurately the learned representation reproduces the shapes used during training, whereas the validation set is used to evaluate the ability of the model to represent previously unseen geometries. For the training set, the latent codes obtained during the training phase are used together with the corresponding sampled spatial points to

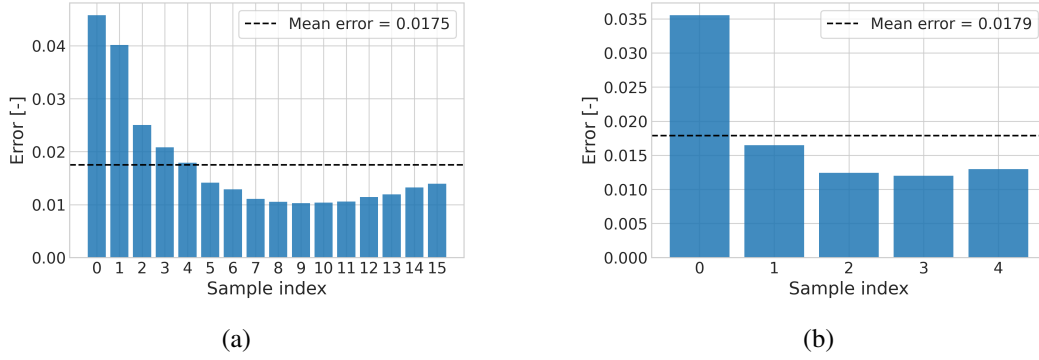


Fig. 3.3 Relative SDF prediction error on (a) training and (b) validation dataset.

reconstruct the SDF values. The reconstruction error is then computed by comparing the predicted and reference SDF values at the same locations used during training. For the validation set, a new latent code is first estimated for each unseen configuration through the latent code optimization procedure described above. The resulting latent code is then combined with the same query points used during this inference step to reconstruct the SDF and evaluate the reconstruction error.

The reconstructed SDF is compared to the ground-truth by computing a relative error defined as:

$$error_i = \frac{\|\mathbf{SDF}_i - \widehat{\mathbf{SDF}}_i\|_2}{\|\mathbf{SDF}_i\|_2}, \quad (3.7)$$

where $\mathbf{SDF}_i$ is a vector collecting the SDF values of the i -th shape evaluated at the points $\mathbf{x}_j$, while $\widehat{\mathbf{SDF}}_i$ denotes the corresponding prediction.

Figure 3.3 shows the relative error for each geometry in the training and validation datasets. The mean error remains below 2% in both cases, with the validation error being only 0.04 percentage points higher than the training error. These results indicate that the model is able to accurately represent the geometry in the latent space, including shapes that were not observed during training.

A further analysis was performed to assess the reconstruction error of the decoded validation shapes by evaluating the predicted SDF values at points sampled on the surface of the geometry. Since these points belong to the true zero level set, their reference SDF value is exactly zero. Therefore, the deviation of the predicted SDF from zero at the same locations provides an estimate of the distance between the

original surface and the reconstructed zero level set of the predicted SDF. In order to assess the reconstruction accuracy, the SDF was evaluated at all the vertices of the surface mesh of each validation geometry. Figure 3.4 shows the predicted SDF on surface for the five validation geometries. The maximum surface reconstruction error is on the order of 10^{-5} , with both positive and negative errors evenly distributed across the surface. This pattern resembles random noise rather than systematic bias, which indicates that the model does not consistently over- or under-predict the surface location and is capable of reconstructing the geometry accurately. To provide a more meaningful assessment of the geometric accuracy, the surface displacement was also normalized by the wall thickness of each validation case. The maximum relative reconstruction error is observed for the smallest thickness ($thickness = 0.001$), where it reaches approximately 4%, while remaining below 1% for all other validation geometries. While the maximum surface displacement provides information on the worst-case reconstruction error, the full error distribution is further analyzed by considering the absolute value of the predicted SDF on the true surface. In Figure 3.5, box plots are reported for each training and validation geometry to quantify the magnitude of the surface reconstruction error independently of its sign. Note that outliers are not displayed for visualization purposes, while all samples are retained for the computation of the box plot statistics. Figure 3.5a shows that the median surface reconstruction error is below 1% of the wall thickness for all training configurations, except for the minimum-thickness case, where it is slightly higher. Similarly, Figure 3.5b shows that the normalized reconstruction error remains below 1% for all unseen validation configurations. The apparent reduction of the normalized error with increasing wall thickness is not related to a change in the reconstruction accuracy of the model, but rather to the normalization procedure. Since the absolute surface displacement remains approximately comparable across the different geometries, its relative value naturally decreases as the wall thickness increases. These results indicate that the reconstructed zero level set remains close to the reference surface, with deviations limited to a small fraction of the characteristic wall thickness.

3.4.2 Gaussian Surface Deformations

For this experiment, a gyroid geometry with a fixed wall thickness of $3 \cdot 10^{-4}$ m is considered. To generate geometric variations, 80 control points were randomly

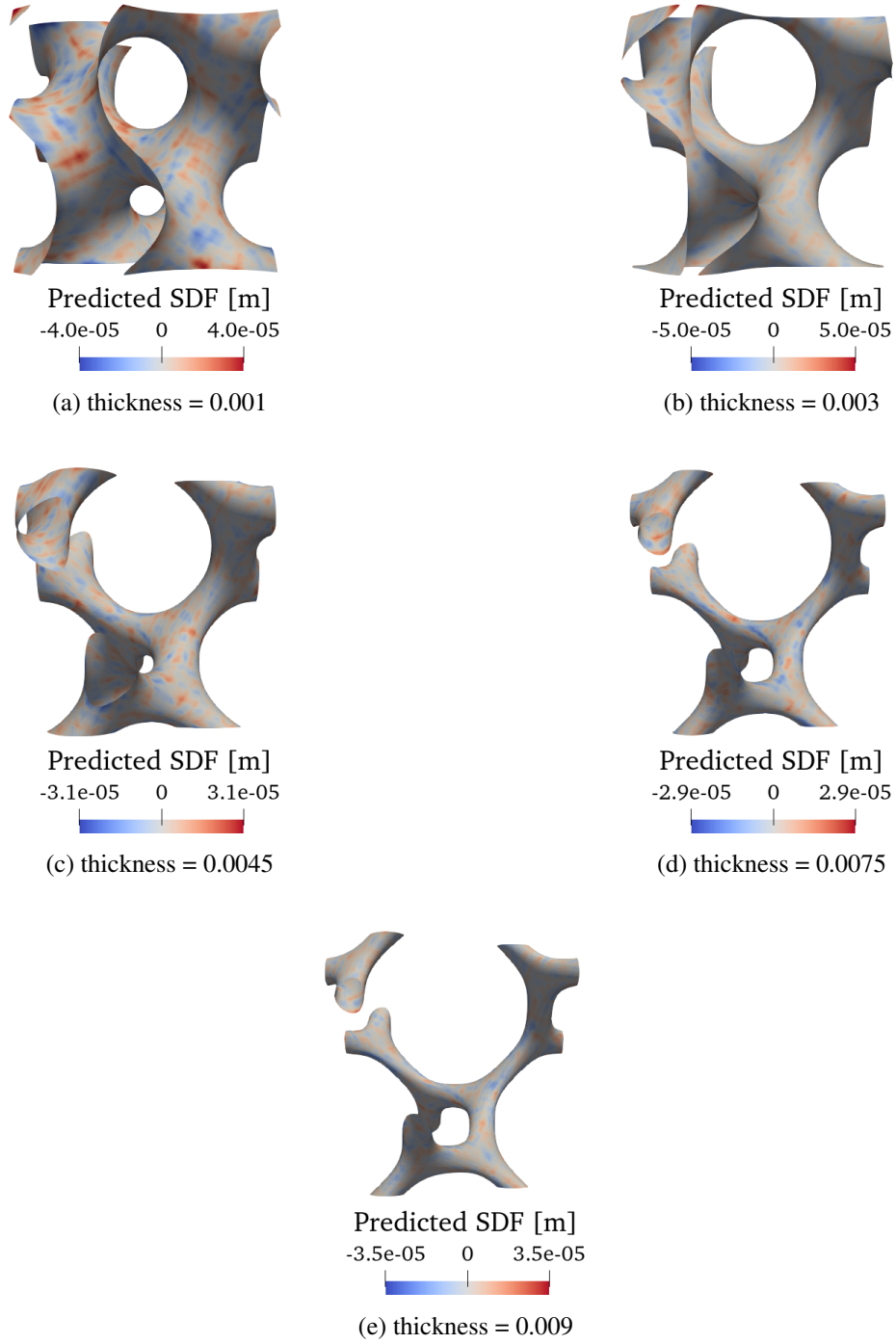
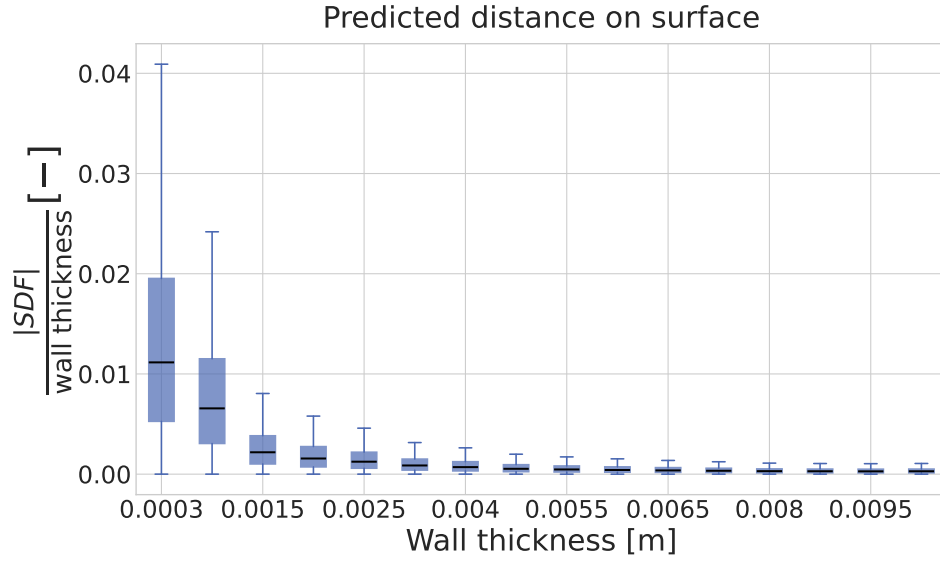
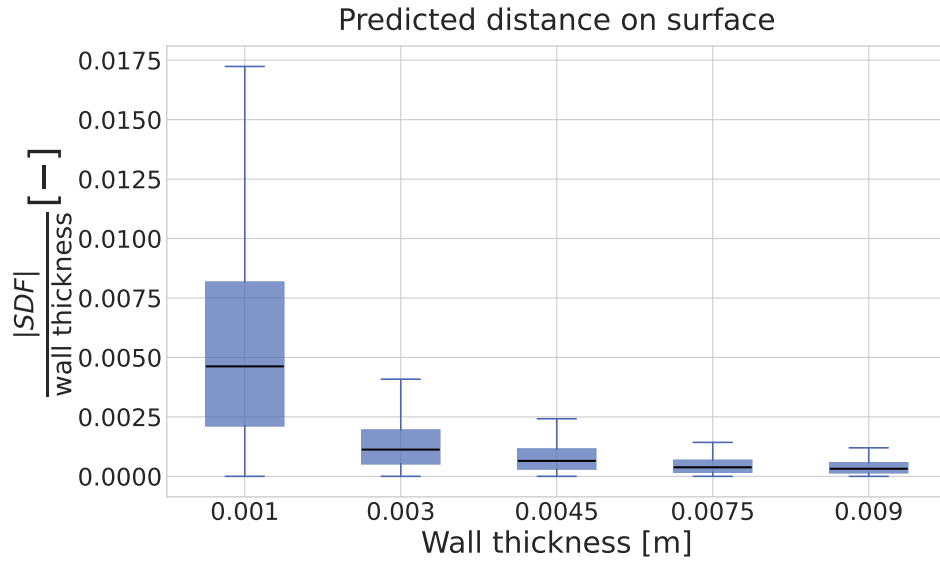


Fig. 3.4 Predicted value of the SDF on surface for the five validation geometries.



(a)



(b)

Fig. 3.5 Distribution of the predicted surface distance, expressed as the absolute SDF normalized by the corresponding wall thickness, for different wall thicknesses. Each box plot summarizes the distribution of the normalized $|SDF|$ values over the surface for (a) training and (b) validation geometries.

selected among the vertices of the surface mesh. Figure 3.6 shows the reference geometry together with the selected control points. For each deformed instance, a displacement amplitude was independently assigned to each control point by sampling values in the range $[0, 5 \cdot 10^{-4}]$ m. Since the Gaussian kernel is normalized, this value represents the maximum local contribution of the corresponding control point to the normal displacement field. The selected range was chosen to introduce geometrically relevant variations while preserving the overall topology of the gyroid structure.

The deformation field is constructed as a weighted sum of Gaussian radial basis functions centered at the control points. Specifically, the normal displacement at a surface vertex $\mathbf{x}$ is computed as

$$\delta(\mathbf{x}) = \sum_{j=1}^{N_c} w_j K(\mathbf{x}, \mathbf{c}_j), \quad (3.8)$$

where $N_c = 80$ is the number of control points, $\mathbf{c}_j$ denotes the j -th control point, w_j is the associated displacement amplitude, and K is a Gaussian kernel. The deformed surface is then obtained by displacing each vertex along its normal direction:

$$\mathbf{x}' = \mathbf{x} + \delta(\mathbf{x})\mathbf{n}(\mathbf{x}). \quad (3.9)$$

The displacement amplitudes were constrained to positive values, so that all local perturbations act consistently along the surface normal direction. This choice was adopted to preserve the topology of the generated geometries and avoid excessive local distortions. No self-intersections were observed among the generated samples.

This relatively limited dataset is intentionally adopted to emulate a data-scarce setting, which is often encountered in practical applications, where generating high-fidelity geometries or simulations can be computationally expensive. In particular, the number of samples is small compared to the dimensionality of the deformation space (defined by the 80 independent control points), making the learning task more challenging. Figure 3.8 shows some examples of deformed geometries.

A set of 30,000 points was sampled for each geometry as described in Section 3.2 and the SDF was evaluated on each point. Figure 3.7 shows an example of SDF computed on a set of points sampled around a deformed geometry. The 100 SDF samples, each corresponding to a point cloud associated with a deformed geometry,

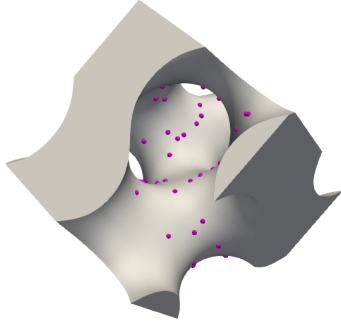


Fig. 3.6 Baseline gyroid geometry with control points for the Gaussian surface deformations highlighted in magenta.

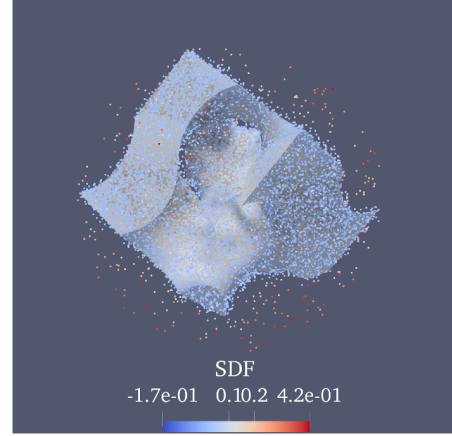


Fig. 3.7 SDF sampled around a deformed geometry.

were split into 80 training and 20 validation instances. The DeepSDF architecture and training setup are identical to those described in the previous experiment. The only difference is the number of training epochs, which was reduced to 2000. In this case, the loss evolution showed no significant improvement beyond this point, and additional training epochs were therefore not considered necessary. The loss value over the training epochs is displayed in Figure 3.9.

After training, the latent representations of the validation geometries are obtained by solving the optimization problem in Equation 3.5. Figure 3.10 shows the loss value through the optimization epochs for each validation instance. As in the previous experiment, the accuracy of the model is assessed by predicting the SDF on each training and validation point cloud. Figure 3.11 shows the relative SDF errors for the two datasets. Despite the limited size of the training set, the model is able to represent the training geometries in the latent space with a prediction error of approximately 4%. Moreover, the model exhibits an error on unseen geometries that is only 1 percentage point higher than on the training geometries, showing that the difference in reconstruction accuracy between training and unseen configurations remains limited. Also in this case, the SDF was predicted on all the vertices of the surface mesh of the geometry, to evaluate the displacement of the predicted surface with respect to the ground-truth. Figure 3.12 shows the surface SDF prediction for the instances that obtained the minimum, median and maximum prediction error across the validation dataset, respectively. Although the model exhibits a tendency to smooth peaks and valleys, the prediction error remains limited, as quantified by

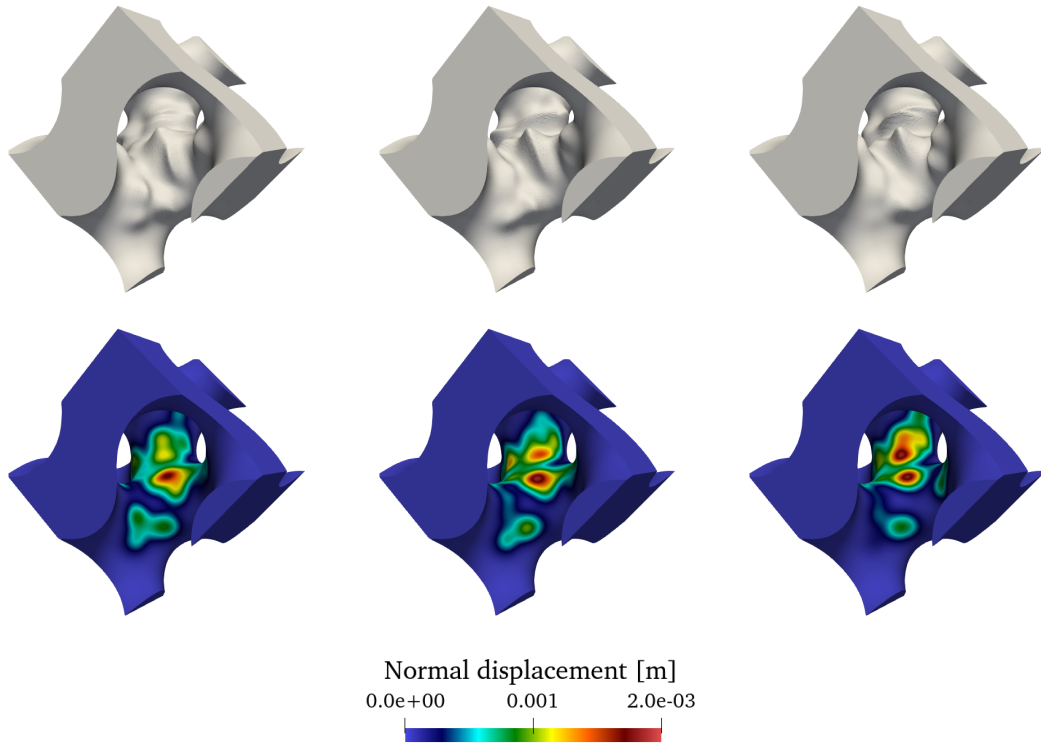


Fig. 3.8 Examples of geometries generated from the baseline gyroid by applying a Gaussian perturbation at randomly selected control points along the surface normal direction. The first row shows the resulting deformed geometries, while the second row reports the corresponding deformation fields used to generate them.

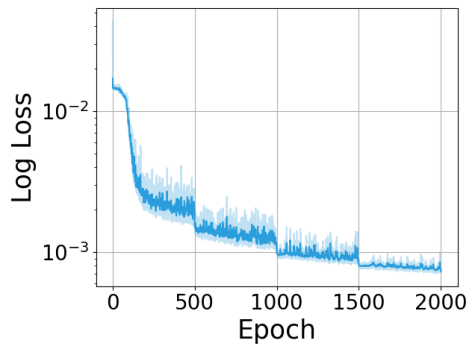


Fig. 3.9 Training loss for the encoding model trained on gyroids with Gaussian surface deformations

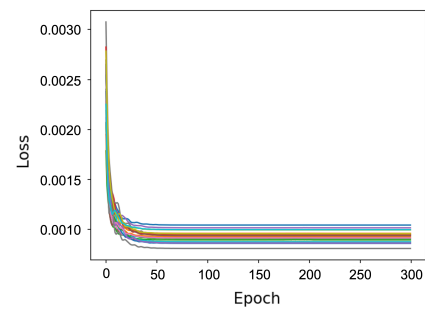


Fig. 3.10 Loss history for latent codes optimization.

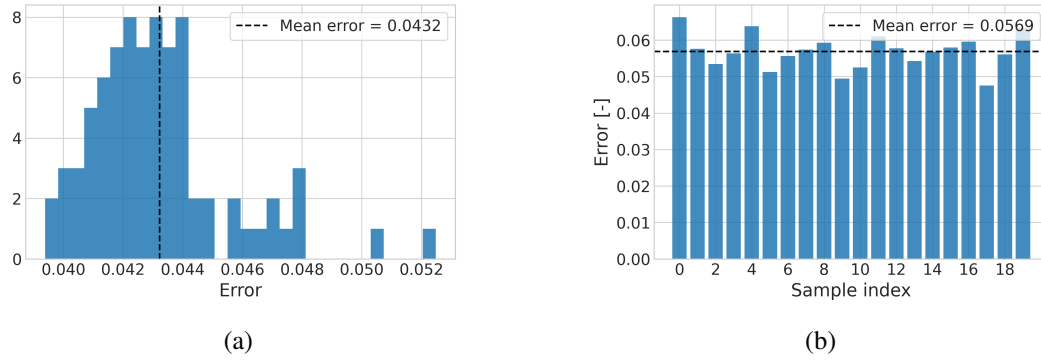


Fig. 3.11 Relative error on SDF predictions for (a) training and (b) validation dataset.

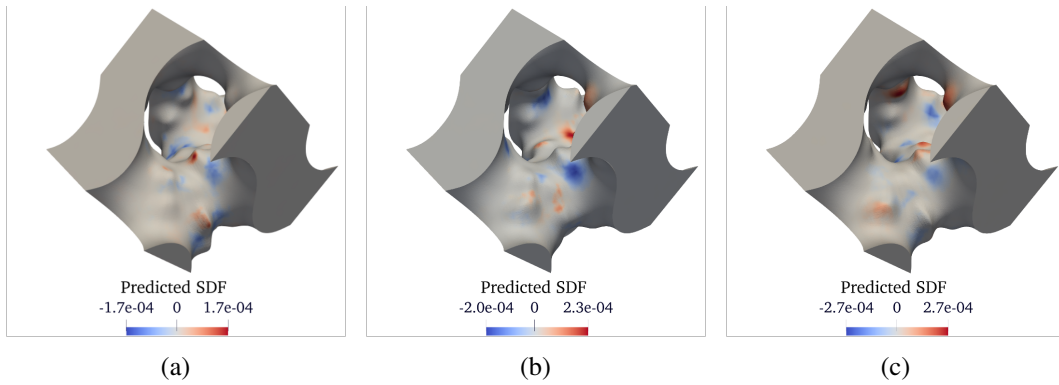


Fig. 3.12 Predicted SDF on surface for the geometries associated to the (a) best, (b) median and (c) worst prediction error.

the reconstruction metrics reported above, despite the relatively small size of the training dataset.

3.5 Final Considerations

The DeepSDF model provides a flexible framework for learning latent representations of geometries with varying shapes. By learning a latent encoding directly from SDF values sampled at arbitrary spatial locations, this approach mitigates two important limitations of more classical methods. First, it removes the need for an explicit, user-defined shape parametrization, enabling the integration of complex and heterogeneous datasets. Second, the implicit representation of geometries through their SDF, evaluated at arbitrary spatial points, eliminates the need for a common reference mesh. This is advantageous in practical scenarios, where different

geometries are typically associated with distinct computational meshes, which may differ in resolution, connectivity, and spatial discretization.

The experiments presented in this chapter show that this approach is able to encode complex geometries while maintaining low reconstruction errors, even in data-scarce scenarios. The resulting fixed-size latent representation can therefore be interpreted as an a posteriori parametrization and used as input for predictive models that map each configuration to the corresponding physical quantities. Although latent codes lack the direct interpretability of explicitly defined geometric parameters, this approach becomes essential when a suitable parametrization of the design space is unavailable or difficult to define.

In the next chapter, this model will be extended to encode spatially distributed physical quantities, and used as a building block to create a predictive framework.

Chapter 4

Learning Framework and Latent Space Regularization

The objective of this work is to design a computational workflow for predicting a physical quantity, such as a signal or a scalar field, arising from the solution of a partial differential equation (PDE) defined over a parametrized domain. In general, the physical field $u(\mathbf{x}; \mu)$ is assumed to satisfy a PDE of the form

$$\mathcal{L}_\mu(u(\mathbf{x}; \mu, \mathbf{p})) = f(\mathbf{x}; \mu), \quad \mathbf{x} \in \Omega(\mathbf{p}), \quad (4.1)$$

where $\mathcal{L}_\mu$ denotes a differential operator depending on a set of physical parameters μ , f is a source term and $\Omega(\mathbf{p})$ represents a domain described through a set of geometric parameters $\mathbf{p}$. As discussed in Chapter 1, the goal is to approximate the solution field without explicit access to the parametrization $\mathbf{p}$ and without assuming the availability of a common discretization of the domains. To address this challenge, a representation learning strategy is adopted. Specifically, compact latent representations are first learned for both the input geometries and the corresponding PDE solutions. Subsequently, a mapping is learned between the latent space associated with the geometries and that of the solutions.

In this chapter, the deep-learning-based workflow proposed by Catalani et al. [41] is presented, which serves as the baseline for the methodology developed in this work. Building upon this framework, a regularization strategy is introduced which acts on the encoding model enforcing a notion of distance in the latent space. This results in a smoother latent representation and improves the generalization capabilities of

the overall workflow. The methodology presented in this chapter is based on the framework previously introduced in Balasso et al. 2026 [52]. Compared with the published work, the material has been reorganized and expanded to provide a more comprehensive description of the proposed learning framework and the latent-space regularization strategies.

4.1 Baseline Workflow

The workflow proposed by Catalani et al. (2024) [41] consists of three main components: geometric encoding, output encoding and latent-space mapping. The geometric encoding block aims at learning a compact representation of the input shapes directly from the set of training geometries. In parallel, the field of interest is compressed into a fixed-dimensional latent vector through the output encoding block. The final component of the workflow is the mapping block, in which a fully connected feed-forward neural network is trained to approximate the relationship between the latent space of the geometries and the latent space of the corresponding PDE solutions. All machine learning models involved in the workflow are trained in a supervised manner on a dataset consisting of geometries with varying shapes, each paired with the corresponding solution of the target PDE.

4.1.1 Encoding Blocks

The geometric encoding block is implemented through the DeepSDF model, introduced by Park et al. [38] and based on implicit neural representations (INRs). As discussed in Chapter 3, this model learns a fixed-size latent representation of the training geometries based on their signed distance function (SDF), without requiring explicit knowledge of the underlying parametrization. The SDF is sampled on point clouds distributed around of the geometry, allowing the model to operate independently of a fixed discretization or resolution.

This framework can be naturally extended to handle generic spatially or temporally distributed quantities, including the solution of a target PDE. Indeed, the output encoding block is implemented as a modification of the DeepSDF model, where the SDF is replaced by the scalar field or signal of interest. In this setting, the independent variable $\mathbf{x}$ represents the spatial or temporal coordinate on which the target

quantity depends. As a result, the model learns a continuous latent representation of the output field, analogously to the geometric case.

In order to train the output encoding model for a set of N fields, M points are sampled in the respective domains, resulting in point-cloud representations of the form

$$X_i = \{\mathbf{x}_j, u_j\} : u_j = \mathbf{u}(\mathbf{x}_j), \quad j = 1, \dots, M, \quad i = 1, \dots, N, \quad (4.2)$$

where $\mathbf{u}$ denoted the field of interest. The spatial locations $\mathbf{x}_j$ can be sampled at arbitrary positions within the domain and the number of sampled points is not required to be the same across different training instances. This flexible representation allows the direct use of data from computational meshes, without the need for a common grid or a consistent discretization across the dataset.

Training the DeepSDF model using the samples defined in Equation 4.2 results in a set of fixed-size latent vectors, each associated with a training field. Once trained, the model can be used to infer latent representations for unseen fields, through the optimization defined in Equation 3.5. Moreover, the trained model can decode any latent code into a corresponding field representation. In particular, the decoder operates independently of the discretization used during training, allowing the reconstruction of fields at arbitrary spatial locations. This implies that both training latent codes and latent codes obtained during inference can be decoded at a resolution that is, in principle, unlimited. However, in practice, the output fields originate from numerical CFD solvers and their effective resolution is constrained by the spatial discretization of the computational mesh used in the simulations.

In the output encoding block, the model architecture, as well as the training and inference procedures, are identical to those adopted for the geometric encoder, described in detail in Chapter 3. Nevertheless, learning a physical field is generally more challenging than learning a signed distance function. Indeed, the SDF is a continuous function that is differentiable almost everywhere, with a unit gradient magnitude in all differentiable points, except for a set corresponding to the medial axis. These properties make it particularly well-suited for neural networks approximation, which can typically achieve a high level of accuracy. In contrast, the target physical fields may exhibit strong non-linearities, abrupt variations in their gradients, discontinuities or multiple frequency components. As a consequence, fully

connected neural networks may struggle to accurately approximate such functions due to their inherent spectral bias [53].

To overcome this limitation, the spatial coordinate $\mathbf{x}$ is first mapped into a higher-dimensional representation using a positional Fourier embedding characterized by the frequency spectrum

$$\gamma(\mathbf{x}) = [\dots, \cos(2^l \pi \mathbf{x}), \sin(2^l \pi \mathbf{x}), \dots], \quad l = 0, \dots, L, \quad (4.3)$$

where L is a hyper-parameter that requires appropriate tuning. With this modification, the input to the encoding network becomes the concatenation $[\mathbf{z}, \gamma(\mathbf{x})]$, where $\mathbf{z}$ denotes the latent code. As shown in [54], this type of projection onto a higher-dimensional Fourier feature space mitigates the spectral bias of neural networks and enables the representation of functions containing multiple frequency components.

4.1.2 Mapping Block

Once the encoding models have been trained, both the geometries and the corresponding output fields are represented by fixed-size latent vectors. Therefore, a fully connected neural network can be trained to learn the mapping between the latent space of the geometries and that of the associated outputs. A schematic overview of the complete training workflow is provided in Figure 4.1.

At inference time, the output field corresponding to a new geometry is predicted starting from its associated SDF sample. First, the geometric latent code is estimated by solving the optimization problem in Equation 3.5. Then, the resulting latent representation is passed to the mapping block, which predicts the latent code of the corresponding output field. Finally, the predicted latent vector is provided as input to the output encoder, together with the query spatial or temporal coordinates. This step reconstructs the latent representation and yields the final output prediction at the desired spatial locations. Figure 4.2 provides a diagram of the inference workflow.

4.1.3 Simplified Workflow

The workflow presented in the previous sections is based on learning latent representations of both the input geometry and the corresponding solution. This workflow,

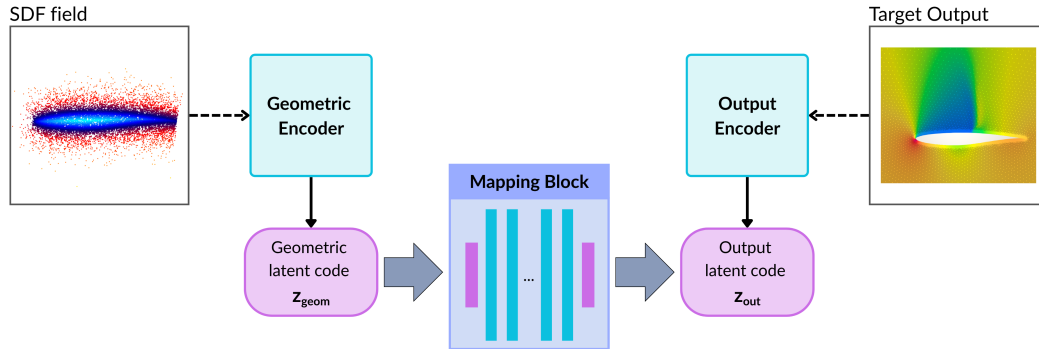


Fig. 4.1 Overview of the complete workflow training procedure. The geometric encoder and the output encoder are first trained separately to learn compact latent representations of the input geometries and target fields. Once the latent spaces have been established, the mapping block is trained to predict the output latent code from the geometric latent code.

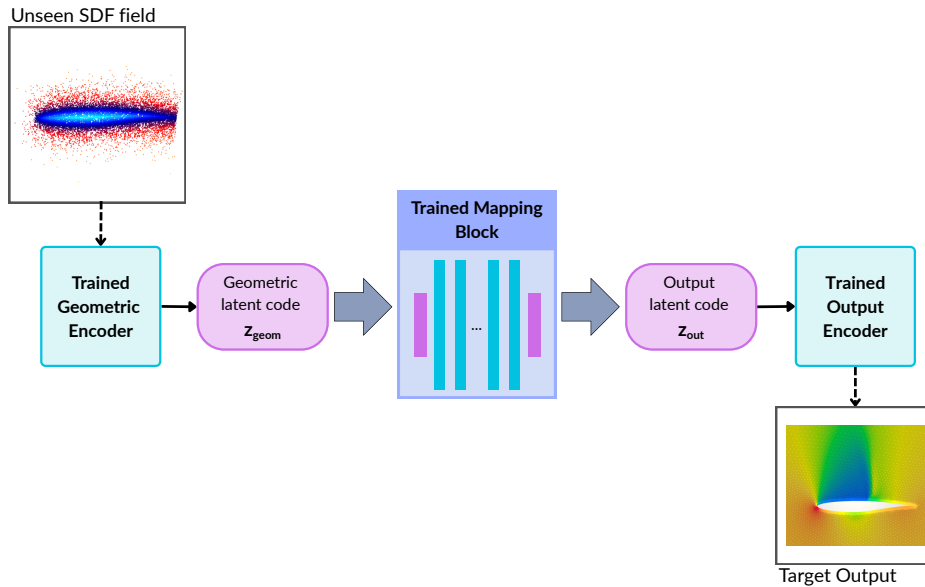


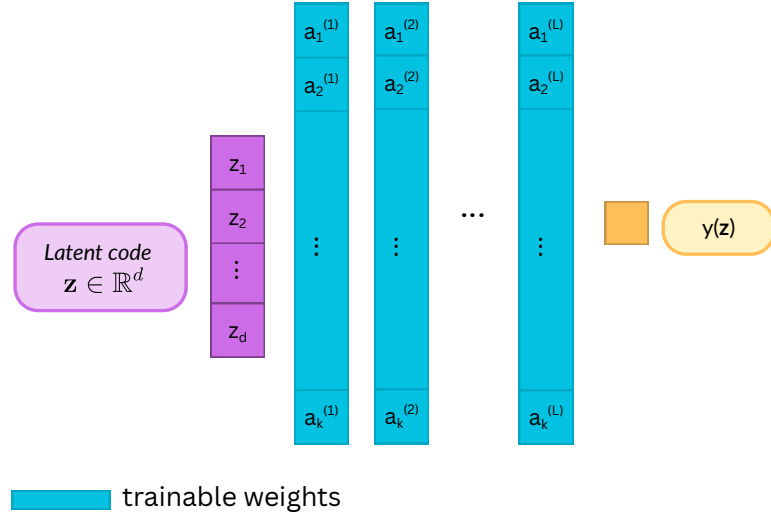
Fig. 4.2 Overview of the complete inference workflow. An unseen SDF field is provided to the trained geometric encoder, which estimates the corresponding latent code through the optimization procedure. The optimized latent code is then provided to the trained mapping block, which predicts the latent representation of the associated field. Finally, the predicted latent code is passed to the trained output encoder, which reconstructs the target field at the desired spatial or temporal locations.

that builds upon the work of Catalani et al. (2024) [41], will be referred to as the *complete workflow* (CWF).

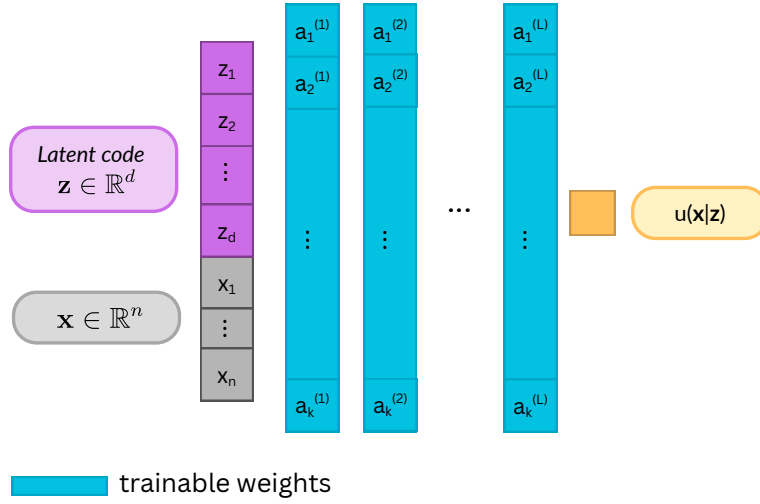
For specific classes of problems, this formulation can be simplified. In particular, the output encoding block can be omitted and the mapping block can be adapted to directly predict the target quantity. This simplification is naturally suited to the prediction of integral outputs, for which it is sufficient to train a network that maps directly the geometric encoding to the target scalar. However, this simplification can also be applied to spatially and temporally distributed quantities. Given a learned geometric representation, the simplified mapping block takes as input the geometric latent code z_{geom} and the spatial (or temporal) coordinate $\mathbf{x}$ and directly predicts the value of the field or signal associated with the shape represented by z_{geom} at that location $\mathbf{x}$.

The architecture of the SWF mapping block for scalar and distributed outputs is shown in Figure 4.3. For spatially or temporally distributed outputs (Figure 4.3b), the architecture is identical to the auto-decoder presented in Figure 3.1, with the only difference lying in the training procedure. Unlike the auto-decoder, the input latent code is not treated as an optimizable variable, but is provided by the pre-trained geometric encoder. Consequently, only the network parameters are updated during training. Moreover, as in the output encoder, a Fourier embedding is also applied to improve the field approximation. Figure 4.4 provides a schematic overview of simplified workflow in case of integral and distributed quantity of interest.

This approach provides a direct prediction framework, avoiding the explicit construction of a latent representation of the output field. The complete workflow, adopted from the literature, decomposes the geometry-to-field prediction task into separate encoding and mapping stages. While this strategy offers a highly expressive formulation, it also requires training multiple models, resulting in a significant computational cost. The simplified workflow was introduced to investigate whether the output encoding stage could be omitted without compromising the overall prediction accuracy. By directly learning the mapping from the geometric latent representation to the target output (and, for distributed fields, the corresponding spatial or temporal coordinates), the SWF provides a more compact formulation, reducing both model complexity and computational cost.

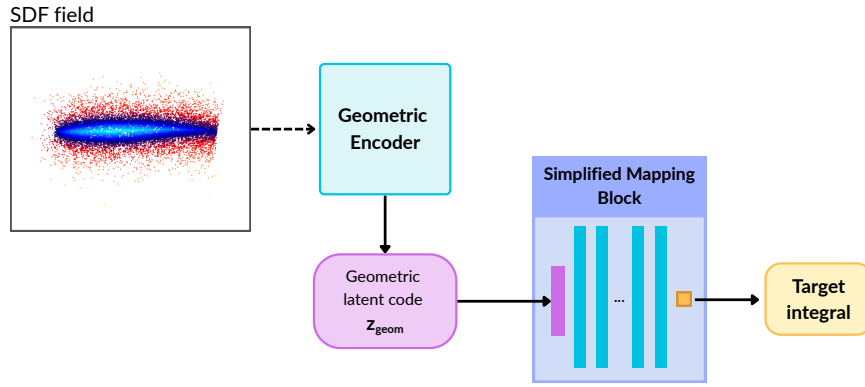


(a)

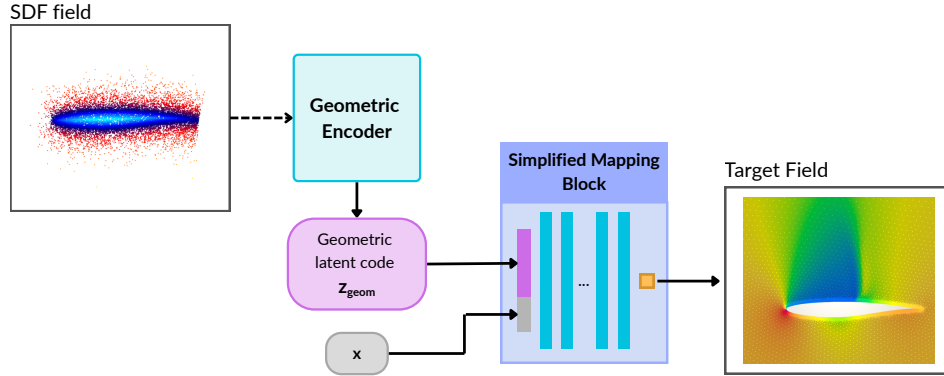


(b)

Fig. 4.3 Architecture of simplified mapping block for (a) integral and (b) spatially / temporally distributed output.



(a)



(b)

Fig. 4.4 Diagram of the simplified workflow (SWF) for (a) integral quantities of interest and (b) spatially or temporally distributed fields. Unlike the complete workflow, the mapping block directly predicts the target output rather than its latent representation. For distributed fields, the spatial or temporal coordinate is additionally provided as input, enabling point-wise prediction.

4.1.4 General Considerations

CWF and SWF enable the prediction of scalar outputs and spatially distributed fields associated with shape-varying geometries, without requiring an explicit parametrization of the underlying shapes. Moreover, since both workflows are trained on point-wise samples of the SDF and the target field, predictions can be evaluated at arbitrary query points, rather than being restricted to a predefined mesh or grid resolution. Another key advantage of the proposed framework lies in its modularity. Each component is trained independently and can be reused for different tasks without retraining from scratch. For instance, when a simplified workflow is trained to predict a scalar quantity from a given set of geometries, the resulting geometric encoder can be reused for other prediction tasks, such as surface or volumetric field reconstruction. In such cases, only the output-related components of the model and the intermediate mapping network need to be retrained, while the geometric encoder can be directly reused off-the-shelf.

In this chapter, we described an inference workflow for predicting the quantity of interest of a new geometry for which the SDF can be sampled in the vicinity of the surface. This is the inference setting adopted in the following chapters, as it enables direct comparison between the predicted quantities and the corresponding ground-truth values, allowing the accuracy of the proposed workflow to be assessed. However, the workflows presented in this chapter are not limited to known geometries. They can also be employed for design exploration and shape optimization by directly generating new configurations through exploration of the learned geometric latent space and evaluating or optimizing the corresponding quantity of interest.

If we want to reconstruct the surface of a newly generated geometry, the trained decoder is evaluated on a dense regular grid of query points covering the domain of interest. This produces a discrete sampling of the SDF, from which the zero-level isosurface can be extracted using standard algorithms such as Marching Cubes [55]. Although this reconstruction procedure relies on a regular grid, the grid is used solely as a numerical sampling structure during inference and does not constrain the underlying shape representation. Unlike mesh-based approaches, the model itself remains a continuous implicit function and is therefore independent of any reference mesh, predefined connectivity, or mesh topology. Furthermore, both the SDF evaluation and the Marching Cubes algorithm are highly parallelizable. The SDF value at each query point is computed independently, and each voxel cell can

be processed independently during isosurface extraction. Consequently, the entire reconstruction pipeline maps efficiently onto modern GPU architectures, allowing high-resolution meshes to be reconstructed within seconds on modern hardware.

4.2 Distance Preserving Regularization

In Section 4.1, a machine-learning-based strategy for the prediction of PDE solutions in problems involving geometrical variations has been presented. The approach relies on three independently trained components: two encoding blocks, which learn latent representations of the input geometries and the corresponding outputs, and a mapping block, which models the relationship between the two latent spaces.

In this section, the focus is placed on the encoding model and a regularization strategy is introduced to enforce a notion of distance in the learned latent space. The DeepSDF model used in the geometry encoding block does not explicitly enforce the preservation of geometric proximity, meaning that two similar shapes that are close in the geometric space may be mapped to distant points in the latent space. In practice, the model learns a latent representation that is optimal for the reconstruction task, which, however, does not necessarily reflect a meaningful notion of similarity between shapes. The resulting latent space can therefore exhibit a highly nonlinear and irregular structure. This behavior does not necessarily represent an issue for the encoding itself, whose objective is to minimize the reconstruction error. However, when the latent representations are arranged in a highly nonlinear and unstructured manner, the relationship between the geometric and output latent spaces may become extremely complex. While neural networks can, in principle, approximate such mappings, this lack of structure makes it difficult to learn a smooth relationship, ultimately leading to poor generalization on unseen geometries or the need of an extensive training dataset.

As an example of the lack of structure in the DeepSDF latent space, consider Figure 4.5, which refers to the vertical-axis wind turbine dataset described later in Section 5.1.1. For each shape in the training set, both the SDF-based distance and the corresponding distance in the latent space are computed with respect to all other shapes. Figure 4.5 shows the relationship between these two quantities. The Pearson correlation coefficient is 0.06, indicating an almost negligible linear correlation.

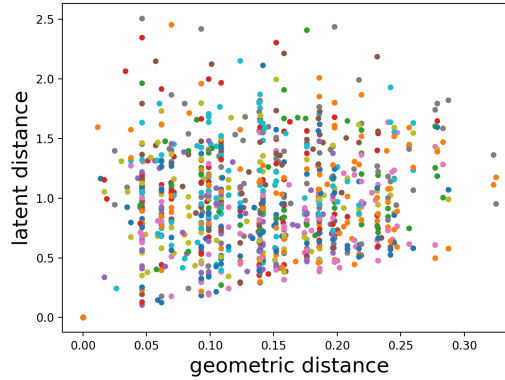


Fig. 4.5 Correlation plot of geometric distance and latent codes distance. Each color represents the distance of training samples from a fixed reference sample.

Therefore, geometrically similar shapes are not necessarily mapped to nearby latent representations.

To address this issue, a regularized version of the DeepSDF model is introduced, in which a penalization term is added to the loss function in order to promote the preservation of geometric distances in the latent space. A key aspect of this formulation is the definition of a suitable geometric distance, measuring the dissimilarity between two shapes. When geometric parameters are available, this distance can be naturally defined as the Euclidean distance in the parameter space. However, in general settings such information is not accessible. To maintain the parameter-agnostic nature of the proposed approach, the SDF representation of the shapes is instead exploited. Since the SDF is already used in the original training procedure, it provides a consistent way to define distances directly from the available data. In practice, evaluating distances between shapes represented through SDFs requires comparing the corresponding implicit functions at the same spatial locations. However, the SDF samples used by the geometric encoder are defined on shape-specific point clouds, generated independently for each geometry according to the sampling strategy described in Section 3.2. Since these point clouds are generally different for each shape, a direct point-wise comparison between SDF values is not possible.

To overcome this issue, an auxiliary shared point cloud is introduced solely for the offline computation of the pairwise regularization distances. For each pair of geometries, the corresponding SDF values are evaluated at these common query locations (through interpolation from the original shape-specific samples), and the resulting distance is computed before the training procedure starts. During training,

the regularization term combines precomputed geometric distances with pairwise distances between the latent codes, which are evaluated online. Since the geometric distances are computed only once, the auxiliary point cloud is no longer involved in the optimization loop, resulting in a negligible computational overhead. Moreover the auxiliary point cloud is not required to follow a specific discretization strategy as its role is only to provide a common reference for comparing implicit functions.

This distinction is important because the density of the auxiliary point cloud does not directly affect the computational cost of the neural network training. Unlike grid-based representations, where increasing the resolution increases the size of the input representation and the computational burden throughout the training and inference procedures, the shared point cloud is only used during an offline pre-processing step. Therefore, its density can be increased when needed to obtain a more accurate approximation of the SDF-based distance, for example to better capture small-scale geometric differences.

Moreover, the introduction of this auxiliary point set does not alter the geometric representation adopted by DeepSDF. During training, the reconstruction loss is still computed exclusively on the original shape-specific point clouds, which remain the primary source of supervision for learning the SDF associated with each geometry. The common query points only contribute to the additional regularization term. Consequently, the auxiliary point cloud does not define the latent representation nor constrain the sampling strategy of the DeepSDF model. The extraction of geometry-specific features remains driven by the SDF reconstruction loss, while the role of the regularization term is limited to organizing the latent space by encouraging geometrically meaningful relationships between the learned latent codes.

Once the SDFs are interpolated on the common point clouds, the geometric distance between training shapes can be computed in terms of Euclidean distance between SDF samples:

$$d_{SDF}^{(i,j)} = \frac{\|SDF_i - SDF_j\|_2}{\max_{i,j \in \{1, \dots, N_{train}\}} \|SDF_i - SDF_j\|_2}, \quad (4.4)$$

where $SDF_i \in \mathbb{R}^M$ is the SDF of the i -th training shape computed or interpolated on the common point cloud of size M . The distance defined in Equation 4.4 is normalized by the maximum value observed in the training dataset, ensuring that the resulting measure is independent of the geometric scale of the shapes.

It is worth emphasizing that preserving geometric organization in the latent space does not imply that geometrically similar configurations necessarily exhibit similar physical responses. The proposed regularization does not aim to enforce such a relationship, which depends on the underlying physical problem. Instead, it aims at avoiding arbitrary latent arrangements and providing a more structured representation of the geometric input, which can facilitate the learning of the geometry-to-output mapping.

4.2.1 Fixed Reference Distance Regularization

Given the distance between signed distance functions defined in Equation 4.4, a penalization term can be introduced in the loss function to encourage the latent space distances to match the corresponding normalized geometric distances. The proposed idea is to select a reference sample, acting as a centroid of the training dataset, and to fix its latent code at the origin of the latent space. During training, an additional regularization term enforces the latent codes of the remaining shapes to lie at a distance from the origin that is consistent with their normalized geometric distance from the reference shape. This strategy will hereafter be referred to as Fixed Reference Distance (FRD) regularization. The centroid is selected according to a min-max strategy, i.e., as the training sample that minimizes the maximum distance from all the other samples in the dataset:

$$i^* = \operatorname{argmin}_i (\max_j d_{\text{SDF}}^{(i,j)}). \quad (4.5)$$

In the original DeepSDF formulation, both the network weights and the latent codes are optimized jointly with respect to the same loss function $\mathcal{L}_{\text{DeepSDF}}$, defined in Equation 3.6, which includes a regularization term penalizing large latent vector norms. In the proposed regularized version, an additional penalty term is introduced to enforce consistency between latent space and geometric distances. In particular, the norm of each latent code, measured with respect to the centroid latent code (i.e., the origin), is encouraged to match the corresponding normalized SDF distance from the centroid in the geometric space.

The penalization term is defined as follows:

$$\begin{aligned}\mathcal{L}_{\text{FRD}} &= \frac{\sum_{i=1, i \neq i^*}^{N_{\text{train}}} |d_{\text{SDF}}^{(i, i^*)} - d_{\text{latent}}^{(i, i^*)}|}{N_{\text{train}} - 1} \\ &= \frac{\sum_{i=1, i \neq i^*}^{N_{\text{train}}} |d_{\text{SDF}}^{(i, i^*)} - \|\mathbf{z}_i\|_2|}{N_{\text{train}} - 1}.\end{aligned}\tag{4.6}$$

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

$$\mathcal{L} = \mathcal{L}_{\text{DeepSDF}} + \lambda \mathcal{L}_{\text{FRD}}\tag{4.7}$$

In order to implement the proposed regularization, the vector $\mathbf{d} = [d_{\text{SDF}}^{(i, i^*)}]_{i=1}^{N_{\text{train}}}$ containing the SDF distances from the reference sample is pre-computed. This allows to reduce the computational cost during training, since only the norm of the current latent codes needs to be evaluated at each iteration. The procedure followed at each training step, for a batch of size B and latent codes of dimension d , is summarized in Algorithm 1. Note that the latent code of the reference geometry is kept fixed during training. For this reason, the reference sample is excluded from the set of training samples.

Algorithm 1 Fixed Reference Distance Regularization

Input:

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

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

- 1: Initialize latent_dist $\in \mathbb{R}^B$
 - 2: Initialize sdf_dist $\in \mathbb{R}^B$
 - 3: **for** $i = 1 \rightarrow B$ **do**
 - 4: latent_dist[i] $\leftarrow \|\mathbf{Z}_{\text{batch}}[i]\|_2$
 - 5: sdf_dist[i] $\leftarrow \mathbf{d}[I_{\text{batch}}[i]]$
 - 6: **end for**
 - 7: $\mathcal{L}_{\text{FRD}} \leftarrow \text{mean}(|\text{latent_dist} - \text{sdf_dist}|)$
-

The regularization presented in this section provides a simple and effective way to enforce consistency between latent codes and geometric distances with respect to a reference shape. However, since the constraint is imposed only relatively to the centroid of the training dataset, the global coherence of the geometric latent space is not fully guaranteed. During training, latent codes are constrained to preserve a fixed distance from the reference sample. As a consequence, they are free to lie anywhere on the surface of a d -dimensional hypersphere, where d denotes the dimensionality of the latent space and the radius is determined by the corresponding normalized geometric distance from the reference shape. The most immediate consequence is that, while a strong correlation is expected between the distances $[d_{\text{SDF}}^{(i,t^*)}]_{i=1}^{N_{\text{train}}}$ and $[d_{\text{latent}}^{(i,t^*)}]_{i=1}^{N_{\text{train}}}$, this property does not necessarily hold for arbitrary pairs of training geometries.

4.2.2 Pair-Wise Distance Regularization

In order to overcome the limitations of the regularization described in Section 4.2.1, a natural extension consists in enforcing constraints on the distances between all pairs of latent codes $(\mathbf{z}_i, \mathbf{z}_j)$. In this way, the relative distances between all training samples are jointly regulated, encouraging the latent space to preserve the metric relationships of the input geometric space. Although the regularization does not explicitly constrain the latent coordinates, enforcing all pairwise distances limits arbitrary deformations of the latent space and preserves the relative proximity between geometrically similar samples. This strategy is referred to as Pair-Wise Distances (PWD) regularization.

The penalization term in Equation 4.6 is modified such that all training samples act as reference points simultaneously. The PWD penalty term is defined as:

$$\mathcal{L}_{\text{PWD}} = \frac{\sum_{i=1}^{N_{\text{train}}} \sum_{j=1, j \neq i}^{N_{\text{train}}} |d_{\text{SDF}}^{(i,j)} - d_{\text{latent}}^{(i,j)}|}{N_{\text{train}}(N_{\text{train}} - 1)}, \quad (4.8)$$

where the latent distance is the Euclidean norm between vectors

$$d_{\text{latent}}^{(i,j)} = \|\mathbf{z}_i - \mathbf{z}_j\|_2. \quad (4.9)$$

The Euclidean distance was adopted as the similarity metric since the latent space is learned as a continuous Euclidean embedding, making the L_2 norm the most natural choice for preserving relative geometric relationships. This modified penalty term can also be interpreted as the average of the term in Equation 4.6 over all possible choices of the reference geometry i^* .

The proposed approach offers a two-fold advantage with respect to the FRD regularization. First, it does not require removing any reference geometry from the training set. In general, the exclusion of a single sample has a negligible impact on model accuracy. However, when the available training samples provide a limited coverage of the data manifold, removing one element may leave portions of the manifold insufficiently represented. Consequently, the model may have to infer the geometric features of these regions from more distant samples, potentially affecting the accuracy of the learned representation. Second, the method removes the dependency on the specific choice of a reference sample, which is not trivial and may have a non-negligible impact on model performance. For instance, in the previous section a selection strategy based on a min-max criterion was proposed, but alternative choices could lead to significantly different behaviors of 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 in an offline stage. The scalability of this computation with respect to the dataset size is analyzed in Appendix A. The procedure used at each training iteration to compute the penalization term is described in Algorithm 2. At each step, latent space distances are computed between the latent codes in the current batch and all latent codes stored from the previous iteration, which act as reference points. A mask is applied to exclude self-comparisons, since otherwise each latent code would be biased towards its previous value, preventing meaningful updates during training.

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

$$\mathcal{L} = \mathcal{L}_{\text{DeepSDF}} + \lambda \mathcal{L}_{\text{PWD}} \quad (4.10)$$

The hyper-parameter λ controls the relative importance of the latent space regularization term with respect to the original SDF reconstruction loss. Increasing λ strengthens the constraint that geometrically similar configurations should remain

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 $I_{\text{batch}} \in \{1, \dots, N\}^B$,
- 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 $\mathcal{L}_{\text{PWD}}$

```

1: Initialize latent_dist  $\in \mathbb{R}^{B \times N}$ 
2: Initialize sdf_dist  $\in \mathbb{R}^{B \times N}$ 
3: for  $i = 1 \rightarrow B$  do
4:   for  $j = 1 \rightarrow N$  do
5:     latent_dist[ $i, j$ ]  $\leftarrow \|Z_{\text{batch}}[i] - Z_{\text{all}}[j]\|_2$ 
6:     sdf_dist[ $i, j$ ]  $\leftarrow D[I_{\text{batch}}[i], j]$ 
7:   end for
8: end for
9: Construct mask  $M \in \{0, 1\}^{B \times N}$  with  $M[i, j] = 0$  if  $I_{\text{batch}}[i] = j$ , else 1
10: latent_dist  $\leftarrow$  latent_dist  $\odot M$ 
11: sdf_dist  $\leftarrow$  sdf_dist  $\odot M$ 
12:  $\mathcal{L}_{\text{PWD}} \leftarrow \text{mean}(|\text{latent\_dist} - \text{sdf\_dist}|)$ 

```

close in the latent space. Therefore, a sufficiently large value is required for the regularization term to effectively modify the organization of the latent representation and reduce the effect of arbitrary latent arrangements learned by the unconstrained DeepSDF model. However, λ must be carefully tuned. If its value is too small, the regularization contribution becomes negligible compared with the reconstruction loss, and the expected benefits of a structured latent space may not translate into improved prediction accuracy. Conversely, an excessively large value may over-constrain the latent optimization, forcing the model to prioritize geometric proximity over the information required to accurately represent the individual shapes and their associated physical responses.

4.2.3 Output Distance Regularization

In the previous sections, a regularization strategy was introduced for the geometric encoding, in order to address lack of consistency between latent space and geometric distances. However, the same issue affects the output encoding, where proximity in the solution space is not necessarily preserved in the learned latent representation. In this section, the regularization approach is extended to the output encoding, with particular focus on PWD regularization.

To achieve this, it is necessary to determine a measure of distance between two solutions. Given a field of interest $\mathbf{u}$, the distance between two instances is defined as

$$d_{out}^{(i,j)} = \frac{\|\mathbf{u}_i - \mathbf{u}_j\|_2}{\max_{i,j \in \{1, \dots, N_{train}\}} \|\mathbf{u}_i - \mathbf{u}_j\|_2}, \quad (4.11)$$

where $\mathbf{u}_i$ is the field associated to the i -th geometry, computed on a set of points X_i . In order to compute the distance between all the pairs $(\mathbf{u}_i, \mathbf{u}_j)$ in the training dataset, the field must be evaluated at the same spatial locations. However, in presence of geometric variations, each solution is defined over a different physical domain, which prevents a direct point-wise comparison. For instance, in external aerodynamics problems, volumetric fields are defined only in the region outside the geometry, which varies from one sample to another. As a result, the domains of definition of the fields do not fully coincide.

To overcome this issue, an approximate distance can still be computed by introducing a common reference set of spatial locations. Specifically, each solution field $\mathbf{u}$

is interpolated onto a shared set of points sampled in a region where all training solutions are defined, enabling a consistent comparison across the dataset. Although this procedure neglects some geometry-specific regions, it introduces an approximation in the evaluation of the field distance. The obtained results, presented in the following sections, show that this approximation is sufficient for enforcing the desired regularization effect and improving the reconstruction accuracy. Using the distance between solutions defined in Equation 4.11, the corresponding regularization term can be formulated as follows:

$$\mathcal{L}_{\text{PWD}} = \frac{\sum_{i=1}^{N_{\text{train}}} \sum_{j=1, j \neq i}^{N_{\text{train}}} |d_{\text{out}}^{(i,j)} - d_{\text{latent}}^{(i,j)}|}{N_{\text{train}}(N_{\text{train}} - 1)}, \quad (4.12)$$

where the latent distance is the Euclidean norm in Equation 4.9. The global loss function for the output encoder with PWD regularization becomes:

$$\mathcal{L} = \mathcal{L}_{\text{DeepSDF}} + \lambda \mathcal{L}_{\text{PWD}}. \quad (4.13)$$

As discussed at the end of Section 4.2.2, the penalty term is applied only during training and is not used in the optimization of validation latent codes.

Overall, the extension of the regularization to the output encoder promotes a more coherent structure of the latent space representing the solutions, ensuring that similarity relationships in the solution space are better reflected in the learned representation. This contributes to a more stable and consistent mapping between geometry and output in the complete workflow.

Chapter 5

Test Cases

In this chapter, we present the experimental evaluation of learning framework proposed in Chapter 4. The goal is to assess the model’s ability to learn meaningful latent representations and accurately predict the target quantities on unseen configurations.

We consider three test cases to demonstrate the capabilities of the proposed framework. The first test case addresses the prediction of thrust and radial forces in a Vertical Axis Wind Turbine (VAWT). For each blade geometry, the objective is to predict the evolution of both force components over a full turbine revolution, as a function of the rotation angle. The second test case focuses on the prediction of the full pressure field around a transonic RAE2822 airfoil. Finally, the third experiment considers a three-dimensional external aerodynamics problem, namely the flow past a car, with the aim of predicting the pressure field on the car surface and the total drag coefficient.

The chapter is organized as follows. In Section 5.1, we describe the datasets used in the experiments, including their generation and pre-processing. In Section 5.2, we report the experimental setup and discuss the obtained results. The experimental validation presented in this chapter partially builds upon the results reported in Balasso et al. 2026 [52]. In addition to reproducing the published benchmark cases, the thesis includes further analyses and an additional three-dimensional application that extends the original study.

5.1 Datasets

In this section, the datasets employed in the numerical experiments are described. Each dataset corresponds to a specific application, characterized by different levels of geometric complexity and output dimensionality. The following subsections provide the relevant details for each test case.

5.1.1 Vertical Axis Wind Turbine

The first experiment was conducted on a three-bladed Vertical Axis Wind Turbine (VAWT) with geometric variations. VAWTs are characterized by blades rotating around a vertical axis, which leads to complex aerodynamic interactions between the blades and the surrounding flow.

The turbine dynamics were approximated through a two-dimensional CFD simulation on the domain illustrated in Figure 5.1.

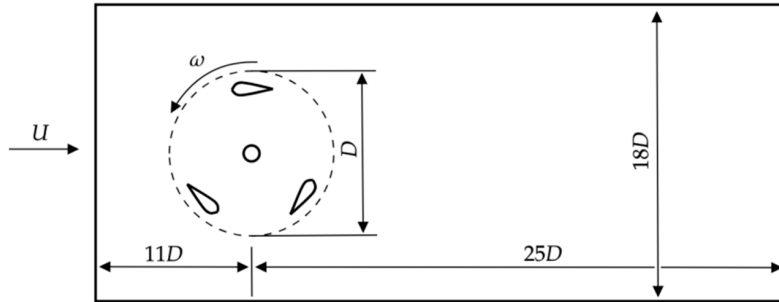


Fig. 5.1 Schematic depiction of the 2D computational domain used for the 3-bladed Darrieus VAWT.

The baseline blade geometry is a NACA0015 airfoil with a chord length of 420 mm, rotating at a constant angular velocity at a distance $R = 1.4$ m from the rotational axis. Further details on the geometry can be found in [56]. Since this work focuses on geometric variations, all operational parameters are kept constant, namely a wind speed of $U = 10$ m/s, a Reynolds number $Re = 290,000$ and a rotational speed $\omega = 14.29$ rad/s. The simulations are performed using Ansys Fluent [57] by solving the transient RANS equations coupled with the $k-\omega$ SST turbulence model. Blade rotation is represented through a cylindrical sliding mesh centered on the

rotational axis of the VAWT. The time-step used in the CFD simulations is defined based on the angular velocity in order to ensure a rotation of 1° per time-step.

In Section 5.2.2, the proposed approach is compared with a multi-fidelity POD method trained on both high-fidelity (411K cells) and low-fidelity (107K cells) simulations. The present workflow, instead, relies exclusively on high-fidelity data. Further details on the simulation setup can be found in [58].

In this test case, the objective is to predict the aerodynamic forces acting on the turbine blade for varying blade section geometries. 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, k_{thick} . Curvature modulation is introduced via a quadratic Bezier curve defined in the (x, y) plane by three control points: the leading edge, the trailing edge and an intermediate control point $(x_{maxThickness}, y^*)$, where $x_{maxThickness}$ denotes the chord-wise location of maximum thickness. The ordinate y^* of the intermediate control point is defined as:

$$y^* = c k_{curv}, \quad (5.1)$$

where c is the chord length and k_{curv} is a dimensionless coefficient controlling the intensity of the deformation. The resulting Bezier curve is given by:

$$B(x) = (1 - x)^2 P_0 + 2(1 - x)x P_1 + x^2 P_2, \quad (5.2)$$

where P_0 is the leading edge, P_1 is the intermediate control point and P_2 is the trailing edge. The deformed upper and lower surfaces are obtained by scaling the baseline profile as follows:

$$\begin{aligned} y_u(x) &= k_{thick} B(x) \bar{y}_u(x), \\ y_l(x) &= k_{thick} B(x) \bar{y}_l(x), \end{aligned} \quad (5.3)$$

where $\bar{y}_u(x)$ and $\bar{y}_l(x)$ represent the y -coordinate of the baseline NACA0015 airfoil.

The dataset consists of 16 geometrical variations of the baseline NACA0015 profile, obtained by sampling the parameter space (k_{thick}, k_{curv}) . In particular, a full factorial design with 4 levels per parameter is employed over the ranges $k_{thick} \in$

$[0.8, 1.4]$ and $k_{curv} \in [0.0, 0.05]$. The resulting 16 airfoil geometries are shown in Figure 5.2.

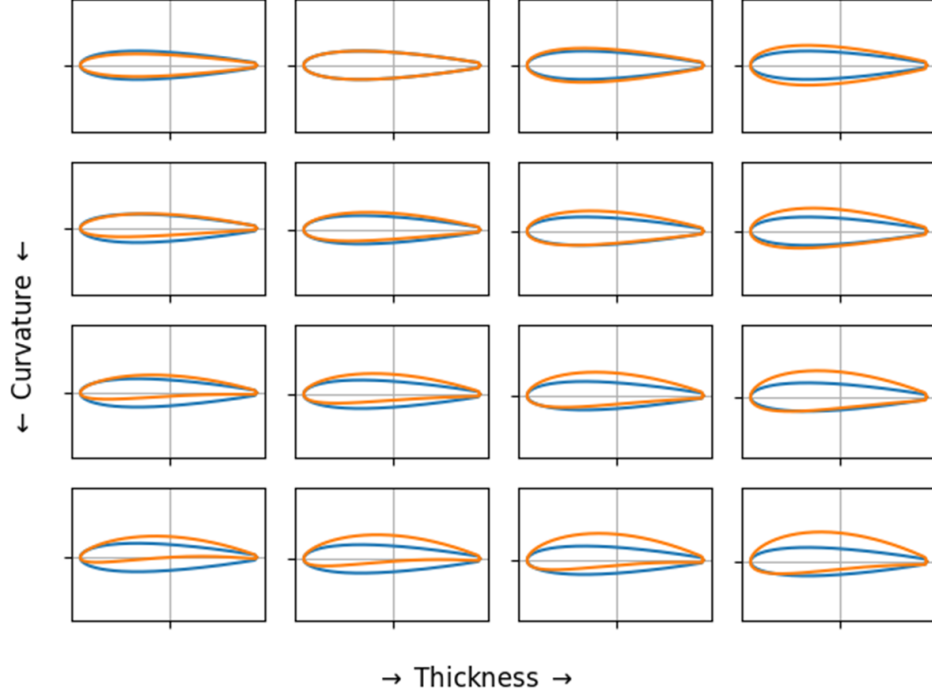


Fig. 5.2 Baseline NACA0015 (blue), and geometrical variations obtained by varying k_{thick} and k_{curv} (orange).

Aerodynamic fields, such as the velocity field shown in Figure 5.4, are obtained for each airfoil geometry through CFD simulations, following the setup described above. For this test case, the quantities of interest are the forces acting on a single blade over a full turbine revolution, namely the thrust force (F_T) and the radial force (F_R), evaluated as a function of the rotation angle $\phi \in [0, 360]$ (see Figure 5.3).

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

$$C_P = \frac{P_t}{P_w} = \frac{\omega R N_b \frac{1}{2\pi} \int_0^{2\pi} F_T(\phi) d\phi}{\frac{1}{2} \rho U^3 3RH}, \quad (5.4)$$

where N_b is the number of blades, ρ is the air density and H is the height of the wind turbine's swept area. In this experiment, we focus primarily on the thrust force

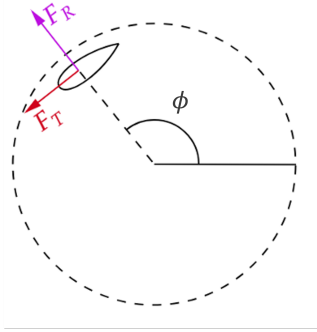


Fig. 5.3 Thrust force (F_T) and radial force (F_R) acting on a rotating turbine blade.

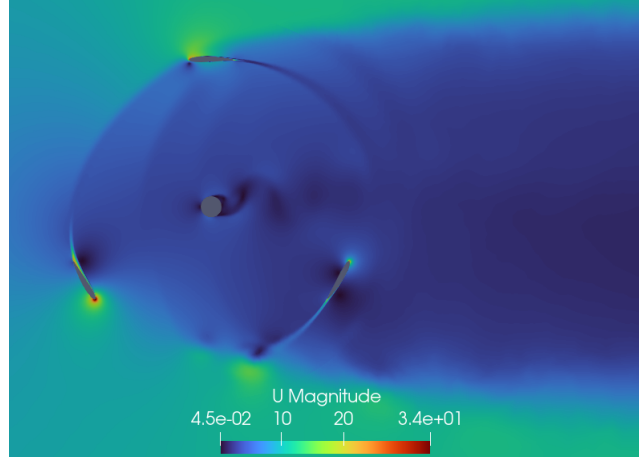


Fig. 5.4 Example of velocity field computed for a deformed airfoil geometry with thickness coefficient $k_{thick} = 0.8$ and curvature coefficient $k_{curv} = 0.016$. The field values are reported in their original physical scale, without normalization.

component, since F_T is the only contributor to the power coefficient, which is the quantity usually considered for optimization.

Figure 5.5a shows the power coefficient C_P computed from the CFD simulations across the sampled geometrical configurations. The results highlight a region, corresponding to low thickness and high curvature, where the C_P exhibits strong gradients. To improve the resolution of the dataset in this region, an adaptive refinement strategy is applied by introducing six additional designs, sampled using a space-filling approach within the identified high-gradient area. This results in a total of 22 geometrical configurations. In addition, five validation samples are selected at the centers of the cells defined by the original 4-level full-factorial grid, in order to assess the generalization capability of the proposed models. The complete dataset is shown in Figure 5.5b, where the original training data points are plotted as white square markers, the additional 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 dataset are shown in Figure 5.6 as a function of the rotation angle.

To train the geometric encoder, an additional set of airfoil geometries was generated, extending the range of configurations available in the simulation dataset. These additional samples belong to the same NACA0015-based family and were

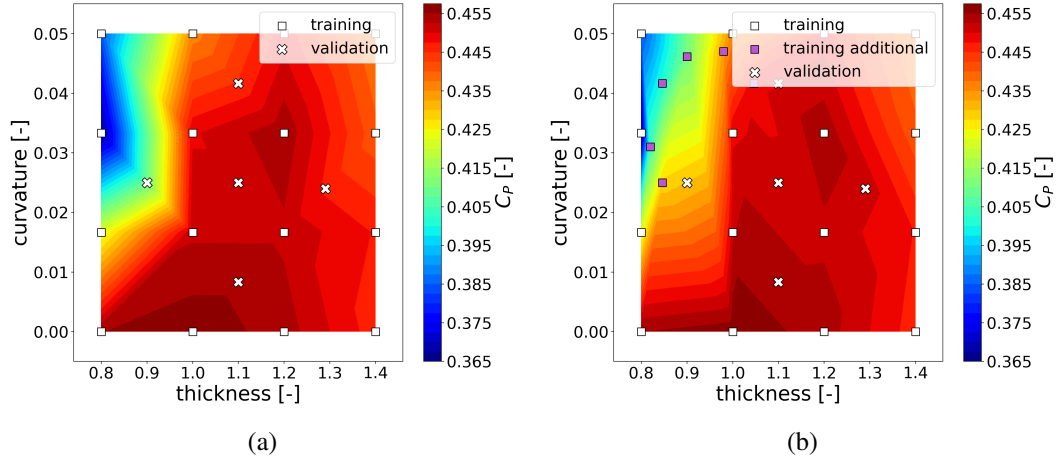


Fig. 5.5 Power coefficient as a function of the deformation parameters for the (a) initial and (b) complete dataset.

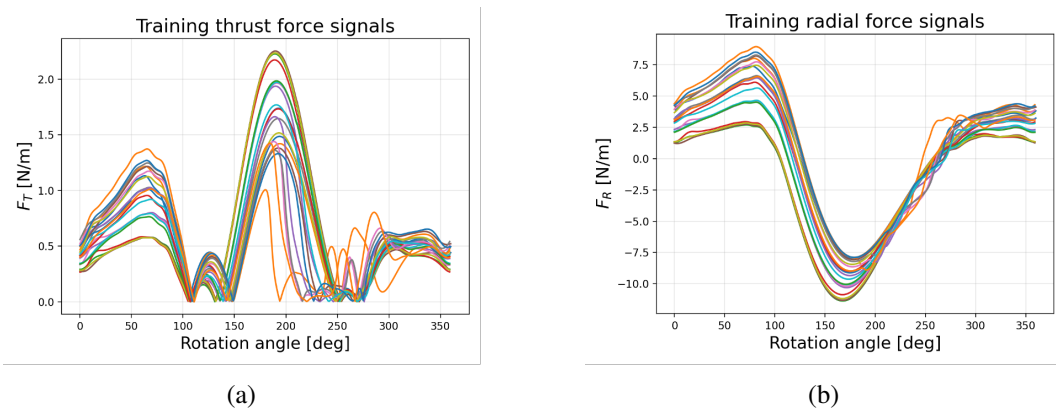


Fig. 5.6 (a) F_T and (b) F_R signals for each airfoil in the training dataset.

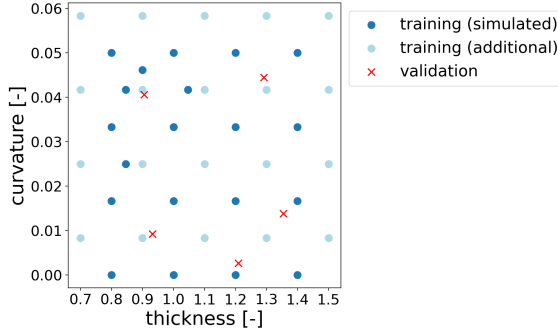


Fig. 5.7 Training dataset for the geometric encoder.

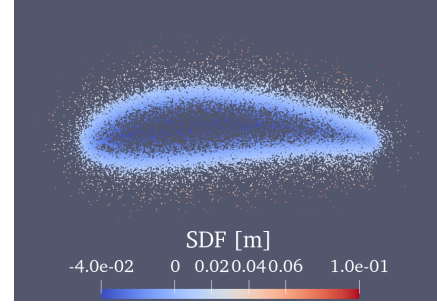


Fig. 5.8 Point cloud used for SDF computation of the airfoil with $k_{thick} = 1.4$ and $k_{curv} = 0.05$. The sampled points are colored according to their corresponding signed distance values, illustrating both the sampling distribution and the resulting SDF representation.

obtained through the same deformation procedure described at the beginning of this section. Since the geometric encoder only requires the shape information and not the associated physical fields, these geometries can be included without performing additional simulations. The training set for the geometric encoder is therefore expanded to 42 airfoils, including the 22 geometries described previously and 20 additional variations generated by sampling the parameter space on a second structured grid, offset with respect to the original full-factorial design. Furthermore, five additional airfoils are generated by random sampling within the same parameter space and are used as a validation set. The resulting augmented dataset used for training the geometric encoder is illustrated in Figure 5.7.

For each of the 42 training geometries, the signed distance function (SDF) is computed on a point cloud of 35,000 samples. The points are sampled around the airfoils' surface following the strategy described in Section 3.2. An example of a resulting point cloud for a training geometry is shown in Figure 5.8.

5.1.2 Transonic RAE2822

In the second experiment, a transonic RAE2822 airfoil is considered. Compared to the previous case, the geometric variations are more nuanced, while the quantity of interest is the two-dimensional pressure coefficient field around the airfoil. The baseline configuration is given by the RAE2822 airfoil, shown in Figure 5.10. The

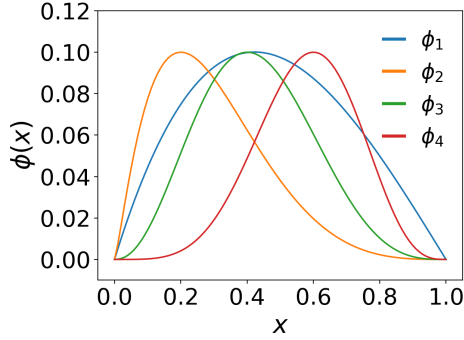


Fig. 5.9 RAE2822 deformation modes used to generate geometric variations of the base airfoil.

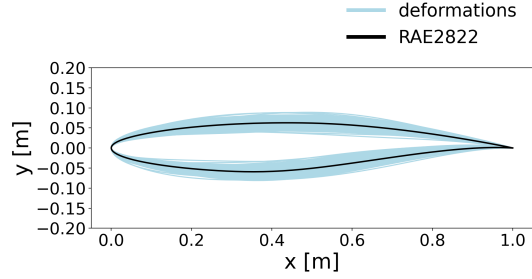


Fig. 5.10 Baseline RAE2822 and deformed airfoils in the training dataset.

reference geometry has a normalized chord length and a maximum thickness of 0.12 m.

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

$$\begin{aligned} y_l(x) &= k \left(\bar{y}_l(x) + \sum_{i=1}^4 c_i^l \phi_i(x) \right), \\ y_u(x) &= k \left(\bar{y}_u(x) + \sum_{i=1}^4 c_i^u \phi_i(x) \right), \end{aligned} \quad (5.5)$$

with $x \in [0, 1]$.

Here, $\bar{y}_l(x)$ and $\bar{y}_u(x)$ denote the lower and upper surfaces of the baseline RAE2822 airfoil, respectively. The scaling coefficient k is applied as a post-processing step to each geometry independently to preserve the original maximum thickness of the airfoil after deformation. The deformation modes $\phi_i(x)$ are predefined functions designed to introduce both local and global shape variations, allowing for a richer and more diverse set of geometric configurations (see Figure 5.9).

The coefficients $c_1^l, \dots, c_4^l, c_1^u, \dots, c_4^u$ control the deformation modes and can be interpreted as the geometric parameters of the problem. To construct the training dataset, 1,000 samples are generated by sampling the deformation coefficients using a Halton sequence [59] in the 8-dimensional parameter space. The Halton sequence is a quasi-random sequence with a nested structure, meaning that any subset formed by the first N_{sub} samples (with $N_{sub} \ll N_{train}$) is itself a Halton sequence. This

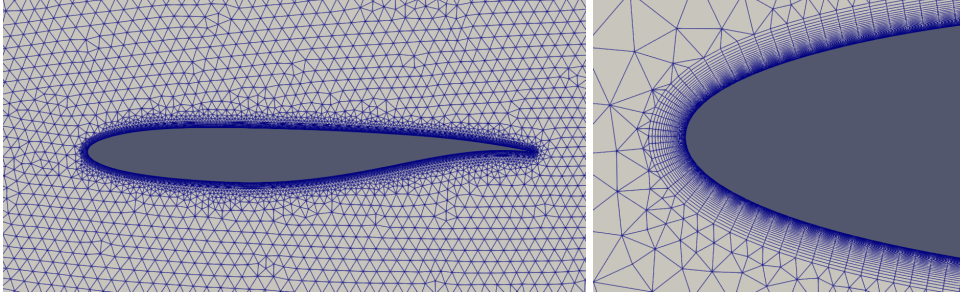


Fig. 5.11 Snapshots of the computational mesh for a geometric variation of the base RAE2922 in the training dataset.

property enables incremental training of the models with increasing dataset sizes, while preserving a uniform coverage of the parameter space at each level. As a result, error analysis can be performed consistently across different dataset sizes using nested subsets. The following ranges for the deformation coefficients are considered:

$$\begin{aligned}
 c_1^l, c_1^u &\in [-2, 2], \\
 c_2^l, c_2^u &\in [-0.5, 0.5], \\
 c_3^l, c_3^u &\in [-1, 1], \\
 c_4^l, c_4^u &\in [-1, 1].
 \end{aligned} \tag{5.6}$$

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 5.10 illustrates the 1,000 training deformations applied to the baseline airfoil compared with the base RAE2822. In order to assess the model accuracy on unseen configurations, 300 additional airfoil variations are sampled 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 strategy described in Section 3.2.

For each training and validation airfoil, the flow field is computed using the open-source CFD solver SU2 [60]. The simulations are performed at a fixed angle of attack $\alpha = 3.5^\circ$, Reynolds number $Re = 6.5 \cdot 10^6$, free-stream Mach number $Ma = 0.734$, and temperature $T = 216.65$ K. The computational domain is defined

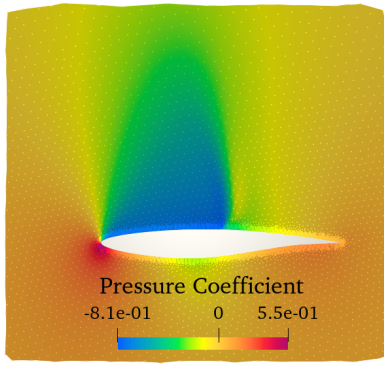


Fig. 5.12 Example of the pressure coefficient computed around a training airfoil and restricted to the inference domain.

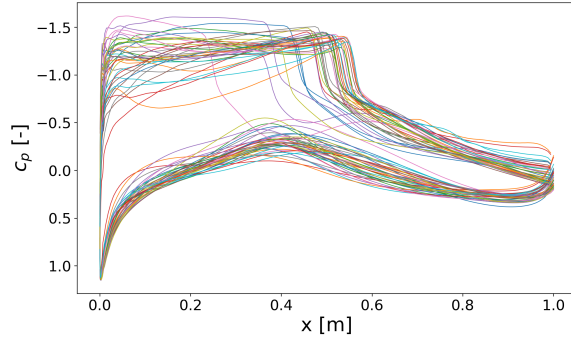


Fig. 5.13 Surface pressure coefficient for 40 geometrical variations of the base RAE2822 in the training dataset.

as a sphere centered at the airfoil leading edge, with a radius equal to 40 times the aerodynamic chord. An example of the mesh around a training geometry is shown in Figure 5.11. For training purposes, the full CFD solution is restricted to a smaller rectangular region centered around the airfoil, which defines the domain considered in the learning tasks. In this experiment, we focus on the pressure coefficient field C_p around the airfoil, which is the target quantity to be predicted. Figure 5.12 shows an example of C_p field for one of the training geometries on the restricted domain. The surface pressure coefficient distribution is reported in Figure 5.13 for 40 different variations of the baseline RAE2822 airfoil to give an idea of the dataset variability. The effect of geometric variation is clearly visible, with significant changes in both the position and strength of the shock on the suction side of the airfoil.

5.1.3 DrivAerML

The dataset employed in the third experiment is based on the DrivAerML dataset [1], which provides high-fidelity CFD simulations for automotive aerodynamics. The original publication provides a comprehensive description of the dataset, including the geometry generation procedure, simulation setup, and data processing pipeline. In this section, only the main characteristics relevant to the present study are summarized. The dataset is built upon the DrivAer notchback baseline geometry and consists of 500 parametrically deformed variants designed to span a wide range of realistic vehicle shapes. The geometries in the dataset are generated through a

parametric morphing approach. Specifically, a set of morphing boxes is constructed around the baseline DrivAer geometry, allowing controlled deformations of the car shape (see Figure 5.14a). A total of 16 design parameters are defined (represented in Figure 5.14b–d), each associated with physically meaningful geometric modifications (e.g., roof shape, rear end, or underbody features) and constrained within ranges chosen to avoid unrealistic configurations. This approach ensures that all generated geometries remain representative of real vehicle designs while still introducing significant variability in aerodynamic behavior. The deformation parameters are sampled using a Modified Extensible Lattice Sequence algorithm, which provides a good space-filling coverage of the parameter space. Figure 5.15 provides some examples of the deformed geometries obtained by varying the 16 deformation parameters. The front and side views highlight the different types of shape variations introduced by the deformation strategy.

The CFD simulations solve the incompressible Navier–Stokes equations using OpenFOAM with a hybrid RANS–LES turbulence model, specifically a variant of Delayed Detached-Eddy Simulation (DDES) and are performed on meshes of approximately 160 million cells. From volumetric flow fields, drag coefficients and pressure distribution on the car surface are computed by numerical integration and post-processing. In this work, the focus is on the mean kinematic pressure field p^* , corresponding to the statistically converged time-averaged pressure quantity provided in the DrivAerML dataset. In particular, the surface distribution of the mean kinematic pressure is considered as the prediction target. Figure 5.16 shows examples of the surface pressure distribution for selected geometry configurations from the training dataset.

In terms of dataset size, the DrivAerML dataset consists of 500 distinct geometry variants, each associated with a full high-fidelity CFD simulation. While this may appear limited compared to the dimensionality of the 16-parameter design space, it is in fact representative of realistic industrial scenarios, where the generation of high-quality CFD data is computationally expensive. This makes the problem particularly relevant and non-trivial from a machine learning perspective, as models must extract meaningful patterns and generalize effectively despite the limited sampling density. More details about the dataset generation and the simulation setup can be found in [1]. In order to train the encoding model, the SDF of each geometry was computed on a dedicated point cloud of size 35,000, generated with the strategy described in Section 3.2. The first target quantity is the drag coefficient (C_D), treated as a scalar

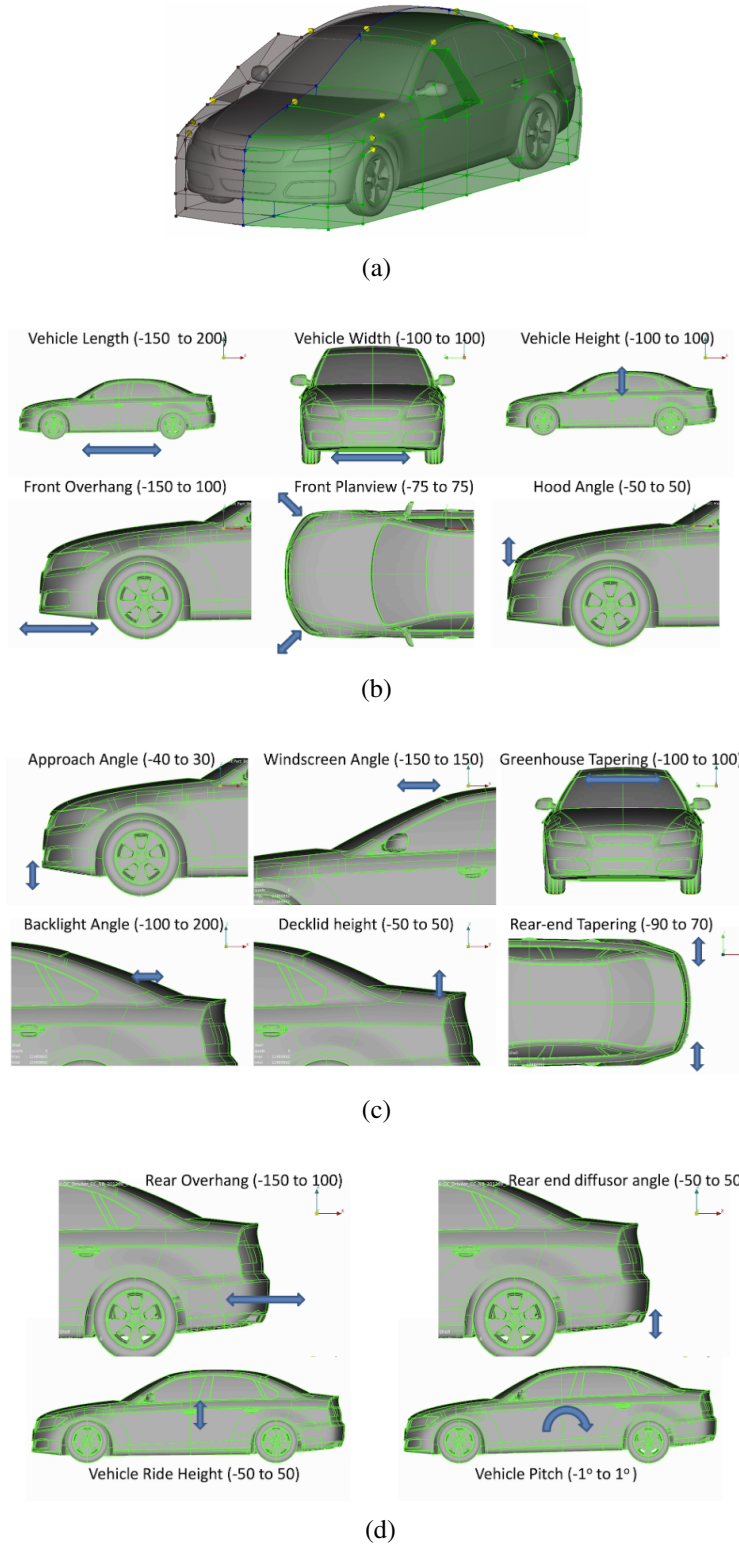


Fig. 5.14 Parametric generation of geometry variants starting from the baseline DrivAer model. (a) Definition of morphing boxes and (b–d) representation of the 16 geometric design parameters. Reproduced from Ashton et al. [1] with author's permission.

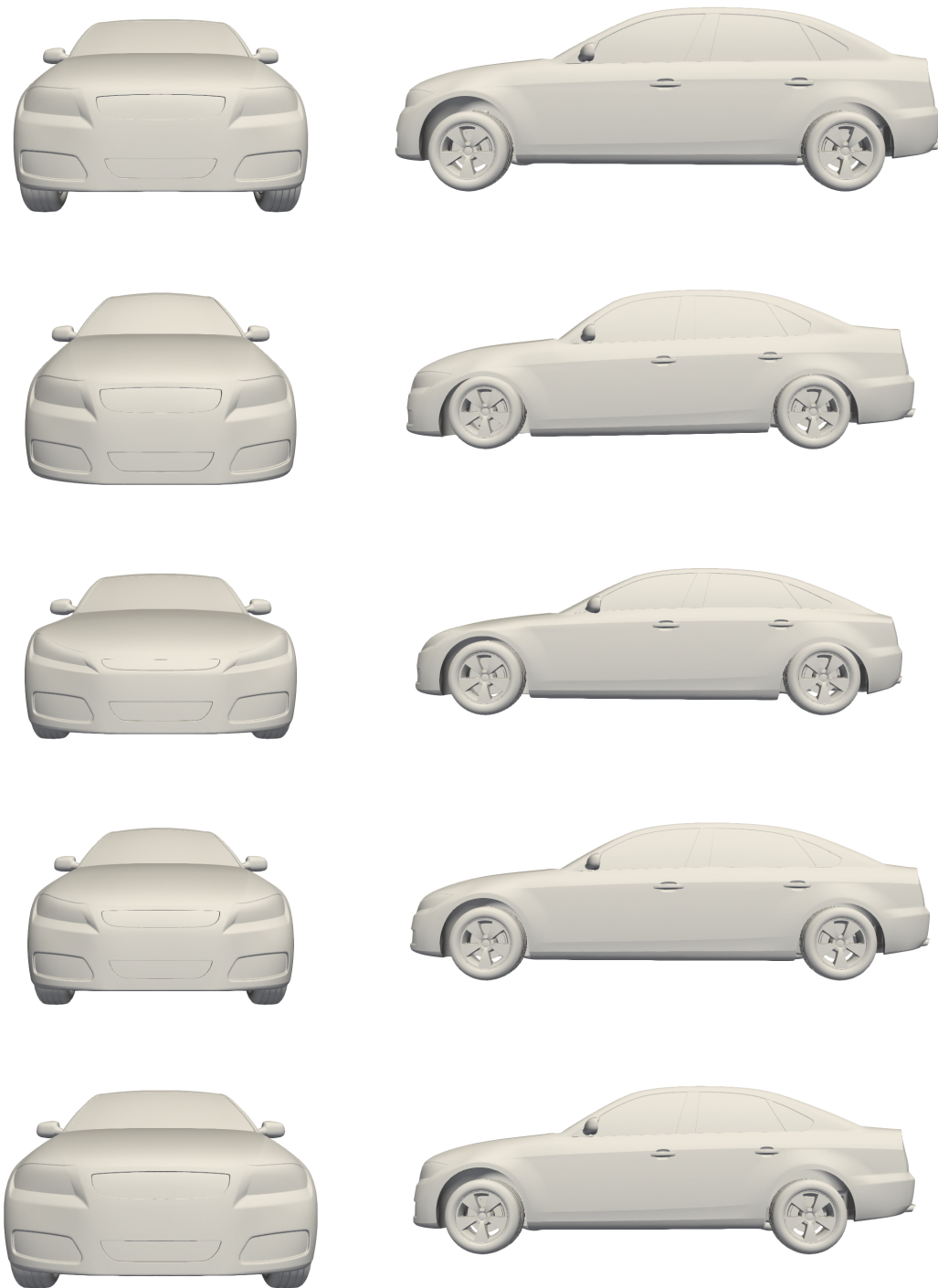


Fig. 5.15 Examples of deformed geometries generated by applying the full set of 16 deformation parameters. The corresponding front views are shown in the first column, while the side views are reported in the second column.

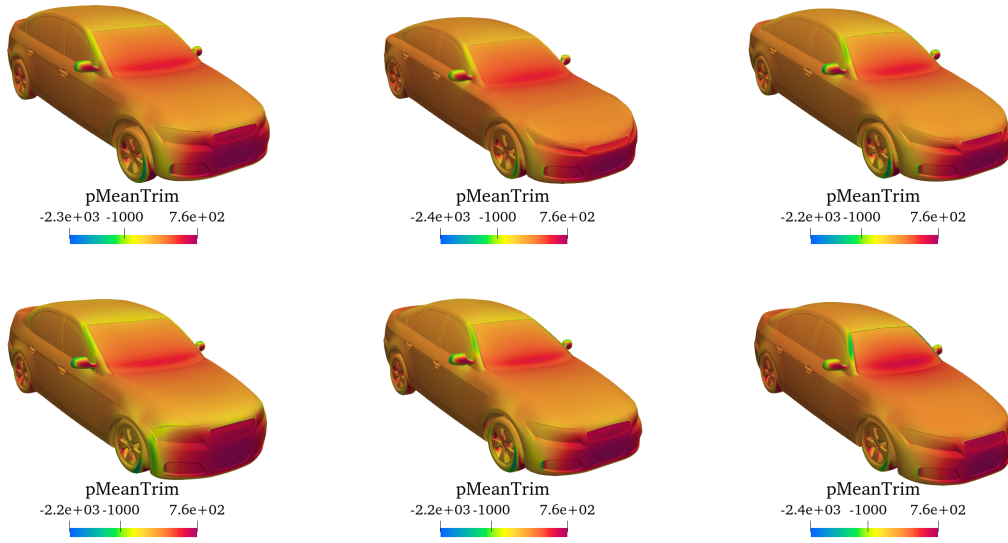


Fig. 5.16 Surface pressure distributions for six geometry configurations. Each configuration corresponds to a different combination of the 16 deformation parameters applied to the baseline vehicle geometry.

output. The second target is the pressure field on the vehicle surface. To reduce the computational cost associated with training, only a randomly sampled subset of 50,000 cell centers from the surface mesh is retained for each geometry. This partial sampling also provides a more challenging learning scenario, as the model is trained using only a subset of the available field values. During inference, however, the pressure field is reconstructed over the entire computational surface mesh, allowing the accuracy of the predicted complete field to be assessed. Figure 5.17 shows the SDF sample and the pressure sample associated to a training geometry.

5.2 Results

In this section, we present the results of the experimental evaluation of the workflow introduced in Section 4.1. As a first experiment, we consider a simplified version of the workflow (SWF), trained to predict the forces acting on a blade of a vertical axis wind turbine over a full turbine revolution. In this setting, geometry variations are parameterized by two design variables and the task consists of predicting one-dimensional force signals corresponding to unseen blade geometries. The second test case focuses on the prediction of the full two-dimensional pressure coefficient

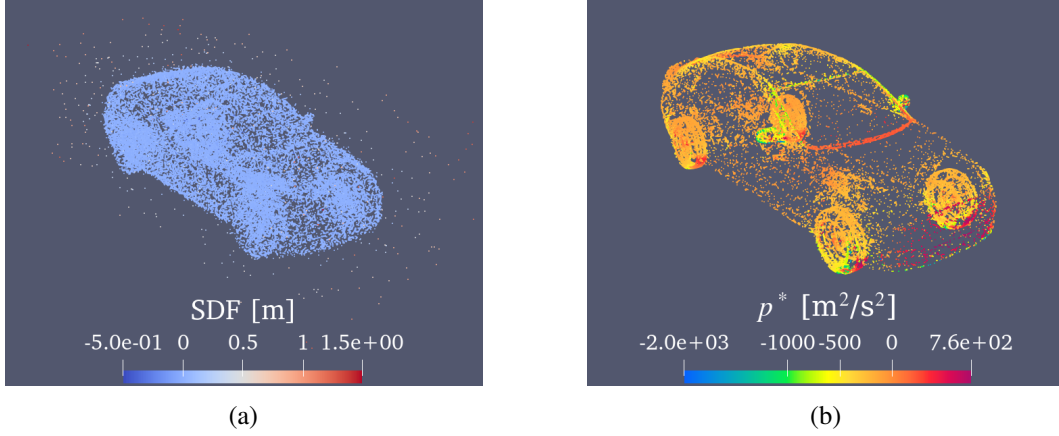


Fig. 5.17 Example of point-cloud samples for a training geometry. (a) SDF sample and (b) target pressure sample, obtained as random subsample of surface mesh cell centers.

field over geometric variations of a RAE2822 airfoil, parameterized by eight deformation variables. In this case, we employ the complete workflow (CWF), which includes the output encoding block. The third experiment focuses on a car geometry parameterized by 16 deformation variables, with the objective of predicting both the total drag coefficient and the three-dimensional surface pressure field.

For each test case, we evaluate the prediction workflow and assess the impact of the regularization strategies introduced in Section 4.2 on the prediction accuracy. All experiments were implemented in PyTorch [61] (v2.1) and trained on a single NVIDIA A10 GPU.

5.2.1 Models Validation and Error Metrics

In all experiments, each block in the workflow is trained independently and from scratch. At inference time, we first evaluate the encoding blocks separately, in order to assess the representation accuracy of both the geometries and (when applicable) the target fields. At this stage, no improvement due to the regularization strategies is expected, since they are designed to influence the mapping between the latent spaces rather than the encoding process itself.

The accuracy of the encoding blocks is assessed in terms of SDF (or output) reconstruction, by comparing the predicted values with the corresponding ground truth. For training samples, the SDF is reconstructed directly from the latent codes learned during training. For validation geometries, instead, the latent code is first

obtained by solving Equation 3.5, with the encoder weights kept fixed. The SDF is then reconstructed from the inferred latent code and compared with the reference solution.

The reconstruction error is evaluated in relative terms, that is:

$$error_i = \frac{\|\mathbf{y}_i - \hat{\mathbf{y}}_i\|_2}{\|\mathbf{y}_i\|_2}, \quad (5.7)$$

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.

The accuracy of the mapping block, which reflects the overall performance of the workflow, is evaluated by computing the prediction error directly in terms of the output of interest. Specifically, the geometric latent code is first passed through the mapping block to predict the corresponding latent representation of the target output. The predicted latent code is then decoded using the trained output encoder to reconstruct the output field, which is finally compared with the ground-truth solution. The end-to-end prediction error is computed using Equation 5.7. If the target quantity is an integral value, the relative error is computed as

$$error_i = \frac{|y_i - \hat{y}_i|}{|y_i|}, \quad (5.8)$$

where y_i and $\hat{y}_i$ are the true and the predicted value.

Lastly, given the error distribution on the training and validation datasets, outliers are defined as errors lying more than five standard deviations away from the mean. They are only omitted in the visualization of error histograms for clarity, but their count and maximum value are reported in the error tables. Outlier detection is not applied to the VAWT experiment due to the limited dataset size, which does not allow a reliable statistical identification of outliers.

5.2.2 Vertical Axis Wind Turbine

In this experiment, we consider the vertical axis wind turbine (VAWT) dataset (Section 5.1.1) and predict the thrust force F_T and radial force F_R acting on a blade

over a full turbine revolution. For this test case, we use the simplified workflow, which consists of the encoding block and the mapping block only (see Section 4.1.3).

In the encoding block, a DeepSDF model with four hidden layers of 128 neurons, ReLU activations and a final tanh activation was trained for 2000 epochs. The learning rate was initialized at $5 \cdot 10^{-4}$ and halved every 200 epochs using a stepwise constant schedule. Latent codes were optimized with the same schedule, starting from 10^{-3} . The best results, in term of validation error, were obtained with a latent space of dimension 3.

As reported in Figure 5.18, the trained model achieves a mean reconstruction error of 4.4% on the training set and 6.7% on the validation set. For a qualitative assessment, we compare the airfoils reconstructed as the zero level set of the predicted SDF with the ground-truth geometries. As shown in Figure 5.19, even in the worst case the reconstructed surface is visually indistinguishable from the true geometry. This indicates that the learned latent space is able to effectively represent both training and validation airfoils.

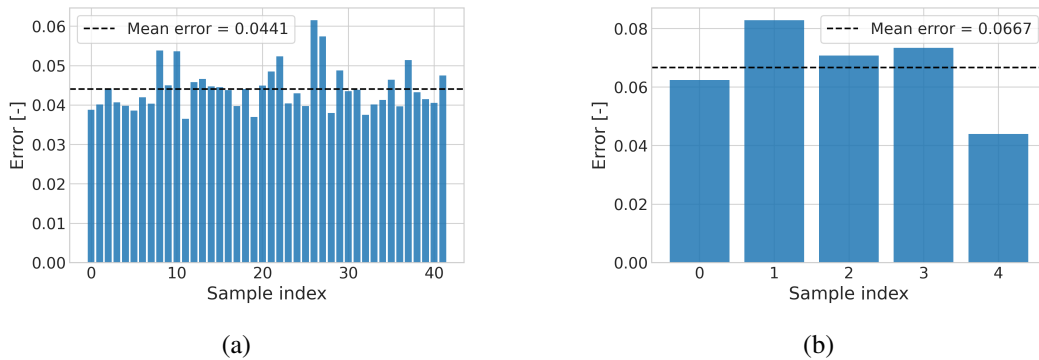


Fig. 5.18 Relative SDF prediction error on (a) training and (b) validation samples.

The model used to map the geometric encoding to the output signals is a feed-forward neural network with four hidden layers of size 256. It is trained to predict the forces acting on the blade as a function of the rotation angle. More specifically, the network takes as input the latent code representing the airfoil geometry together with the rotation angle and outputs the corresponding scalar target, i.e., either the thrust force or the radial force. Two separate models, sharing the same architecture, are trained for 2000 epochs each, using an initial learning rate of $3 \cdot 10^{-4}$ and a stepwise constant scheduler that halves the learning rate every 200 epochs.

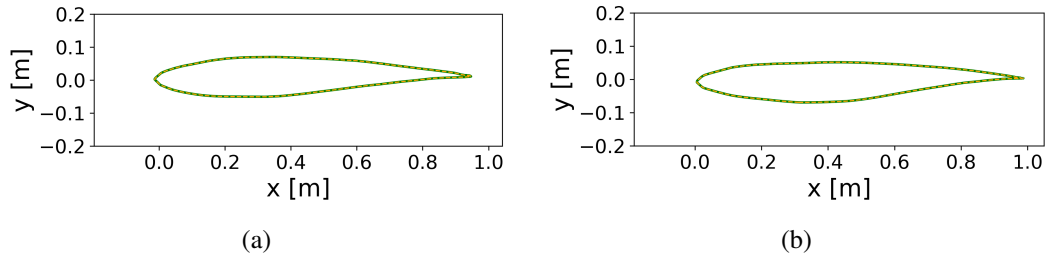


Fig. 5.19 Airfoil surface reconstructions of the (a) training and (b) validation cases corresponding to the largest error. The green line is used to show the ground-truth airfoil, while the encoded-decoded geometry is shown in orange. The two profiles almost completely overlap, indicating the high reconstruction accuracy even for the worst-case examples.

For each geometric latent code, the force is predicted for all values of the rotation angle in $[0^\circ, 360^\circ]$. The resulting predictions are then concatenated over the angle domain to reconstruct the full signal over a complete turbine revolution. Figure 5.20 shows the error distribution on the training dataset. The average prediction error is 0.28%, with a maximum error below 1%. In contrast, the mean validation error is substantially higher, reaching approximately 35% in the worst case. This gap suggests that, although the network accurately reproduces the training samples, its generalization capability is limited for unseen geometries. A likely contributing factor is the limited size of the available dataset relative to the complexity of the geometry-to-output mapping, which makes it difficult for the model to learn a representation that generalizes well beyond the training configurations. This behavior is also reflected in Figures 5.21 and 5.22, which compare the predicted and ground-truth signals for representative training and validation cases.

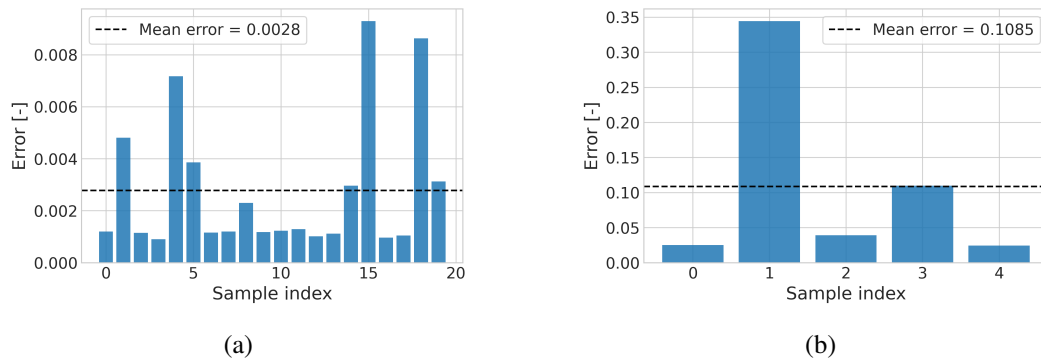


Fig. 5.20 F_T relative prediction error on (a) training and (b) validation samples.

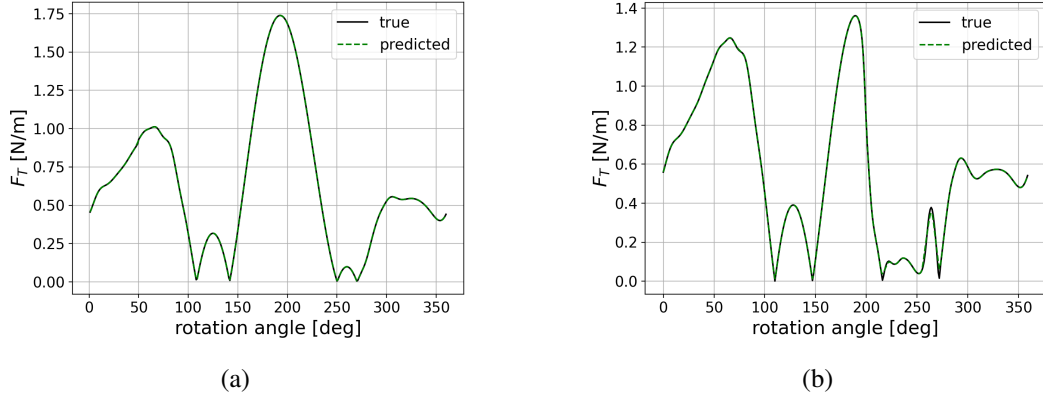


Fig. 5.21 F_T true and predicted signal for (a) average case and (b) worst case in the training dataset.

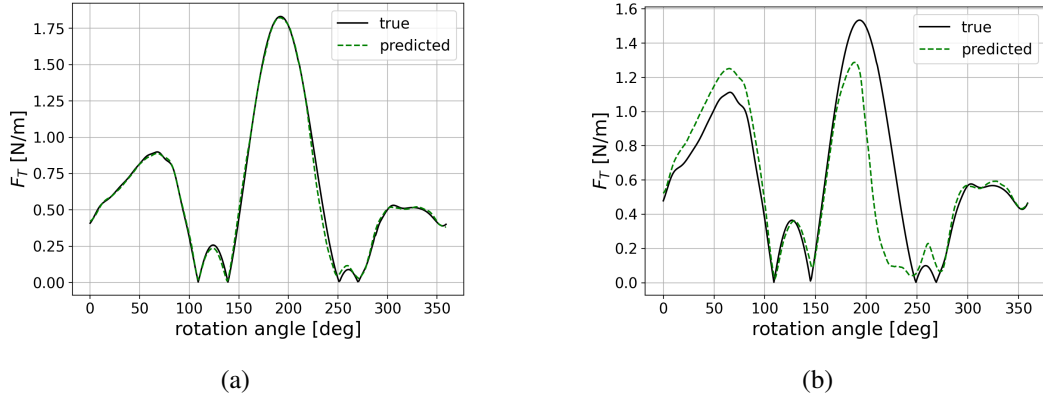


Fig. 5.22 F_T true and predicted signal for (a) average case and (b) worst case in the validation dataset.

Having established the baseline performance of the non-regularized workflow on this limited dataset, the effect of the proposed latent-space regularization is now investigated. In order to quantify the benefit of the two regularization methods, we retrained the geometry encoder using the same datasets and hyper-parameters. The only difference lies in the addition of a regularization term in the loss function, weighted by $\lambda = 0.1$. The value of λ was selected as the one yielding the lowest validation error and the corresponding tuning results are reported in Appendix B. After retraining the encoder with both regularization strategies, we analyzed the latent codes obtained with each method. In particular, pairwise distances between training geometries were computed in both the latent space and the SDF space, and the Pearson correlation coefficient r between these two quantities was evaluated. This analysis aims to assess whether the learned latent representation preserves the relative

geometric proximity between instances. As shown in Figure 5.23, the non-regularized model exhibits a negligible correlation between geometric and latent distances ($r = 0.06$), while FRD regularization provides only a limited improvement ($r = 0.24$). Conversely, PWD regularization leads to a substantial increase in correlation ($r = 0.99$), indicating that the resulting latent space closely preserves the pairwise geometric relationships among the training instances.

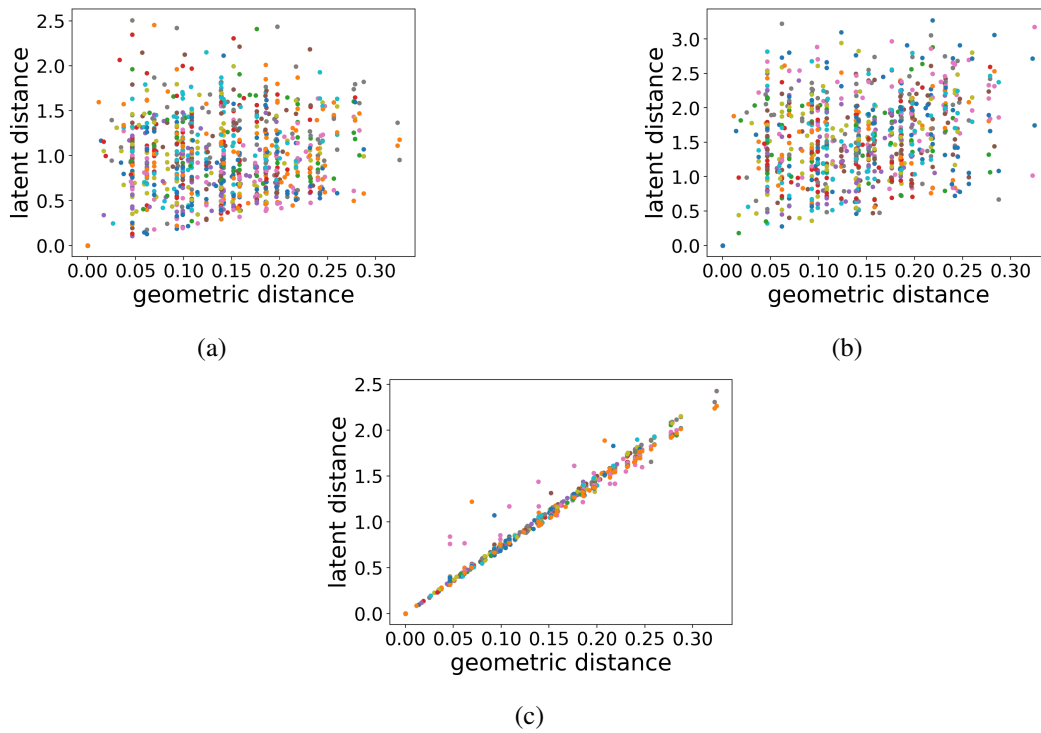


Fig. 5.23 Geometric distance vs. latent codes distance for all possible couples of training geometries. Comparison between (a) non-regularized DeepSDF, (b) FRD regularization and (c) PWD regularization.

Another interesting effect of the PWD regularization can be observed from the principal component analysis (PCA) of the learned latent codes. Figure 5.24 compares the explained variance distribution for the non-regularized and PWD-regularized latent spaces. While the non-regularized representation distributes the variance across multiple latent dimensions, the regularized latent space exhibits a more compact organization, with 99.64% of the variance explained by the first two PCA components.

This result indicates that PWD regularization promotes the concentration of geometric variability into a reduced number of latent directions, consistent with the bi-dimensional nature of the underlying geometric parameter space. However, the latent dimensionality adopted in the final workflow was kept equal to 3, as this configuration provided the best prediction performance during the hyper-parameter tuning phase. Although this dimensionality is expected due to the parameterization adopted to generate the dataset, it is not explicitly provided to the encoder during training. Therefore, the PCA analysis does not demonstrate the discovery of the original parametrization, but rather highlights the effect of the regularization in inducing a more compact and structured latent representation.

A qualitative comparison between the true parameter space and the regularized latent space is shown in Figure 5.25, where the power coefficient is visualized in both spaces to facilitate the comparison between their structures. The PWD-regularized latent space exhibits a spatial organization consistent with the original parameter space, despite the fact that the geometric parameters are not provided during training.

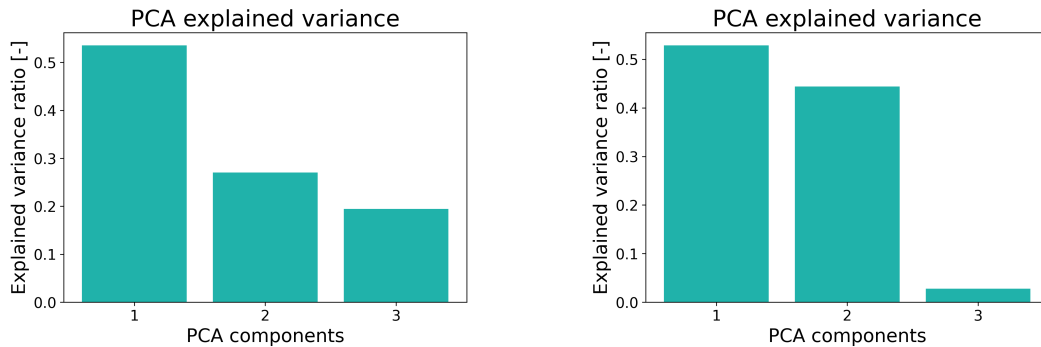


Fig. 5.24 Explained variance of the three PCA components computed on the three-dimensional latent codes obtained (a) without regularization and (b) with PWD regularization.

Figure 5.26 reports the SDF validation errors obtained with the two regularization strategies. Table 5.1 summarizes the SDF reconstruction errors for the original DeepSDF model and for the regularized versions. For unseen geometries, the prediction error associated with the FRD regularization is approximately 4.5 times higher than that of the non-regularized model. In contrast, the PWD regularization reduces the SDF reconstruction error by approximately 45% on both the training and validation datasets.

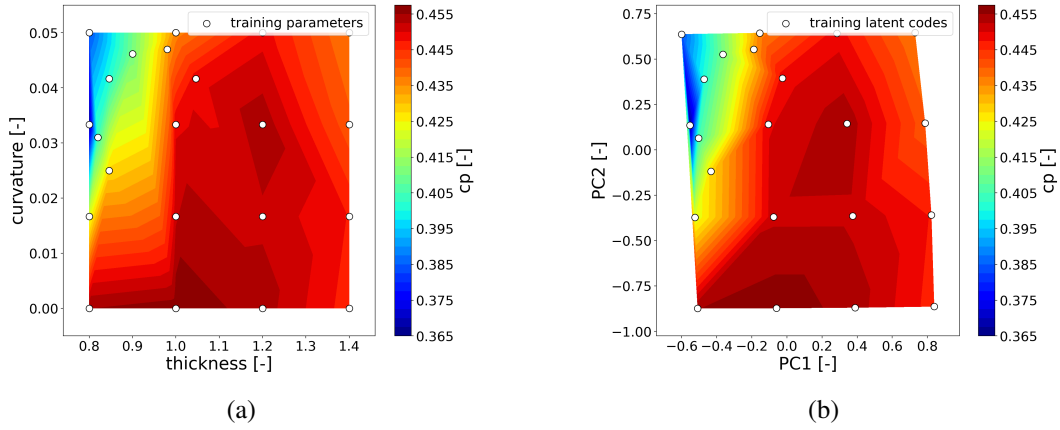


Fig. 5.25 Power coefficient on (a) true parameters space and (b) first two PCA components of regularized latent space. PCA components are rotated to facilitate the visual comparison.

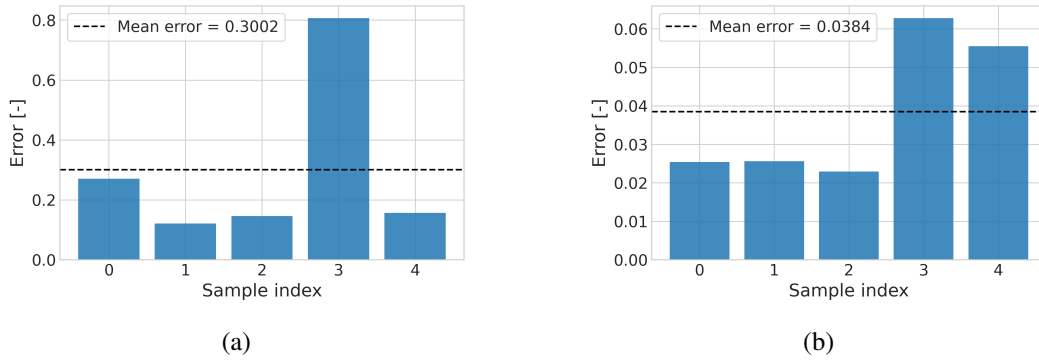


Fig. 5.26 Relative error on SDF predictions obtained with (a) FRD regularization and (b) PWD regularization.

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

Table 5.1 SDF reconstruction errors. Boldface denotes the best result both in terms of mean validation error.

Next, the mapping block was retrained using the regularized latent codes as input, together with the rotation angle. Relative errors in thrust force prediction are

reported in Figure 5.27. The results show that the average prediction error obtained with FRD regularization is comparable to that of the original model. In contrast, the PWD regularization leads to a substantial improvement, reducing the average error by approximately one order of magnitude.

Lastly, the results obtained with the PWD regularization are compared with more standard approaches, namely single- and multi-fidelity POD combined with Gaussian processes. In contrast to our approach, these models rely on an explicit, user-defined parametrization, as both regression models predict the POD coefficients from the shape parameters and the rotation angle. The signal is then reconstructed by projecting the predicted coefficients onto the POD basis computed during the offline training phase. This approach was applied in [58] to a test case involving the same wind turbine, considering a fixed geometry at varying rotational speeds. Further details on the multi-fidelity model and its coupling with POD can be found in [62].

Figure 5.28 compares the predictions obtained with the three models. Table 5.2 reports the prediction errors on F_T for the Proper Orthogonal Decomposition (POD) combined with Gaussian Processes (GP) models (single- and multi-fidelity) and for our approaches, namely the baseline SWF and its FRD- and PWD-regularized variants. The results show that, on this dataset, PWD regularization not only improves the overall accuracy of the workflow, but also that this approach outperforms standard methods that directly exploit the underlying geometric parametrization. The same trend is observed for the prediction of the F_R signal (see Appendix C).

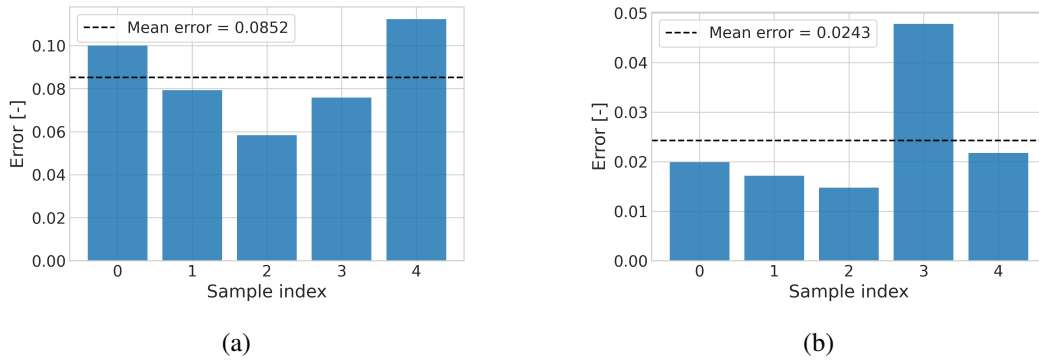


Fig. 5.27 Relative error on the prediction of the thrust force, F_T on the validation dataset obtained with the (a) FRD regularization, and (b) with PWD regularization.

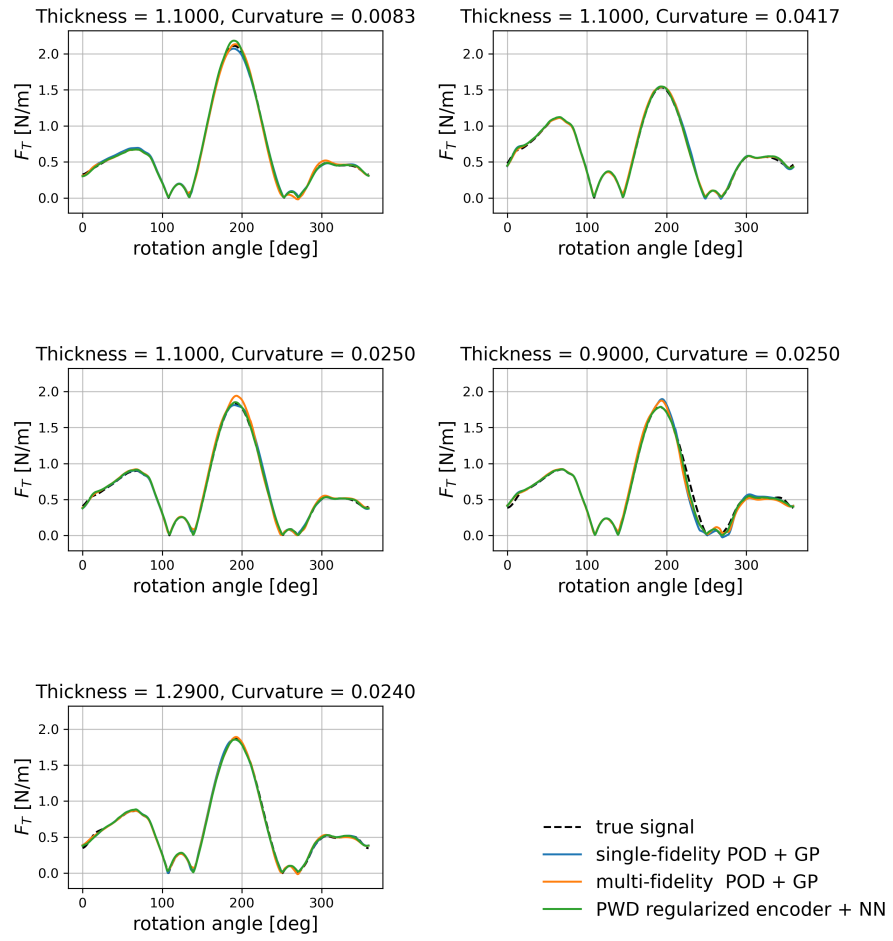


Fig. 5.28 Thrust force predictions obtained with regularized DeepSDF + NN, compared with single-fidelity and multi-fidelity POD + GP.

Method	Dataset	Mean Error	Standard Deviation
POD+GP single-fidelity	Training	0.0005	0.0006
	Validation	0.0357	0.0254
POD+GP multi-fidelity	Training	0.0007	0.0004
	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

Table 5.2 F_T prediction errors. Boldface denotes the best result both in terms of mean validation error and standard deviation.

5.2.3 Transonic RAE2822

In the second experiment, the goal is to predict the pressure coefficient C_p around an airfoil with variable shape. Geometric variations of the baseline RAE2822 airfoil are generated by applying deformation modes, as described in Section 5.1.2. For this case, we use the complete workflow introduced in Section 4.1, which consists of the geometric encoding, the output encoding and the mapping blocks. For the geometric encoding, a DeepSDF model with four hidden layers of size 128, ReLU activations, and a final tanh activation was trained for 2,000 epochs. The learning rate was initialized at $1 \cdot 10^{-4}$ for the network weights and $1 \cdot 10^{-3}$ for latent code optimization. Both learning rates were reduced by half every 200 epochs using a stepwise scheduler. The best results in terms of mean validation error were obtained with a latent dimensionality of 8. The error distribution for SDF prediction on the training and validation datasets is shown in Figure 5.29. The mean relative error is approximately 4% on the training set and 4.7% on the validation set, indicating strong representation capability.

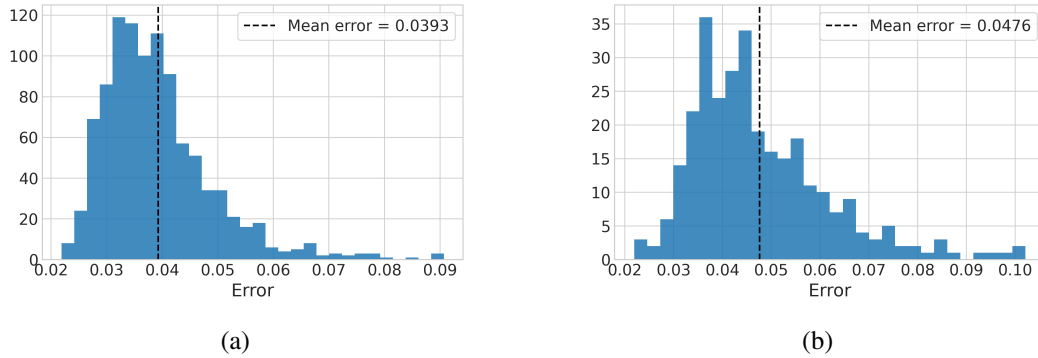


Fig. 5.29 Relative SDF prediction error on (a) training and (b) validation samples.

The output encoder adopts the same architecture as the geometric encoder described above. The only modification is the width of the hidden layers, which is increased to 256 neurons. In this block, i.e., the output encoding block, the spatial coordinates at which the pressure coefficient is defined are first transformed using the Fourier feature embedding method described in Section 4.1, with $L = 4$ frequencies. The model was trained for 2,400 epochs, with initial learning rates of $6 \cdot 10^{-4}$ for the network weights and $1 \cdot 10^{-3}$ for latent code optimization. Both learning rates were halved every 400 epochs. The reconstruction errors are reported in Figure 5.30, showing that the model accurately represents both training and validation pressure

coefficient fields, with average errors of 2.7% and 2.3%, respectively. Note that these errors are not representative of the final prediction accuracy, as they only reflect the reconstruction error associated with the latent representation. Furthermore, in this initial experiment no regularization is applied to either the geometric or the output encoder.

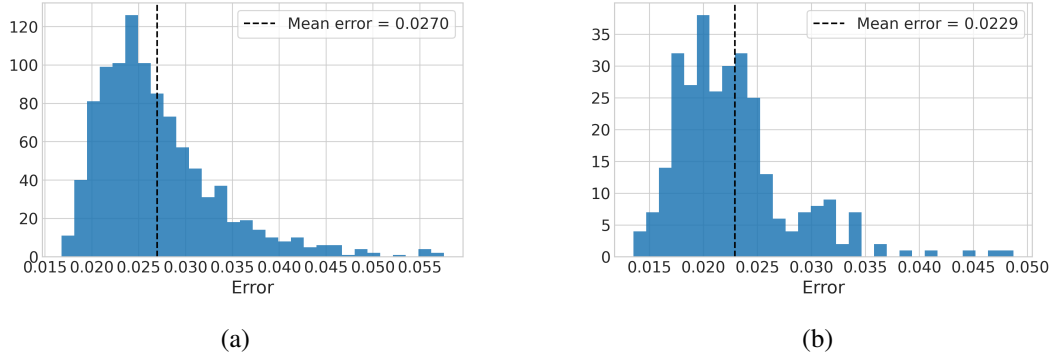


Fig. 5.30 Relative C_p representation error on (a) training and (b) validation samples.

At this stage, both the training geometries and the corresponding C_p fields are represented by latent vectors of size 8. In the mapping block, a neural network with four hidden layers of size 64 is trained to map the geometric latent space to the C_p latent space. By evaluating the trained model, a prediction of the target latent encoding is obtained for each training and validation airfoil. The predicted latent code is then decoded through the output encoder to reconstruct the C_p field at the desired spatial locations. The distribution of the prediction error is shown in Figure 5.31, where the error is computed as the difference between the reconstructed and ground-truth C_p fields. Both training and validation errors are approximately one order of magnitude higher than the corresponding reconstruction errors (Figure 5.30), reaching up to 50% in the worst cases. This indicates that, although the encoding blocks accurately represent both the geometric and solution spaces, the mapping model struggles to learn the relationship between the two latent spaces.

Therefore, the encoding blocks were retrained using the regularization strategies introduced in Section 4.2, while keeping the same training data and hyper-parameters. The tuning of the regularization weight is reported in Appendix B. A comparison between the two proposed approaches, namely FRD and PWD regularization, is shown in Figure 5.32. The non-regularized model exhibits a negligible correlation between geometric and latent distances ($r = 0.10$). FRD regularization significantly improves this correspondence ($r = 0.88$), while PWD regularization further increases

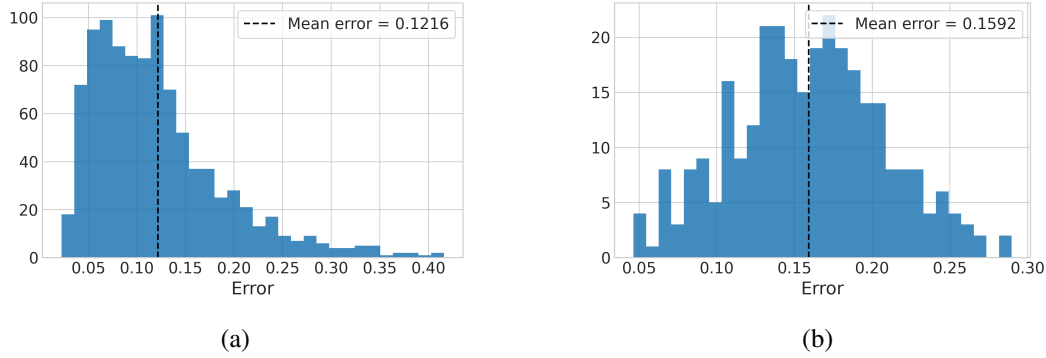


Fig. 5.31 Relative C_p prediction error on (a) training and (b) validation samples.

the correlation ($r = 0.99$). This difference can be attributed to the distinct constraints imposed by the two formulations: FRD enforces consistency between latent distances and geometric distances measured with respect to a common reference geometry, whereas PWD directly constrains the pairwise relationships between all geometry instances. Consequently, PWD provides a more complete preservation of the geometric relationships within the latent space. A similar behavior is observed for the output encoder, as shown in Figure 5.33. In this case, the regularization is directly applied to the field latent representation to promote a correspondence between distances in the output space and in the latent space. The non-regularized model already exhibits a moderate correlation between output and latent distances ($r = 0.41$). FRD regularization further improves this correspondence by enforcing consistency with respect to a common reference field ($r = 0.86$), while PWD provides an additional improvement by directly constraining the pairwise relationships between all field representations ($r = 0.99$). These results indicate that PWD regularization leads to a latent space whose distance relationships are closely aligned with those of the original output space.

Figures 5.34 and 5.35 show that both regularization strategies lead to a significant improvement in the reconstruction of unseen samples. In particular, the mean SDF reconstruction error is reduced by approximately one order of magnitude compared to the non-regularized encoding (see Figure 5.29b). This improvement can be attributed to the role of the regularization term in organizing the latent space according to the underlying structure of the input data. Without regularization, the latent codes are optimized mainly to minimize the reconstruction error of individual samples, which does not explicitly enforce any relationship between different geometries and their representations. As a consequence, the mapping between geometric and output latent

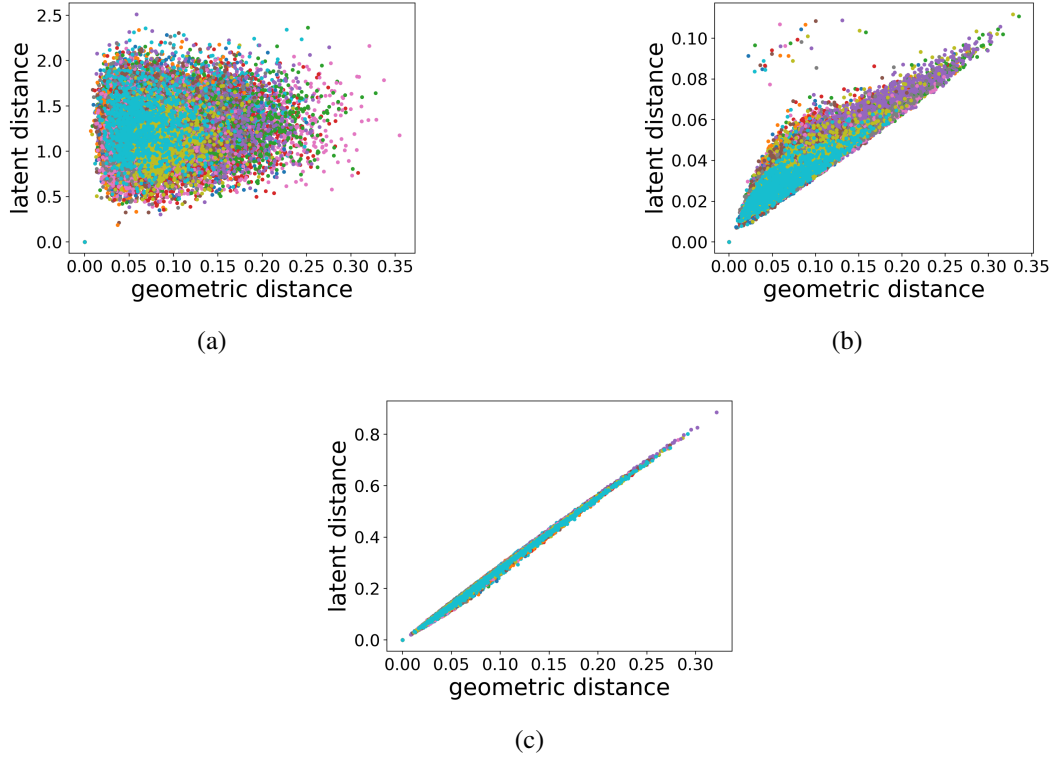


Fig. 5.32 Correlation between geometric distance and latent codes distance obtained by training the geometric encoding block (a) without regularization, (b) with FRD regularization and (c) with PWD regularization. Each color represents the distance of training samples from a fixed reference sample.

spaces may become more complex, reducing the ability of the model to generalize to unseen configurations. By constraining latent distances to reflect the original space relationships, the regularized models provide a more structured representation, facilitating the prediction of latent codes for unseen geometries and improving the final SDF reconstruction accuracy. Tables 5.3 and 5.4 report detailed error statistics for both the training and validation datasets, including the count and maximum value of the outliers omitted from the histograms.

The mapping model was retrained to map the regularized geometric latent space to the output latent space using both regularization strategies. The training and test losses are shown in Figure 5.36, where the training loss of the non-regularized model is also reported for comparison. FRD regularization substantially improves the mean loss on the test dataset, although a significant variability is still observed. In contrast, PWD regularization leads to a marked reduction in both the mean test error and the loss variability. This behavior is also reflected in the prediction error distribu-

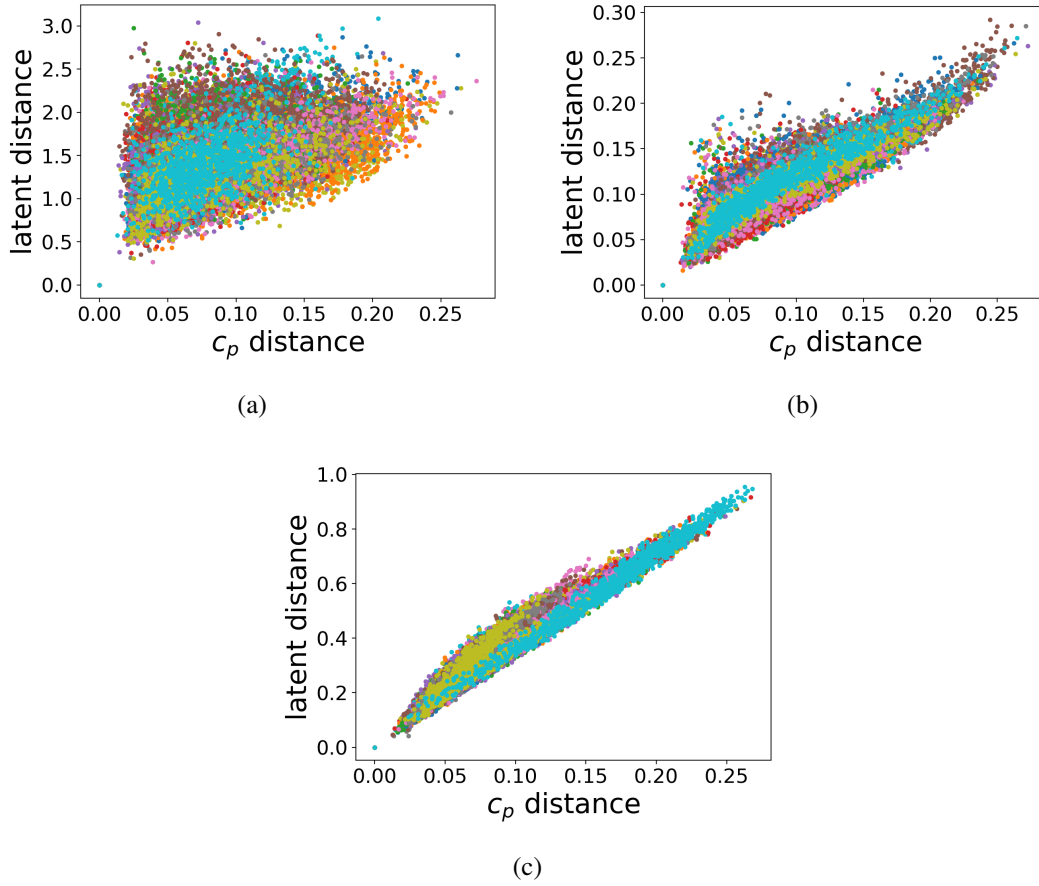


Fig. 5.33 Correlation between C_p distance and latent codes distance obtained by training the output encoding block (a) without regularization, (b) with FRD and (c) with PWD regularization. Each color represents the distance of training samples from a fixed reference sample.

tions shown in Figure 5.37. Both regularization methods reduce the mean error by approximately one order of magnitude compared to the non-regularized encoders (see Figure 5.31), with PWD exhibiting a considerably lower variability. Table 5.5 summarizes the C_p prediction errors on both training and validation datasets for the baseline workflow and the two regularized variants. Figure 5.38 shows the true, predicted and error fields for the best and median validation cases obtained with the non-regularized workflow. The corresponding results for the PWD-regularized workflow are reported in Figure 5.39. Notably, the median case with PWD regularization outperforms even the best case of the non-regularized model. Moreover, the PWD-based model achieves higher accuracy in the boundary layer, which is particularly challenging to predict and plays a key role in the evaluation of integral

Method	Dataset	Mean	Std	# 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 5.3 SDF reconstruction errors. Boldface denotes the best result both in terms of mean validation error.

Method	Dataset	Mean	Std	# 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	-

Table 5.4 C_p reconstruction errors. Boldface denotes the best result in terms of mean validation error.

quantities such as lift and drag coefficients. Additional C_p predictions for validation samples, obtained without regularization and with both regularization strategies, are provided in Appendix D.

Method	Dataset	Mean	Std	# 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

Table 5.5 C_p prediction errors. Boldface denotes the best result in terms of mean validation error.

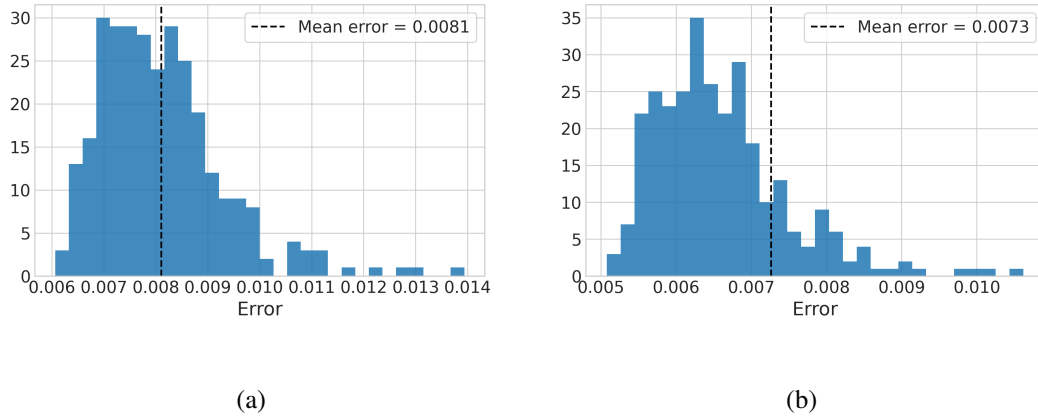


Fig. 5.34 Distribution of the relative error on SDF reconstruction on validation dataset obtained with (a) FRD and (b) PWD regularization.

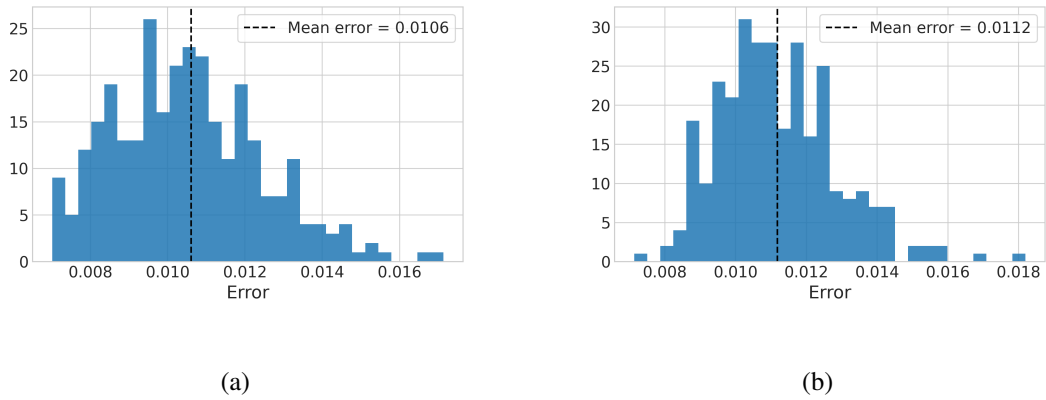


Fig. 5.35 Distribution of the relative C_p error on the validation dataset obtained with (a) FRD and (b) PWD regularization.

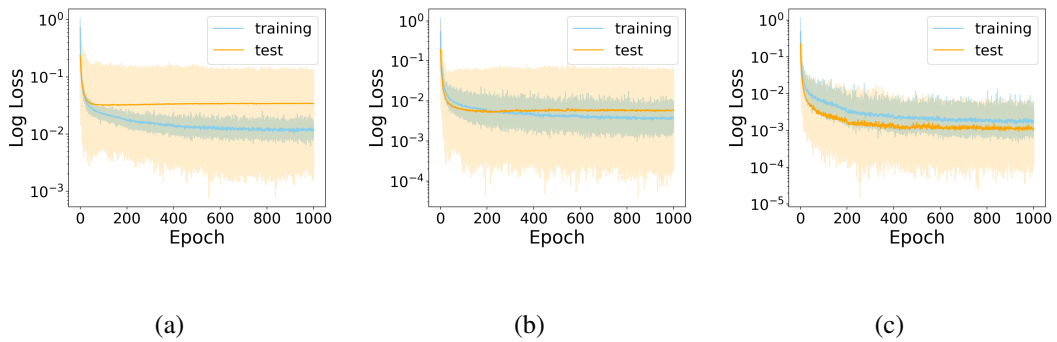


Fig. 5.36 Train and test loss during the training of the mapping block (a) without regularization, (b) with FRD regularization and (c) with PWD regularization. Shaded areas represent the variation of the loss across batches within each epoch.

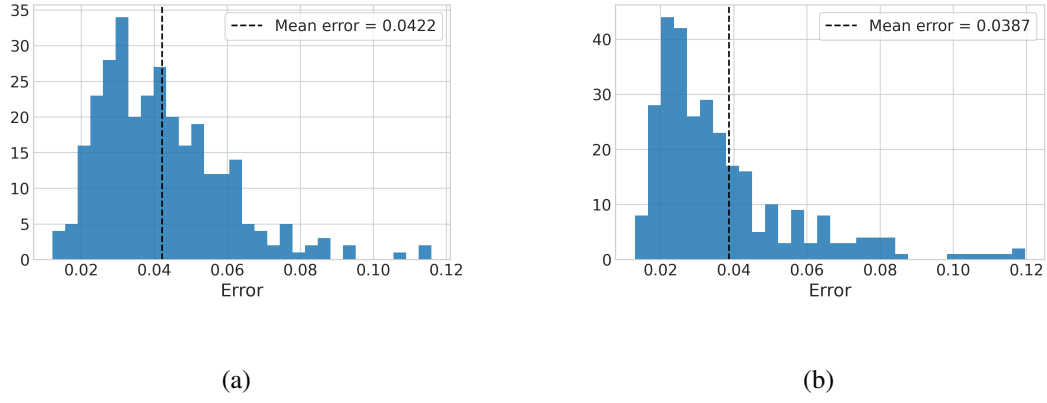


Fig. 5.37 Distribution of the relative prediction error on C_p for the validation dataset obtained with (a) FRD and (b) PWD regularization.

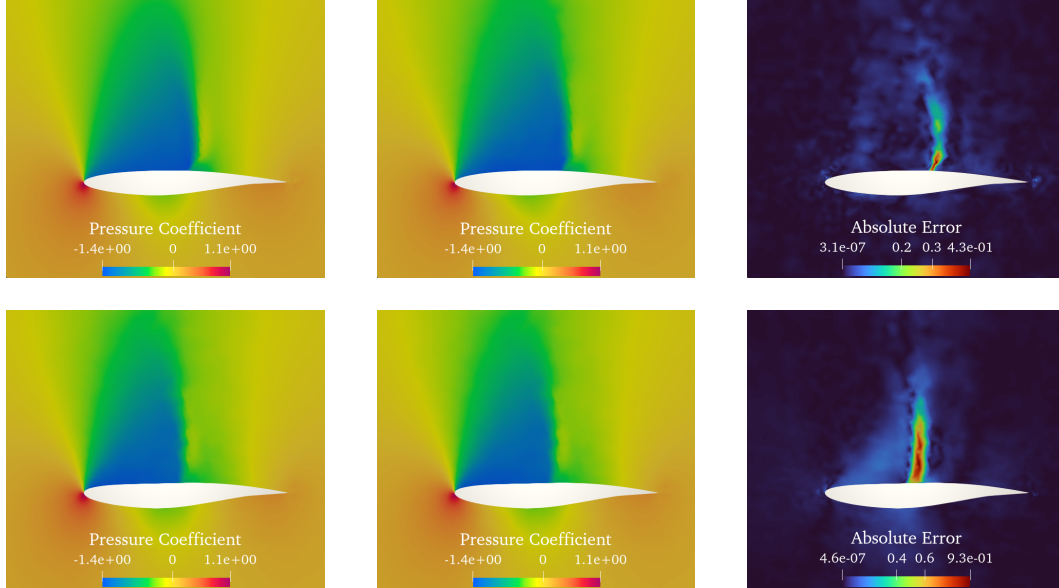


Fig. 5.38 Comparison of true and predicted C_p fields obtained with the non-regularized workflow. From top to bottom: best and median prediction cases. From left to right: ground-truth field, predicted field and absolute error.

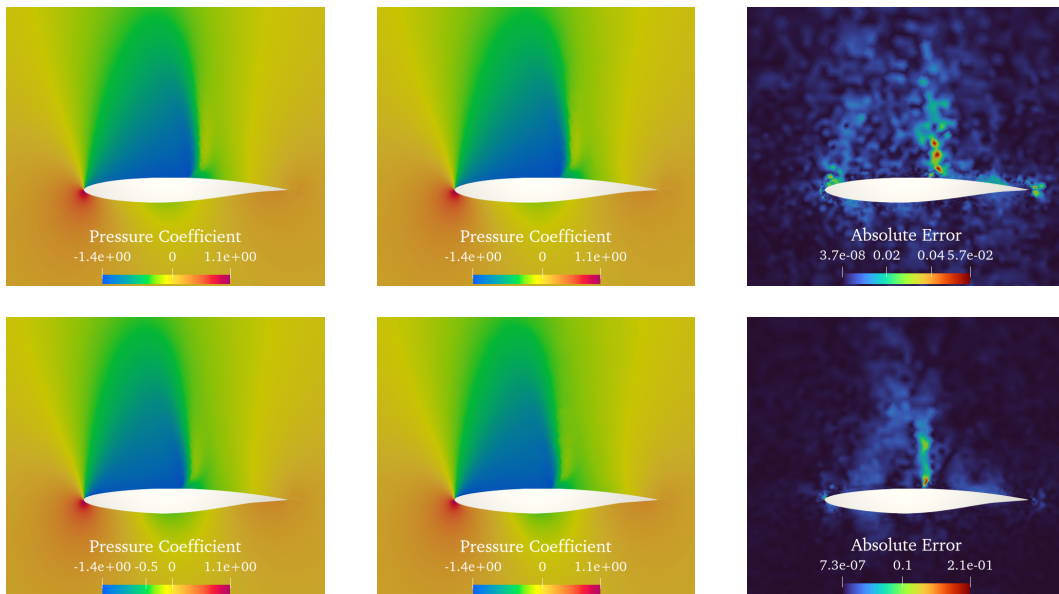


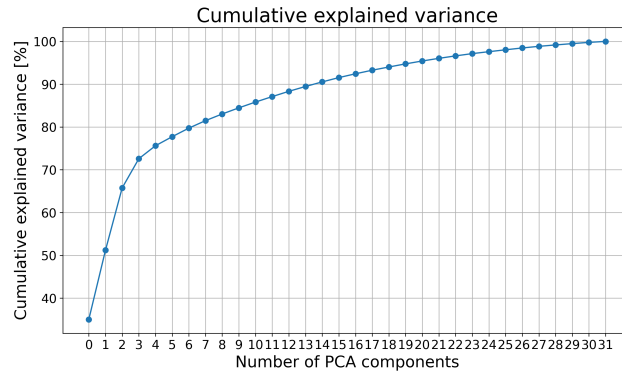
Fig. 5.39 Comparison of true and predicted C_p fields obtained with the PWD-regularized workflow. From top to bottom: best and median prediction cases. From left to right: ground-truth field, predicted field and absolute error.

5.2.4 DrivAerML

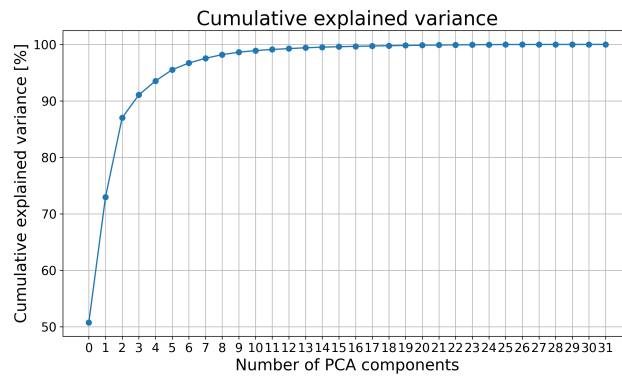
In this test case, we consider the DrivAerML dataset described in Section 5.1.3 and aim to predict both drag coefficient and the pressure field on the vehicle body surface, excluding wheels. For this test case, the simplified workflow is adopted for the prediction of the scalar quantity, since no spatial distribution needs to be encoded. The same workflow is also used for the surface pressure field, which is defined only on the vehicle surface and exhibits localized variations near sharp geometric features. These local variations make the prediction task more challenging, as they require the model to capture small-scale features of the field. However, the overall mapping between geometric variations and the corresponding pressure distribution remains sufficiently regular to be effectively learned through a direct formulation. As a beneficial consequence, removing the output encoding block also reduces the computational cost of the training process. This aspect is particularly relevant in three-dimensional settings, where achieving an adequate field resolution requires a significantly larger number of sampled points compared to two-dimensional cases.

To assess the effect of PWD regularization, two geometric encoders are trained using the same architecture and training hyper-parameters, with the only difference being the inclusion of the PWD regularization term in the loss function. The dataset was randomly split into training and validation sets using an 80/20 ratio. The geometric encoders, consisting of five hidden layers of size 512, were then trained for 2,000 epochs using identical architectures and training hyper-parameters, except for the regularization term. The learning rates were initialized at $5 \cdot 10^{-4}$ for the network weights and $1 \cdot 10^{-3}$ for latent code optimization and halved every 500 epochs using a stepwise scheduler. The latent space dimensionality was fixed at 32.

An interesting effect of the regularization can be observed from the PCA of the learned latent spaces. Figure 5.40 shows the cumulative explained variance, which increases more slowly for the non-regularized latent space, indicating that the variance is distributed across a larger number of principal directions. This observation indicates that, being the latent dimensionality fixed, the standard DeepSDF model does not explicitly promote a compact representation and instead exploits the full latent space dimensionality to encode the geometric variability. In contrast, the cumulative explained variance for the regularized latent space grows significantly faster, indicating that 99.9% of the information is concentrated in 16 principal components. This behavior is consistent with the action of PWD regularization,



(a)



(b)

Fig. 5.40 Cumulative explained variance resulting from PCA on (a) non-regularized and (b) PWD-regularized latent space.

which enforces the relative positioning of latent codes and promotes a representation that reflects the structure of the source geometric space. While this effect can be directly visualized in the two-dimensional VAWT test case (Figure 5.25), it cannot be observed visually in the present case due to the higher dimensionality of the parameter space. Nevertheless, the PCA results suggest that the PWD-regularized latent space can be reduced to 16 dimensions while preserving 99.9% of the explained variance of the original latent representation. Notably, this behavior is already observed after few training epochs, as shown in Figure 5.41. This observation has a practical implication: combining PWD regularization with PCA provides a computationally efficient strategy to estimate the intrinsic dimensionality of the latent space. In contrast to the standard approach, this method enables a more direct selection of the latent dimensionality without extensive hyperparameter tuning, which requires (re)training multiple models with different latent space sizes.

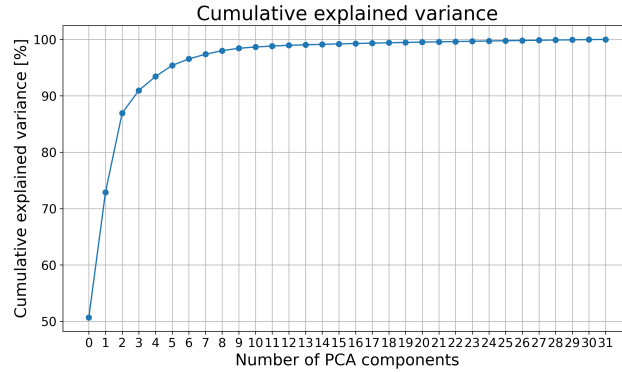


Fig. 5.41 Cumulative explained variance resulting from PCA on PWD-regularized latent space after 100 training epochs.

Based on these results, the geometric encoder was re-trained with a latent size of 16. Figure 5.42 shows the distribution of the relative SDF prediction error on the validation dataset. The mean error is approximately 9%, with a maximum value below 14%, which is slightly higher compared to the previous test cases. In Table 5.6 are summarized the SDF error statistics of training and validation datasets. To better interpret the magnitude of this error, the SDF was evaluated at all vertices of the surface mesh for each validation geometry. At these locations, the true SDF is zero by definition and the predicted value provides a direct measure of the deviation between the reconstructed surface and the ground-truth geometry. Figure 5.43 shows the distribution of the absolute predicted SDF on the surface of two validation geometries, corresponding to the median and worst prediction cases. The maximum absolute SDF is approximately 0.01 m in both cases, which is small compared to the overall vehicle lengths (4.7 m and 4.46 m, respectively). Moreover, the error is mainly concentrated in localized regions, such as behind the door handle and around the wheels, which are not directly affected by the applied deformations. This behavior indicates that the reconstruction error is not primarily related to the deformation process itself, but rather to the difficulty of encoding complex local geometric features. In this test case, the applied deformations are not localized but rather affect the geometry at a scale comparable with its overall dimensions. Therefore, the number of sampled points used to represent the SDF provides sufficient information to capture the deformation patterns. However, the observed deviations in fixed regions suggest that the sampling strategy may be inappropriate to fully represent small-scale geometric details. Although this limitation does not significantly affect the present test case, since these regions are not modified by the applied deformations, it may become

relevant for problems in which the target variations are localized or occur at a smaller scale. In such cases, increasing the number of SDF samples, particularly in regions of interest, could improve the ability of the implicit representation to capture fine geometric features.

Overall, the results indicate that the encoding model accurately captures the relevant geometric variations and provides a sufficiently precise representation for the subsequent prediction task.

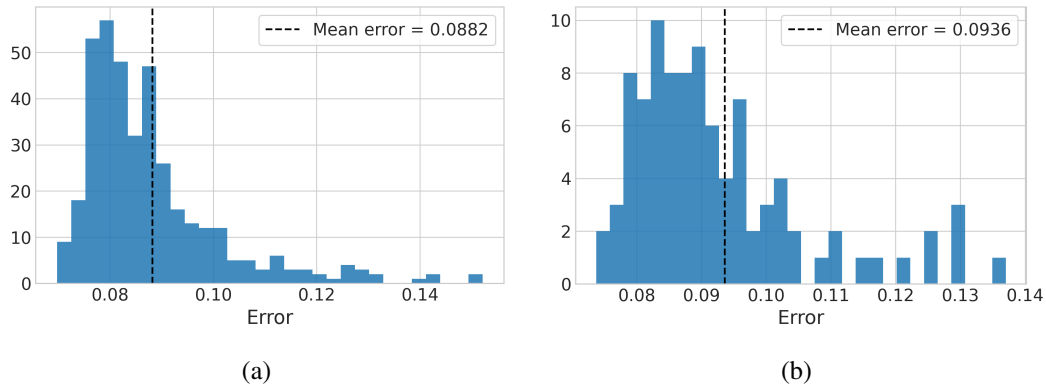


Fig. 5.42 Relative SDF prediction error on (a) training and (b) validation dataset.

Dataset	Mean Error	Std	# Outliers	Max Outlier
Training	0.0882	0.0149	2	0.1809
Validation	0.0935	0.0158	0	-

Table 5.6 SDF prediction error statistics on training and validation dataset.

The learned latent codes are used for two prediction tasks: the estimation of the drag coefficient C_D and the reconstruction of the surface pressure field on the vehicle body. For the first task, a neural network is trained to map the geometric latent codes to the corresponding C_D values. To assess the effect of PWD regularization, two identical models are trained using the same architecture and training hyper-parameters, with latent codes obtained with and without PWD regularization. The network consists of two hidden layers of size 16 and is trained for 1000 epochs with an initial learning rate of 10^{-2} , halved every 200 epochs. To monitor generalization performance, the training dataset is further split into training and test. The resulting training and test cases were used for both models, with and without PWD regularization, to ensure a fair comparison. Figure 5.44 shows the evolution of the

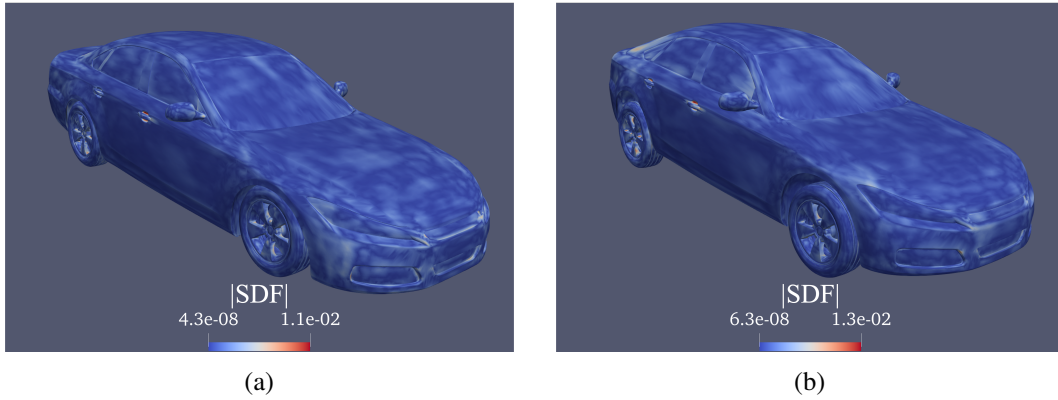


Fig. 5.43 Absolute value of the predicted SDF on surface for the geometries associated to the (a) median and (b) worst prediction error.

loss over the training epochs for both sets. The model based on non-regularized latent codes exhibits a highly noisy training loss and a relatively large gap between training and test performance, indicating unstable learning and limited generalization capability. In contrast, the model trained on PWD-regularized latent codes shows a smoother convergence and reduced variability, suggesting a more structured latent space that facilitates the learning of the mapping to C_D .

After training, the prediction error on unseen geometries was computed using Equation 5.8 and its distribution is shown in Figure 5.45. The model based on regularized latent codes yields slightly lower mean and maximum errors compared to the non-regularized counterpart. The overall similarity between the two approaches is further illustrated in Figure 5.46, which compares the true and predicted C_D values obtained with both models. The limited difference between the two approaches suggests that, for this specific prediction task, the additional latent-space constraint provides only a marginal benefit. This may be related to the lower complexity of the mapping between the geometric variations considered in this test case and the scalar drag coefficient, which is already captured with sufficient accuracy by the non-regularized model. Table 5.7 summarizes the errors obtained with the two models on training and validation datasets.

To predict the surface pressure field, the simplified workflow described in Section 4.1.3 is adopted. In this setting, the geometric latent code is concatenated with the spatial coordinates of a point on the vehicle surface and used as input to a feed-forward neural network, which predicts the pressure value at the queried surface location. For the prediction of the surface pressure field, wheel regions were

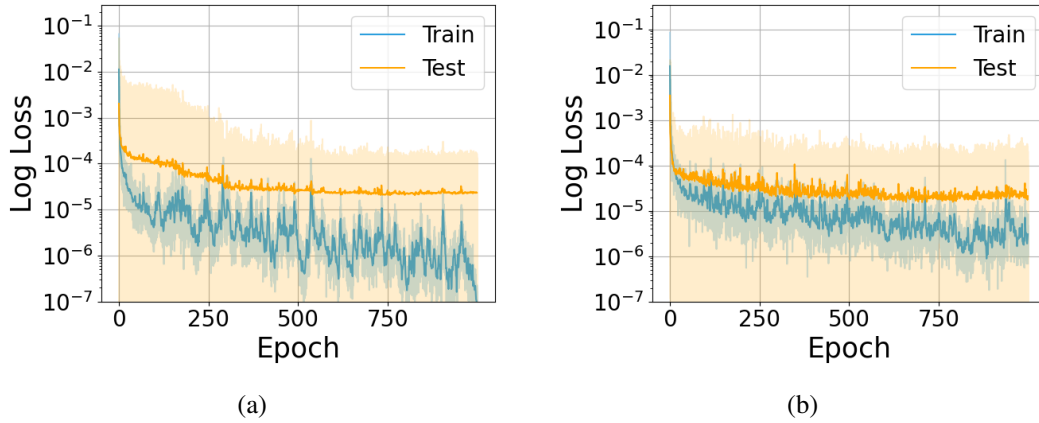


Fig. 5.44 Training and validation loss of the C_D prediction model based on (a) non-regularized and (b) PWD-regularized geometric encoding. The shaded regions indicate the range of loss values across batches within each epoch. Their lower extent is clipped for visualization purposes to improve the visibility of the mean loss curves.

Method	Dataset	Min	Max	Average
DeepSDF	Training	4.5×10^{-7}	6.7×10^{-4}	0.038
	Validation	3.7×10^{-4}	0.01	0.07
PWD reg.	Training	5.8×10^{-5}	3.2×10^{-3}	0.043
	Validation	4.8×10^{-4}	0.01	0.06

Table 5.7 C_D prediction errors.

excluded from the analysis. Accurately resolving these components would require a significantly higher number of surface points, resulting in increased computational cost and memory requirements during training. Given the available hardware resources, this additional cost was not affordable. Moreover, since the wheels remain unchanged throughout the dataset, their inclusion was not necessary to assess the ability of the proposed framework to predict the effect of the geometric variations on the pressure field.

The simplified mapping block, composed of six hidden layers of size 512, was trained for 2000 epochs. The same training and validation samples used for the geometric encoder were considered. For the mapping block, the training set was further split to define an internal test set, which was used to monitor the generalization behavior during training. This additional evaluation is not performed for the encoding networks, since assessing their performance on unseen geometries requires

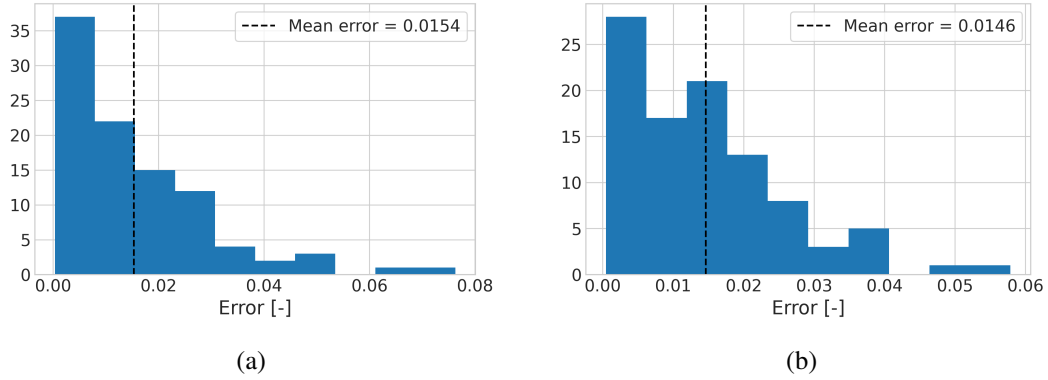


Fig. 5.45 C_D relative prediction error on unseen geometries obtained with (a) non-regularized and (b) PWD-regularized geometric encoding.

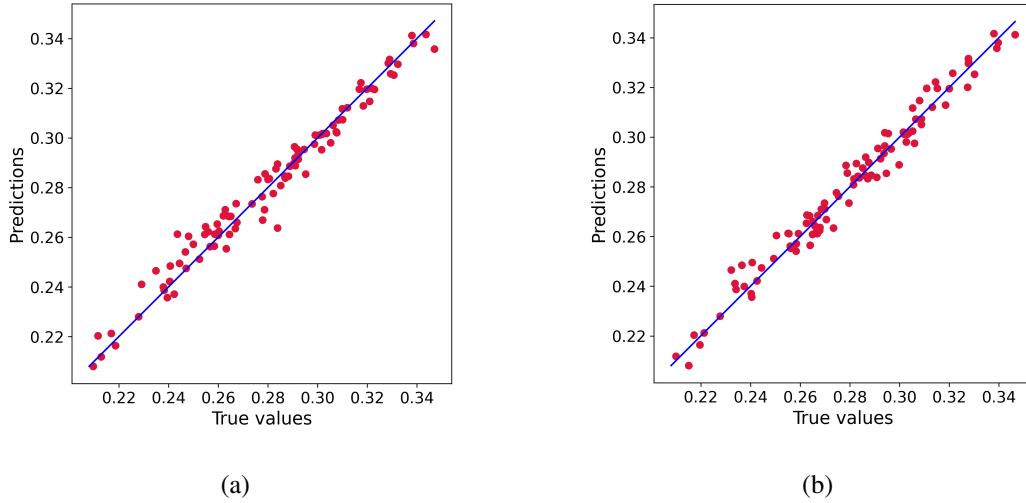


Fig. 5.46 True vs predicted C_D for unseen geometries obtained with (a) non-regularized and (b) PWD-regularized geometric encoding.

a separate latent-code optimization procedure, which would introduce a significant computational overhead during training. In Figure 5.47 we can observe that the test loss remains slightly higher than the training loss throughout the training process. However, both curves follow a similar trend throughout the training process, with no evident divergence between the training and validation losses. The variability of the test loss is higher, as expected, but does not suggest significant overfitting. After training, the model was evaluated by predicting the pressure field for each validation geometry. For this purpose, a random set of 50,000 surface points (matching the size of the training point clouds) was sampled for each geometry. The corresponding

geometric latent code was then concatenated with the spatial coordinates and used as input to the trained mapping block, which predicts the pressure value point-wise.

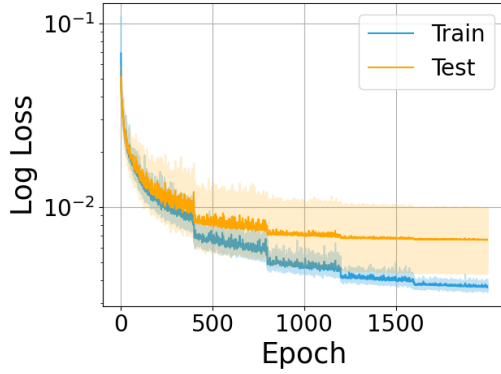


Fig. 5.47 Training and validation loss of the surface pressure prediction model. The shaded regions indicate the range of loss values across batches within each epoch.

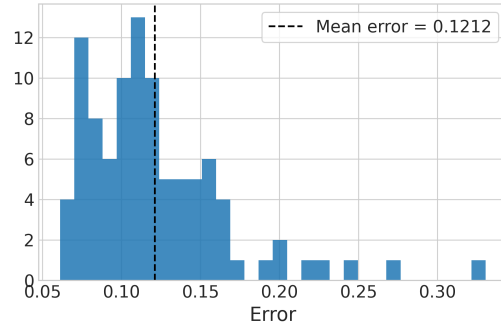


Fig. 5.48 Relative pressure prediction error on unseen geometries.

Figure 5.48 shows the distribution of the relative error across the validation dataset. The mean error is approximately 12%, with a few outliers exceeding 20%. This level of accuracy is satisfactory considering the complexity of the task, which involves predicting a three-dimensional field from a limited dataset of 309 samples and a 16-dimensional parameter space. The error statistics of training and validation datasets are summarized in Table 5.8.

Dataset	Mean Error	Std	Max	# Outliers
Training	0.0623	0.0273	0.1951	0
Validation	0.1212	0.0461	0.3305	0

Table 5.8 Surface pressure prediction error statistics on training and validation dataset.

In order to qualitatively assess the prediction error, Figure 5.49 shows the ground-truth pressure field, the corresponding prediction and the absolute error field on the surface mesh, for the best, median and worst validation case. The error field is computed by predicting the pressure at the cell centers of the surface meshes for all validation geometries, excluding the wheel regions, which are not considered in this prediction task. Overall, the predictions exhibit very good agreement with the ground-truth fields and even in the worst case the two are visually difficult

to distinguish. The largest errors are localized in regions characterized by sharp geometric features, where the pressure field presents steep gradients. However, these regions are limited in spatial extent and therefore have a reduced impact on the overall prediction quality. Note that, for better clarity, the error field has been clipped to a maximum value of 100. Error reaches higher values up to the order of 10^3 in localized regions associated with sharp geometric features of the vehicle surface. This behavior is likely due to the presence of very steep gradients in these areas.

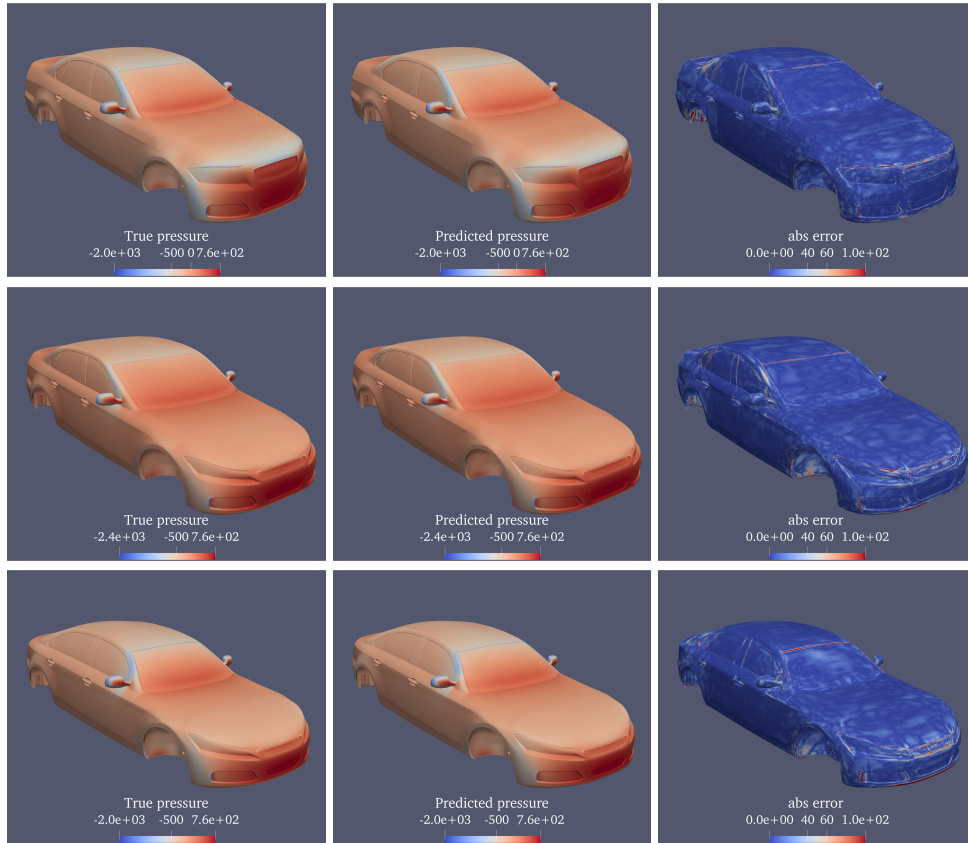


Fig. 5.49 Comparison of true and predicted surface pressure fields for three representative validation geometries. From top to bottom: best, median and worst prediction cases. From left to right: ground-truth field, predicted field and absolute error.

To evaluate the impact of these localized errors on integral quantities, the drag coefficient was computed from both the true and predicted pressure fields. More precisely, we consider only to the pressure drag $c_{D,p}$, evaluated without the wheels. Figure 5.50 shows the distribution of the relative prediction error on the training and validation datasets, while Figure 5.51 compares the true and predicted values of $c_{D,p}$. The mean error on unseen geometries is below 2%, with a maximum error of

approximately 5%. These results indicate that, despite the presence of high localized errors in the pressure field, their impact on integral quantities remains limited. Moreover, these results are comparable to those obtained by directly predicting C_D from the geometric latent codes (see Figure 5.45), suggesting that the proposed approach is able to preserve the information required to accurately predict the drag coefficient even when the learning task is formulated in terms of the pressure field rather than a directly integrated quantity.

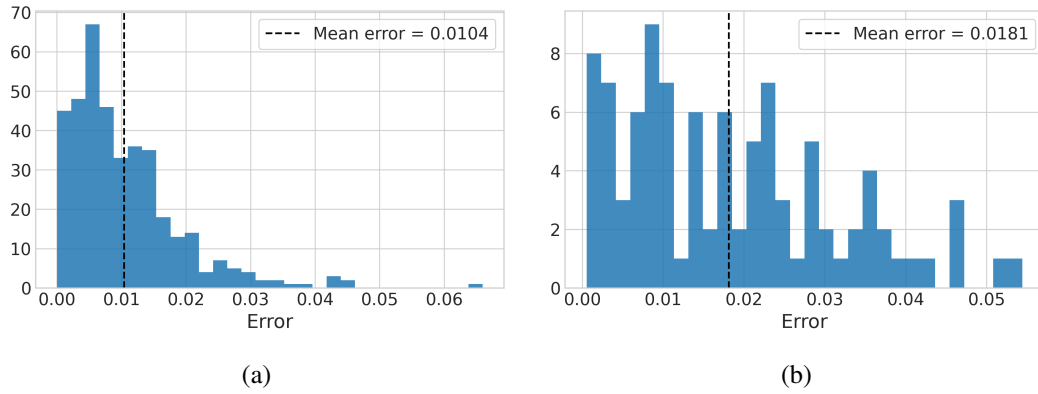


Fig. 5.50 Relative $c_{D,p}$ computed from true and predicted surface pressure. Distribution for (a) training and (b) validation dataset.

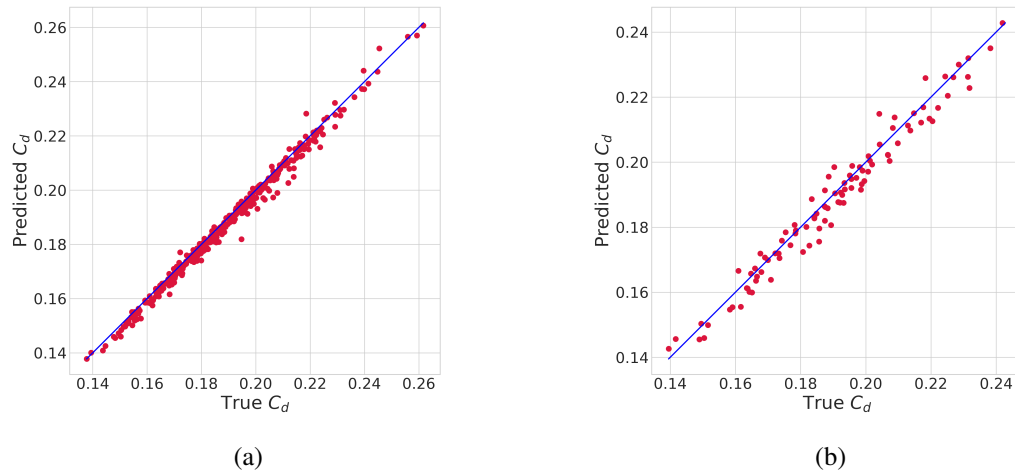


Fig. 5.51 $c_{D,p}$ computed from true vs predicted surface pressure for (a) training and (b) validation dataset.

Chapter 6

Conclusions

6.1 Summary of the Work

In this work, we proposed a modular deep-learning-based framework for the prediction of solutions to partial differential equation (PDE) and derived physical quantities on parametrized domains. The key idea is to encode both geometry and associated solution fields into fixed-size latent representations using a specialized implicit neural representation (INR)-based encoder. The two latent spaces are then mapped through a feed-forward neural network.

In addition to the full pipeline, we introduced a simplified workflow, where the mapping is performed directly from the geometric latent code to the quantity of interest evaluated at arbitrary spatial (and temporal) locations, removing the need for an explicit output encoder. Furthermore, we proposed two regularization strategies for INR-based encoders aimed at preserving geometric structure in the latent space, enforcing consistency between distances in the input geometry space and their latent representations.

6.2 Key Results and Experimental validation

The proposed framework has been evaluated on three different test cases of increasing complexity, covering both one-dimensional and multi-dimensional physical problems. The first experiment considers the prediction of one-dimensional signals, specifically

radial and thrust force evolution on a vertical-axis turbine blade under varying thickness and curvature. The second test case focuses on two-dimensional pressure coefficient fields over parametric deformations of the transonic RAE2822 airfoil. Finally, the framework has been tested on a three-dimensional problem involving the prediction of surface pressure distributions on a car geometry subject to controlled shape variations.

Across all experiments, the results demonstrate the effectiveness of the proposed workflow and the latent-space regularization strategies. In particular, the Pairwise Distance regularization (PWD) improves generalization performance, leading to more accurate predictions on unseen geometries. Across the considered test cases, the average relative error is consistently reduced, reaching improvements of up to one order of magnitude compared to the non-regularized model.

A further analysis of the learned latent representations, conducted via Principal Component Analysis (PCA), shows that the number of principal components required to describe the regularized latent spaces is consistent with the intrinsic dimensionality of the underlying geometric parameter space. In particular, embeddings obtained with PWD regularization can be effectively projected onto a lower-dimensional subspace whose dimensionality is equal to the true number of underlying geometric degrees of freedom, while preserving predictive performance with negligible loss of information. This behavior suggests that the proposed regularization not only improves predictive accuracy, but also promotes a latent space organization that better reflects the structure of the input data.

Interestingly, this structure emerges already in the early stages of training. From a practical standpoint, this property can be leveraged to guide the selection of an appropriate latent space dimensionality. In particular, the PCA spectrum of a high-dimensional PWD-regularized embedding can provide a criterion to identify an effective reduced representation, based on the variance explained by each principal component.

6.3 Limitations and future work

Implicit Neural Representations (INRs) provide a flexible and powerful framework for mesh-free geometry encoding. However, their performance is sensitive to the

choice of spatial sampling strategy. In this work, we adopted the sampling procedure introduced by Park et al. [38], which increases point density in the vicinity of the surface in order to better capture the zero-level set. While generally effective, this approach introduces several hyper-parameters that must be carefully selected, including the number of surface samples, the variance of the Gaussian perturbations, and the direction of the perturbation (normal or random) during the training of the geometric encoder.

From a broader perspective, several extensions of the proposed framework can be considered. A first natural direction concerns the inclusion of non-geometrical, PDE-related parameters. Although this work focuses on shape variations, the modular structure of the architecture allows for a straightforward extension by augmenting the input representation with additional physical or operational parameters. Nevertheless, a systematic study is required to assess how such augmentations affect latent space structure and prediction accuracy. Another relevant extension concerns the scalability of the proposed framework to three-dimensional volumetric fields prediction. Although the proposed workflow is formulated independently of the spatial dimensionality and has been successfully applied to 3D surface prediction, extending it to volumetric fields requires a suitable increase in the number of points used to represent the field. In particular, the increased dimensionality leads to a substantial growth in the computational and memory requirements of INR-based encoding and inference. As a consequence, although the formulation is general, its practical applicability to large-scale 3D volumetric simulations, such as the three-dimensional velocity or pressure field within the fluid domain, is ultimately constrained by computational resources. Dedicated studies are therefore required to assess the performance of the framework in this scenario.

Beyond methodological extensions, the proposed framework is particularly well suited for applications requiring fast and accurate evaluations of PDE solutions across parameterized geometries, such as shape optimization and design space exploration. In this context, the ability to reconstruct unsteady flow fields is a key enabling factor for more advanced downstream tasks. For instance, in canonical aerodynamic configurations such as the RAE2822 airfoil, accurate reconstruction of unsteady pressure fields represents a fundamental prerequisite for data-driven approaches to complex phenomena, including transonic buffet-induced aerodynamic noise, as investigated in recent studies [63, 64]. In this direction, future work could benchmark the proposed framework against state-of-the-art data-driven models used

for aerodynamic noise prediction, providing a systematic comparison in terms of accuracy, efficiency and generalization capabilities using problem-specific evaluation metrics. Finally, future work could investigate the integration of this framework into active flow control pipelines. In such settings, surrogate models must capture the dynamic response of the flow to control actions such as blowing, suction or surface actuation. By enabling efficient reconstruction of unsteady fields and time-dependent signals, the proposed approach could support almost-real-time control strategies, accelerate the evaluation of control effectiveness across varying geometries, and ultimately be embedded within optimization-based closed-loop control frameworks.

6.4 Strengths and Contributions

The proposed workflows are characterized by a combination of modularity, generality, and mesh-free representation, which collectively enable flexible learning of PDE solutions across varying parametrized domains. In contrast to monolithic end-to-end approaches, the modular design allows individual components to be reused across different tasks without requiring full retraining of the entire pipeline. This becomes particularly advantageous in data-intensive scenarios, where training a single unified model would incur substantial computational cost.

A further key characteristic of the framework is its data-driven, parameter-agnostic nature. The model does not rely on explicit knowledge of the parameters governing the geometric variations, nor on the specific mechanism used to generate them. Instead, it learns a compact representation directly from observed data, making it applicable to a wide range of scenarios where such parameters are unknown or not explicitly available.

Moreover, the use of INRs removes the dependency on a fixed spatial discretization. As a consequence, the learned model operates in a continuous domain and can, in principle, provide predictions at arbitrary spatial resolution.

The main methodological contribution of this work lies in the proposed regularization strategies for INR-based encoders. In particular, the Pairwise Distance (PWD) regularization enforces consistency between geometric distances in the input space and their corresponding representations in the latent space. This leads to a

more structured embedding space that better preserves the intrinsic geometry of the data.

As demonstrated by the numerical experiments, this improved latent space organization translates into enhanced generalization performance of the downstream mapping model, resulting in significantly more accurate predictions on previously unseen geometries. Beyond performance improvements, the analysis of the latent structure suggests that the learned representations are closely aligned with the underlying parameter manifold, indicating that the proposed regularization also improves interpretability of the embedding space.

References

- [1] Neil Ashton, Charles Mockett, Marian Fuchs, Louis Fliessbach, Hendrik Hetmann, Thilo Knacke, Norbert Schonwald, Vangelis Skaperdas, Grigoris Fotiadis, Astrid Walle, et al. Drivaerml: High-fidelity computational fluid dynamics dataset for road-car external aerodynamics. *arXiv preprint arXiv:2408.11969*, 2024.
- [2] Lawrence Sirovich. Turbulence and the dynamics of coherent structures. *Quart. Appl. Math.*, 45:561–571, 1987.
- [3] Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.
- [4] Nadine Aubry, Philip Holmes, John L Lumley, and Emily Stone. The dynamics of coherent structures in the wall region of a turbulent boundary layer. *Journal of fluid Mechanics*, 192:115–173, 1988.
- [5] Saifon Chaturantabut and Danny C Sorensen. Nonlinear model reduction via discrete empirical interpolation. *SIAM Journal on Scientific Computing*, 32(5):2737–2764, 2010.
- [6] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016.
- [7] Yichen Li, Peter Yichen Chen, Tao Du, and Wojciech Matusik. Learning preconditioners for conjugate gradient pde solvers. In *International Conference on Machine Learning*, pages 19425–19439. PMLR, 2023.
- [8] Rahul Trivedi, Logan Su, Jesse Lu, Martin F Schubert, and Jelena Vuckovic. Data-driven acceleration of photonic simulations. *Scientific reports*, 9(1):19728, 2019.
- [9] Azzeddine Tiba, Thibault Dairay, Florian De Vuyst, Iraj Mortazavi, and Juan Pedro Berro Ramirez. Machine-learning enhanced predictors for accelerated convergence of partitioned fluid-structure interaction simulations. *Computer Physics Communications*, 310:109522, 2025.

- [10] Peter Sharpe, Rishikesh Ranade, Kaustubh Tangsali, Mohammad Amin Nabian, Ram Cherukuri, and Sanjay Choudhry. Accelerating transient cfd through machine learning-based flow initialization. *arXiv preprint arXiv:2503.15766*, 2025.
- [11] Mohammad Sadegh Eshaghi, Cosmin Anitescu, Navid Valizadeh, Yizheng Wang, Xiaoying Zhuang, and Timon Rabczuk. Nows: Neural operator warm starts for accelerating iterative solvers. *arXiv preprint arXiv:2511.02481*, 2025.
- [12] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.
- [13] Xinyu Hui, Junqiang Bai, Hui Wang, and Yang Zhang. Fast pressure distribution prediction of airfoils using deep learning. *Aerospace Science and Technology*, 105:105949, 2020.
- [14] Cihat Duru, Hande Alemdar, and Ozgur Ugras Baran. A deep learning approach for the transonic flow field predictions around airfoils. *Computers & Fluids*, 236:105312, 2022.
- [15] Zhenguo Nie, Haoliang Jiang, and Levent Burak Kara. Stress field prediction in cantilevered structures using convolutional neural networks. *Journal of Computing and Information Science in Engineering*, 20(1):011002, 2020.
- [16] Enjamamul Hoq, Osama Aljarrah, Jun Li, Jing Bi, Alfa Heryudono, and Wenzhen Huang. Data-driven methods for stress field predictions in random heterogeneous materials. *Engineering Applications of Artificial Intelligence*, 123:106267, 2023.
- [17] Jiang-Zhou Peng, Xianglei Liu, Zhen-Dong Xia, Nadine Aubry, Zhihua Chen, and Wei-Tao Wu. Data-driven modeling of geometry-adaptive steady heat convection based on convolutional neural networks. *Fluids*, 6(12):436, 2021.
- [18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.
- [19] Mohammad S Khorrami, Jaber R Mianroodi, Nima H Siboni, Pawan Goyal, Bob Svendsen, Peter Benner, and Dierk Raabe. An artificial neural network for surrogate modeling of stress fields in viscoplastic polycrystalline materials. *npj Computational Materials*, 9(1):37, 2023.
- [20] Nils Thuerey, Konstantin Weißenow, Lukas Prantl, and Xiangyu Hu. Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA journal*, 58(1):25–36, 2020.
- [21] Sam Jacob Jacob, Markus Mrosek, Carsten Othmer, and Harald Köstler. Deep learning for real-time aerodynamic evaluations of arbitrary vehicle shapes. *arXiv preprint arXiv:2108.05798*, 2021.

- [22] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.
- [23] Haoliang Jiang, Zhenguo Nie, Roselyn Yeo, Amir Barati Farimani, and Levent Burak Kara. Stressgan: A generative deep learning model for two-dimensional stress distribution prediction. *Journal of Applied Mechanics*, 88(5):051005, 2021.
- [24] Jiajun Hu, Zhen Lu, and Yue Yang. Generative prediction of flow field based on the diffusion model. *arXiv e-prints*, pages arXiv–2407, 2024.
- [25] Donglin Chen, Xiang Gao, Chuanfu Xu, Shizhao Chen, Jianbin Fang, Zhenghua Wang, and Zheng Wang. Flowgan: A conditional generative adversarial network for flow prediction in various conditions. In *2020 IEEE 32nd international conference on tools with artificial intelligence (ICTAI)*, pages 315–322. IEEE, 2020.
- [26] Claudia Drygala, Francesca di Mare, and Hanno Gottschalk. Generalization capabilities of conditional gan for turbulent flow under changes of geometry. *arXiv preprint arXiv:2302.09945*, 2023.
- [27] A Padmaprabhan, Shriram Hari, Nived Philip Thomas, Khaish Singh Chadha, Sai Sidhardh, Viswanath Chinthapenta, and Prabhat Kumar. Go-gan: Geometry optimization generative adversarial network for achieving optimized structures with targeted physical properties. *arXiv preprint arXiv:2502.00416*, 2025.
- [28] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.
- [29] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael 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*, pages 5115–5124, 2017.
- [30] Farnoosh Hadizadeh, Wrik Mallik, and Rajeev K Jaiman. A graph neural network surrogate model for multi-objective fluid-acoustic shape optimization. *arXiv preprint arXiv:2412.16817*, 2024.
- [31] Jiaqian Wu, Chaoyu Du, Benjamin Dillenburg, and Michael Anton Kraus. Fast prediction of stress distribution: A gnn-based surrogate model for unstructured mesh fea. In *Proceedings of the IASS 2024 Symposium: Redefining the Art of Structural Design*, volume 2024. International Association for Shell and Spatial Structures, 2024.
- [32] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International conference on learning representations*, 2020.

- [33] Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, et al. Geometry-informed neural operator for large-scale 3d pdes. *Advances in Neural Information Processing Systems*, 36:35836–35854, 2023.
- [34] Xingyu Fu, Fengfeng Zhou, Dheeraj Peddireddy, Zhengyang Kang, Martin Byung-Guk Jun, and Vaneet Aggarwal. An fea surrogate model with boundary oriented graph embedding approach. *arXiv preprint arXiv:2108.13509*, 2021.
- [35] Siyi Li, Mingrui Zhang, and Matthew D Piggott. End-to-end wind turbine wake modelling with deep graph representation learning. *Applied Energy*, 339:120928, 2023.
- [36] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.
- [37] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5939–5948, 2019.
- [38] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 165–174, 2019.
- [39] Xiang Huang, Zhanhong Ye, Hongsheng Liu, Shi Ji, Zidong Wang, Kang Yang, Yang Li, Min Wang, Haotian Chu, Fan Yu, et al. Meta-auto-decoder for solving parametric partial differential equations. *Advances in Neural Information Processing Systems*, 35:23426–23438, 2022.
- [40] Jonas Zehnder, Yue Li, Stelian Coros, and Bernhard Thomaszewski. Ntopo: Mesh-free topology optimization using implicit neural representations. *Advances in Neural Information Processing Systems*, 34:10368–10381, 2021.
- [41] Giovanni Catalani, Siddhant Agarwal, Xavier Bertrand, Frederic Tost, Michael Bauerheim, and Joseph Morlier. Neural fields for rapid aircraft aerodynamics simulations. *Scientific Reports*, 14(1):25496, 2024.
- [42] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.
- [43] Elijah Ang and Bing Feng Ng. Physics-informed neural networks for flow around airfoil. In *AIAA Scitech 2022 Forum*, page 0187, 2022.
- [44] Hamidreza Eivazi, Mojtaba Tahani, Philipp Schlatter, and Ricardo Vinuesa. Physics-informed neural networks for solving reynolds-averaged navier–stokes equations. *Physics of Fluids*, 34(7), 2022.

- [45] Zhiping Mao, Ameya D Jagtap, and George Em Karniadakis. Physics-informed neural networks for high-speed flows. *Computer Methods in Applied Mechanics and Engineering*, 360:112789, 2020.
- [46] Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, 2021.
- [47] Ameya D Jagtap, Zhiping Mao, Nikolaus Adams, and George Em Karniadakis. Physics-informed neural networks for inverse problems in supersonic flows. *Journal of Computational Physics*, 466:111402, 2022.
- [48] Marco Baldan, Paolo Di Barba, and David A Lowther. Physics-informed neural networks for inverse electromagnetic problems. *IEEE Transactions on Magnetics*, 59(5):1–5, 2023.
- [49] Wei Chen, Boqiang Liu, Suting Peng, Jiawei Sun, and Xu Qiao. S3d-unet: Separable 3d u-net for brain tumor segmentation. In Alessandro Crimi, Spyridon Bakas, Hugo Kuijf, Farahani Keyvan, Mauricio Reyes, and Theo van Walsum, editors, *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pages 358–368, Cham, 2019. Springer International Publishing.
- [50] Hrvoje Jasak, Aleksandar Jemcov, and Zeljko Tukovic. Openfoam: A c++ library for complex physics simulations. In *International Workshop on Coupled Methods in Numerical Dynamics*, 2007.
- [51] Facebook Research. Deepsdf: Learning continuous signed distance functions for shape representation. <https://github.com/facebookresearch/DeepSDF>, 2019. Official reference implementation.
- [52] Laura Balasso and Alessandro Alaia. A non-parametric regularization framework for surrogate learning based on deep encoding. *Physics of Fluids*, 38(1), 2026.
- [53] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International conference on machine learning*, pages 5301–5310. PMLR, 2019.
- [54] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in neural information processing systems*, 33:7537–7547, 2020.
- [55] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. 1998.

- [56] Kevin McLaren. *A numerical and experimental study of unsteady loading of high solidity vertical axis wind turbines*. PhD thesis, McMaster University, 2011. Available at <http://hdl.handle.net/11375/11096>.
- [57] ANSYS, Inc. Ansys®free student fluent, release 25.1.
- [58] Fausto Dicech and Lucia Parussini. A multi-fidelity reduced-order model to quantify aerodynamic forces on a vertical-axis wind turbine with uncertain rotational speed. In *UNCECOMP 2025 Proceedings*, 2025. url: <https://2025.uncecomp.org/proceedings/pdf/21079.pdf>.
- [59] John H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2:84–90, 1960.
- [60] SU2 Foundation. Su2 version 8.3.0 "harrier", August 2025.
- [61] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems* 32, pages 8024–8035. Curran Associates, Inc., 2019.
- [62] Fausto Dicech, Konstantinos Gkaragkounis, Lucia Parussini, Anna Spagnolo, and Haysam Telib. Integration of multi-fidelity methods in parametrized non-intrusive reduced order models for industrial applications. *Journal of Computational Science*, 85:102511, 2025.
- [63] Qiao Zhang, Xu Wang, Dangguo Yang, and Weiwei Zhang. Data-driven prediction of aerodynamic noise of transonic buffeting over an airfoil. *Engineering Analysis with Boundary Elements*, 163:549–561, 2024.
- [64] Qiao Zhang, Kuijun Zuo, Weiwei Zhang, and Zihao Dou. Prediction of transonic buffet aerodynamic noise using multi-expert feature fusion network. *AIAA Journal*, pages 1–5, 2025.

Appendix A

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 pairwise distances. In Figure A.1 are shown the elapsed time on 16-cores CPU and the memory usage for the computation of pairwise 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 consumption remain affordable with 10,000 samples, 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. In Figure A.2 are plotted the memory usage and the time consumption due to this computation for a single training epoch.

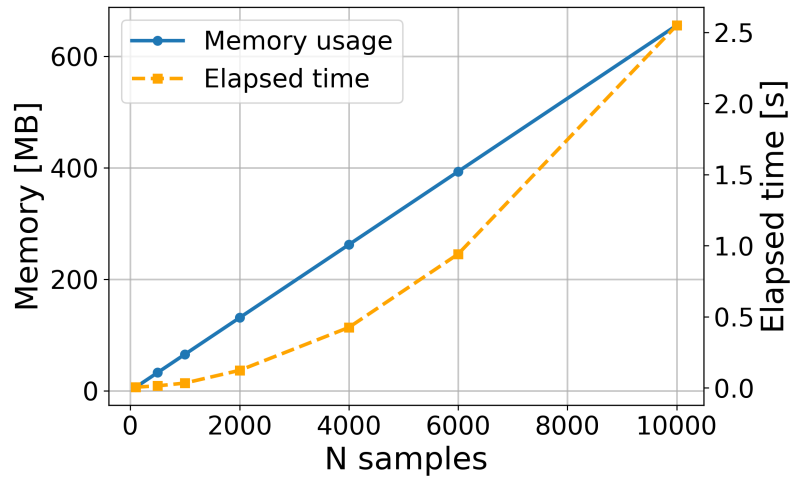


Fig. A.1 Distances computation time and memory usage for an increasing number of samples, each one of size 16,384.

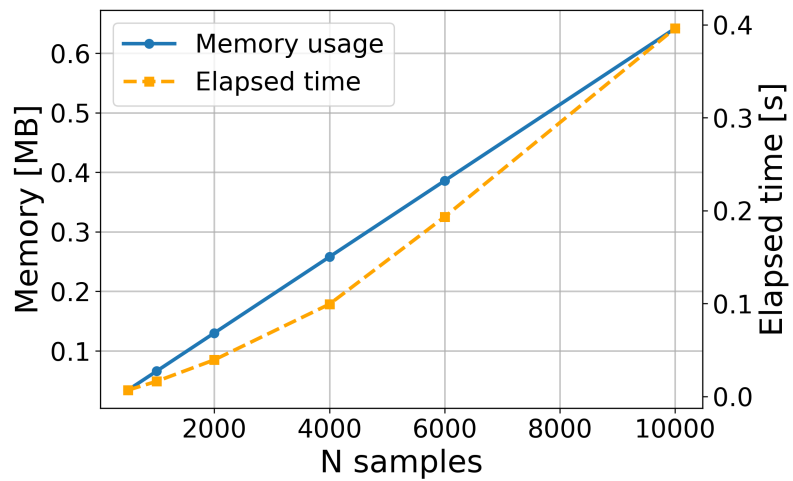


Fig. A.2 Latent distances computation time and memory usage for an increasing number of samples.

Appendix B

Tuning of Regularization Weight

The regularization weight λ in Equation 4.10 was tuned for each experiment presented in Section 5.2. 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 Figure B.1. 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.

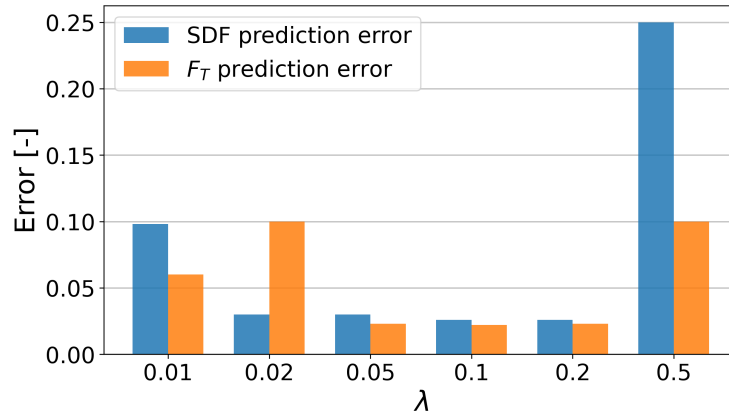


Fig. B.1 SDF prediction error and F_T prediction error for six values of regularization weight λ in VAWT regularized geometric encoder.

In the RAE experiment, the regularization weight was tuned for both the geometric and the output encoders. Figure B.2 shows that the best geometric encoding results were obtained with $\lambda = 0.1$. In this case, the corresponding output prediction error is 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 Figure X) 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, Figure X also shows significantly reducing λ leads to deteriorated results, confirming that the regularization term plays an important role in improving the model's accuracy.

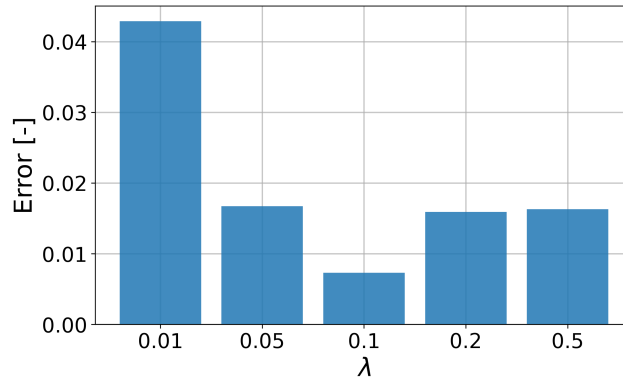


Fig. B.2 SDF prediction error for different values of regularization weight λ .

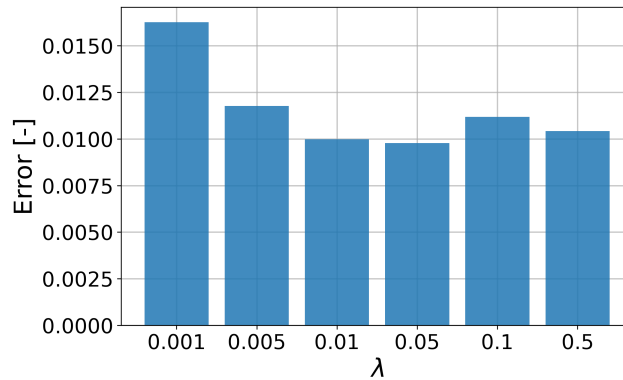


Fig. B.3 C_p reconstruction error for different values of regularization weight λ .

Appendix C

VAWT Radial Force Prediction

In this appendix, the results reported in Section 5.2.2 for F_T signal are reproduced for the radial force F_R . The bar plots in Figure C.1 show the relative prediction errors on training and validation datasets obtained with the non-regularized SWF. In Figure C.2 are displayed 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. In Table C.1 are reported the error statistics obtained with POD+GP models (single and multi-fidelity) and the errors obtained with our methods. Figure C.3 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.

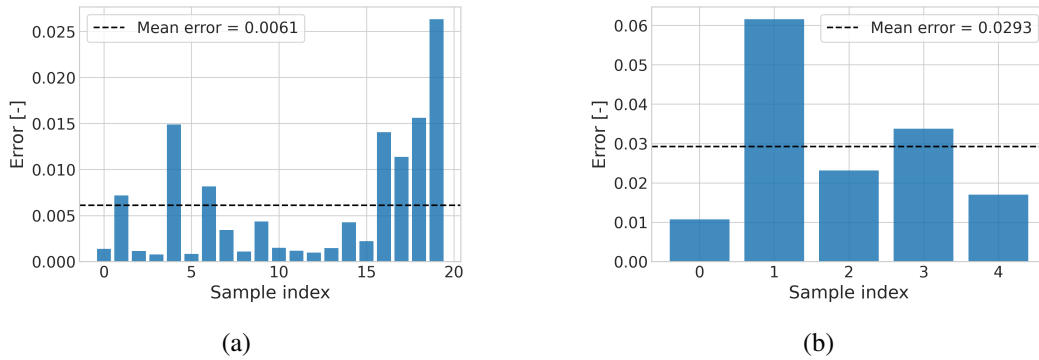


Fig. C.1 F_R relative prediction error on (a) training and (b) validation samples.

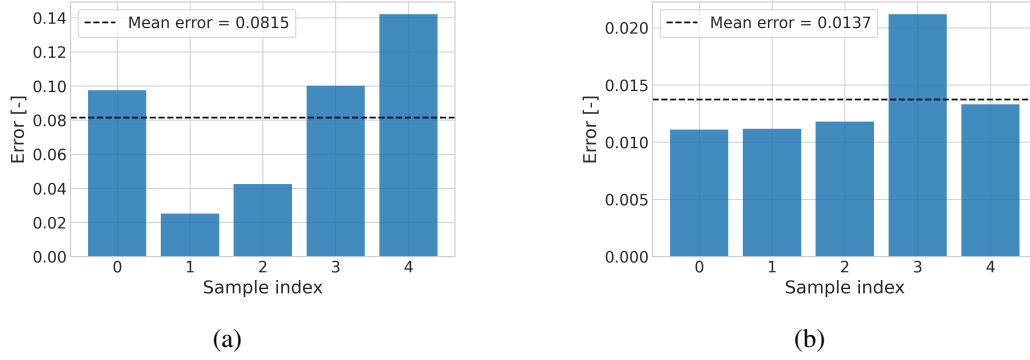


Fig. C.2 Relative error on the prediction of the thrust force, F_R on the validation dataset obtained with the (a) FRD regularization, and (b) with PWD regularization.

Method	Dataset	Mean Error	Std
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

Table C.1 F_R prediction errors. Boldface denotes the best result both in terms of mean validation error and standard deviation.

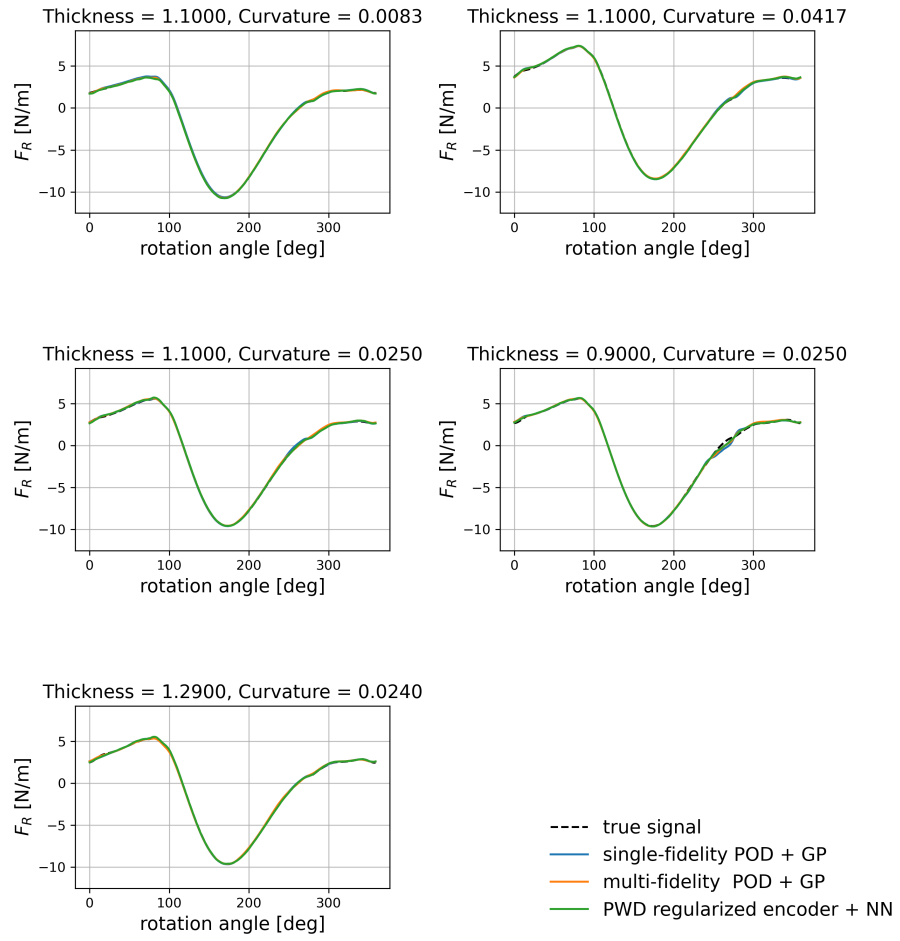


Fig. C.3 Radial force predictions obtained with regularized DeepSDF + NN, compared with single-fidelity and multi-fidelity POD + GP.

Appendix D

RAE2822 Pressure Coefficient Predictions

In this appendix, two examples of C_p prediction are presented. In Figure [D.1](#), the same validation case is predicted using the baseline workflow and the workflow with FRD and PWD regularization. The same visualization is shown in Figure [D.2](#) for another validation case. In the first case, both regularization methods reduced the 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.

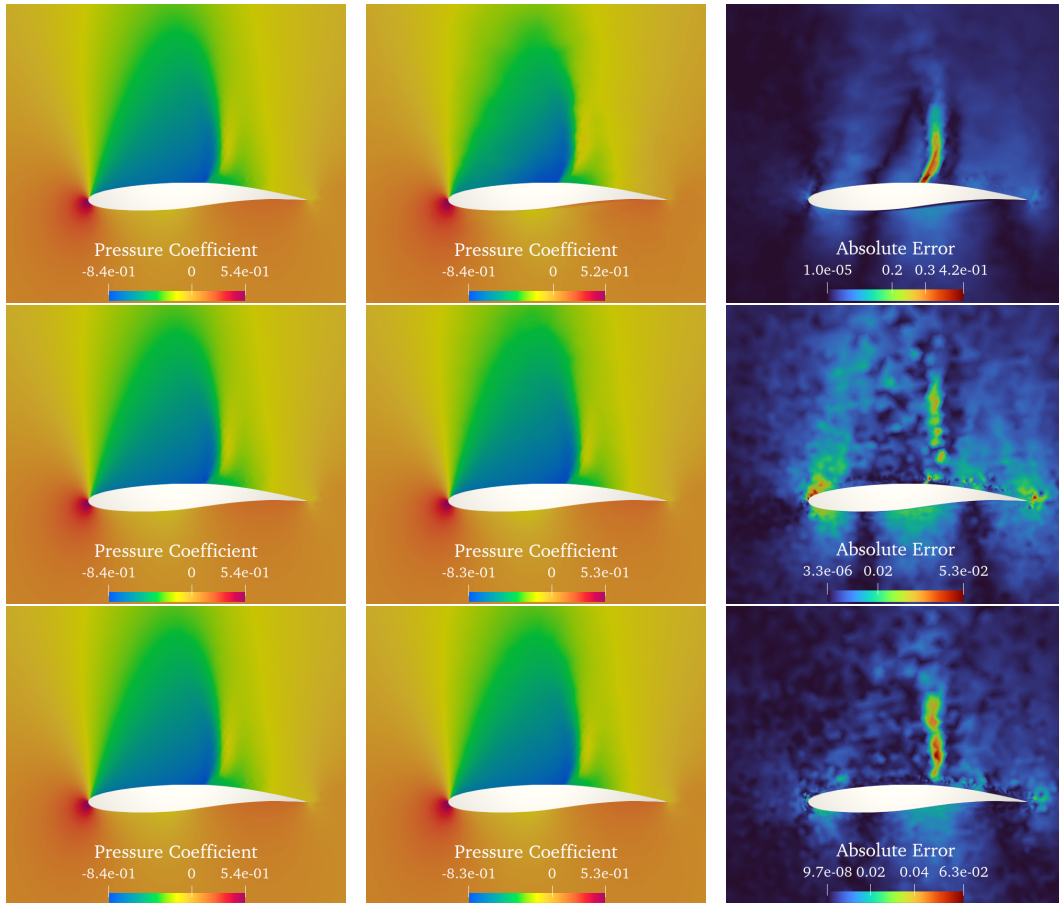


Fig. D.1 Pressure coefficient prediction for an unseen geometry obtained with the complete workflow without regularization (upper row), with FRD regularization (middle row) and with PWD regularization (bottom row). For each method, true C_p field (left column), predicted C_p (central column) and absolute error field (right column).

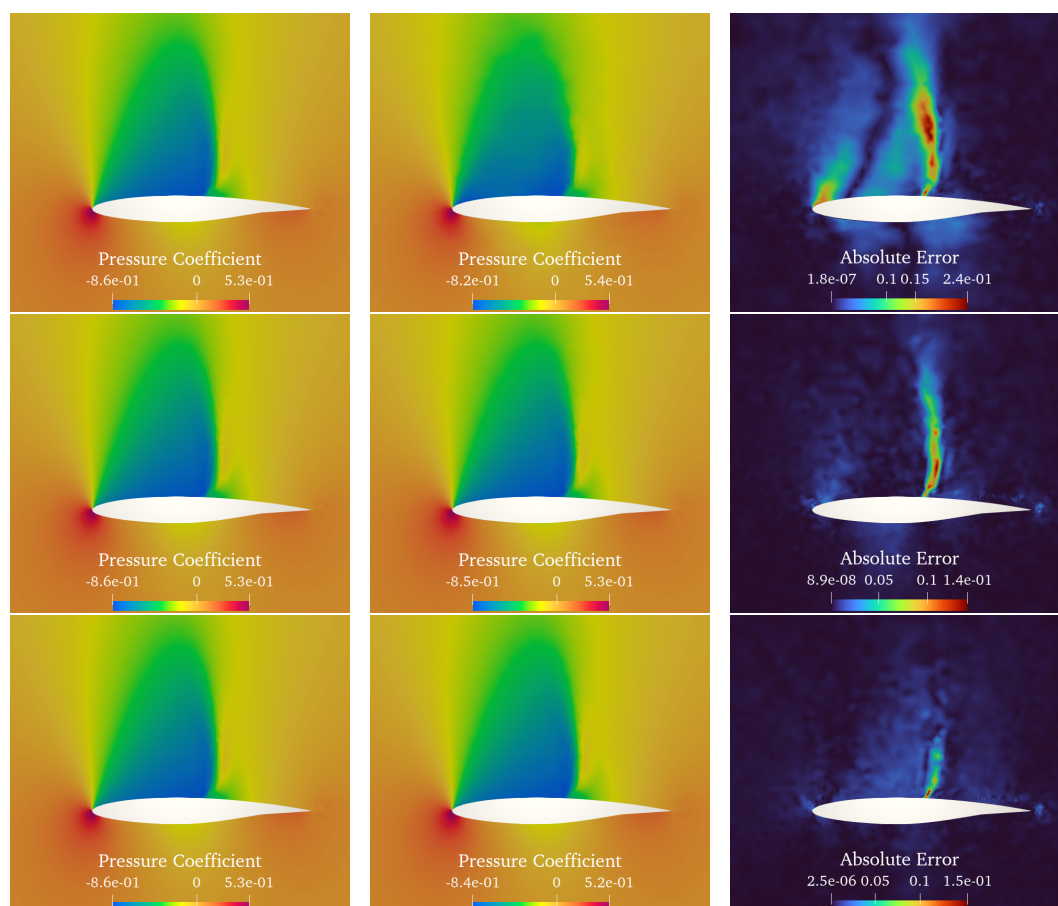


Fig. D.2 Pressure coefficient prediction for an unseen geometry obtained with the complete workflow without regularization (upper row), with FRD regularization (middle row) and with PWD regularization (bottom row). For each method, true C_p field (left column), predicted C_p (central column) and absolute error field (right column).

Appendix E

Training and Inference Time

All experiments were implemented in PyTorch [61] v2.1. The training of the encoding models was performed on a single NVIDIA A10 GPU. Mapping models were trained on a single CPU.

VAWT

The three geometric encoders, i.e. DeepSDF, FRD regularization and PWD regularization, have 4 hidden layers of size 128 and were trained for 2000 epochs on a training dataset composed by 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 E.1. 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 4 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 minutes on a single CPU.

RAE2822

The geometric encoders architecture is composed by 4 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 regularized encoders is reported in Table E.1. Latent code optimization required on average 1.5 s for 300 epochs per validation geometry. The output encoders, consisting of 4 layers of size 256, were trained for 2000 epochs on 1000 C_p fields (25,900 points per field, batch size 16). The corresponding training times are also reported in Table E.1. Latent code optimization in this case required on average 2.3 s for 400 epochs per validation field. The mapping block, composed of 4 layers of size 64, was trained for 1000 epochs on 1000 samples (800 training, 200 test) with batch size 32, with a total training time of approximately 6 min on a single CPU.

	VAWT	RAE2822	
	geom. encoder	geom. encoder	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

Table E.1 GPU training time of encoding models.

Prediction of Unseen Samples

Table E.2 reports the elapsed time for each inference step. 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 latent code-to-field reconstruction is significantly faster in the VAWT case, where the output is predicted directly from the geometric latent code using a simplified workflow. The total prediction time is approximately 1.4 seconds for both methods, indicating that the proposed approach is suitable for near-real-time applications such as shape optimization and design exploration.

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

Table E.2 Elapsed time for each step of the prediction of an unseen sample.