

Prediction of surface pressure distributions of non-parametric airfoils using geometric deep learning methods

Derrick Hines^{1,*}, Philipp Bekemeyer²

DLR German Aerospace Center, Lilienthalpl. 7, Brunswick, 38108, Germany

ARTICLE INFO

Keywords:

Surrogate model
Geometric deep learning
Graph neural network
Implicit neural representation
Multi-layer perceptron
Aerodynamics

ABSTRACT

Computational Fluid Dynamics (CFD) simulations are one of the cornerstones in providing aerodynamic data required for aircraft design and optimization. However, using these simulations extensively is limited by their high computational demands. Therefore, it is essential to create efficient data-driven surrogate models for CFD solvers. In many practical scenarios an explicit unifying parameterization of aircraft configurations is not available. This highlights the need for models that operate directly on raw geometric representations. Geometric deep learning has emerged as a class of deep learning techniques capable of operating on such data, enabling predictive modeling without the need of an explicit parameterization. In this paper, we extend and investigate two geometric deep learning approaches for the prediction of surface pressure distributions of non-parametric airfoils. These methods are Bi-Stride Multi-Scale Graph Neural Network and Implicit Neural Representation of the signed distance function coupled with a Multi-Layer Perceptron. To enhance both of these methods, we propose the use of area-weighted loss functions to better account for variations in node density in the meshes. Moreover, in the formulation of the graph neural network we introduce edge completion at the coarsest level to account for interactions between different connected components, such as flap, main element and slat in a 3-element high-lift airfoil. These methods are compared to the well-established method Proper Orthogonal Decomposition coupled with Interpolation, which is allowed to use an explicit parameterization and serves as a baseline. Two high-fidelity datasets with CFD simulations solving the Reynold-Averaged Navier Stokes equations are created. The first one features a varied set of single-element airfoils with simulations in the subsonic and transonic regime, while the second one features high-lift multi-element airfoils with a variable flap position with simulations in the subsonic regime. The results show that both geometric deep learning approaches outperform the established baseline across various data regimes. These approaches can capture shocks and flow separation with more accuracy. The use of an area-weighted loss function enhances area-weighted performance and leads to faster performance gains in the early training epochs. These findings support the potential of geometric deep learning methods as data-driven surrogates of CFD solvers for varying geometries.

1. Introduction

Aircraft design and optimization relies heavily on Computational Fluid Dynamics (CFD) to evaluate aerodynamic performance, structural loads and handling qualities. However, high-fidelity CFD simulations solving the Reynolds-Averaged Navier-Stokes (RANS) equations require significant time and resources. This limits their use in tasks that require numerous and fast evaluations, such as early-stage large-scale design exploration, loads predictions, rapid assessment of design changes, shape optimization and uncertainty quantification. To mitigate this limitation, surrogate models emerged as alternatives that provide rapid aerodynamic predictions.

Since the end of the last century, various data-driven surrogate models have been proposed, with interest growing significantly over the past decade due to the rapid and widespread adoption of deep learning techniques. Such models are nearly always built using a database of CFD simulations from which they learn patterns. After being trained, the models provide predictions for new flow conditions or configurations with a much lower computational cost than the CFD solver. Unlike traditional solvers, data-driven models cannot provide one-shot predictions for unseen out-of-distribution CFD cases without prior exposure during training. Although the generation of the CFD database involves a significant upfront investment, this cost can be offset by the benefits it provides in the downstream phases of the design process when time

* Corresponding author.

E-mail addresses: Derrick.HinesChaves@dlr.de (D. Hines), Philipp.Bekemeyer@dlr.de (P. Bekemeyer).

¹ Research Scientist, Institute of Aerodynamics and Flow Technology, Center for Computer Applications in Aerospace Science and Engineering

² Team Leader, Institute of Aerodynamics and Flow Technology, Center for Computer Applications in Aerospace Science and Engineering

constraints need to be met. This upfront cost can also be partially mitigated in industry by using CFD results already being produced as part of the standard design, certification and validation processes. However, the volume and diversity of CFD data available in industry remains limited, posing a significant challenge for the data-hungry nature of deep learning models.

Proper Orthogonal Decomposition (POD) coupled with Interpolation (PODI) is one of the pioneering approaches in data-driven modeling and is now a well-established method with widespread use in aerodynamics [1–5]. However, it is a global linear method and is prone to failure in cases that feature local nonlinear phenomena, such as shocks and flow separation. Gaussian Process Regression (GPR) was also one of the proposed surrogate modeling techniques, however its main limitation is poor scalability, which restricts its applicability to high dimensional field prediction problems [6,7]. In the last decades, the challenges faced by these methods encouraged the adoption of alternative methods from the field of deep learning (DL). Convolutional neural networks (CNN) were among the earliest deep learning models as surrogates for field predictions, following earlier use of classical machine learning methods [8–12]. Shallow neural networks had been used previously but mainly for the prediction of scalar aerodynamic coefficients [13,14].

While convolutional neural networks had been successful in handling data on regular grids, their limitations in processing irregular geometries motivated two main research directions: the development of graph neural networks (GNNs), which directly operate on non-Euclidean data such as meshes and graphs, and coordinate-based multi-layer perceptrons (MLPs) that map directly any given coordinates to the target values at those coordinates. In this second direction earlier work focused on using as input the pointwise spatial coordinates and a known geometric parameterization [7], while recent approaches have extended this concept by learning latent codes representing the signed distance functions (SDFs) around the geometries, which are then used as additional inputs to the MLP without relying on a known geometric parameterization [15–18]. In this paper, we consider geometric deep learning (GDL) [19] to broadly include methods in both research directions. Originally, geometric deep learning focused primarily on graphs, however the concept has extended to include other approaches designed to operate on unstructured geometric data.

Graph Neural Networks have progressively advanced as a powerful tool for predicting steady and unsteady-state physical fields on graph-structured domains. Initial applications used message passing networks focused on local interactions and showed good results on small to medium scale problems simulating the dynamics of physical systems with particles expressed as nodes in a graph [20]. The framework of MeshGraphNets was introduced in [21], which utilized iterative message passing over mesh nodes to learn physical dynamics. Although originally targeted at unsteady simulations, MeshGraphNets demonstrated applicability to steady-state field prediction [22,23]. Nevertheless, their reliance on local and uniform message passing introduced scalability issues as mesh resolution increased and information needed to be propagated at different scales, with computational and memory costs growing rapidly with graph size. In previous work [23] we employed a MeshGraphNet for meshes with around 500,000 nodes, however geometric parameterizations were provided explicitly to the model so that propagating information at large scales was not required, enabling the use of message passing on small subgraphs.

In order to address these scalability issues various algorithms were studied to introduce multi-scale structures to GNNs [24–28]. The Bi-Stride Multi-Scale Graph Neural Network (BSMS-GNN) [24] emerged as one of the most promising solutions over other approaches requiring labor intensive creation of coarser meshes. In this approach message passing alternates between fine and coarsened versions of a mesh to efficiently capture local and global interactions with an architecture resembling a U-Net, using downsampling and upsampling paths to aggregate

multiscale context while preserving high-resolution detail. While this approach is able to propagate information throughout the graph more efficiently, its implementation requires processing the whole graph at once, limiting its scalability. Recent developments as XMeshGraphNets [29] have focused on making MeshGraphNets scalable by partitioning large graphs and incorporating halo regions that enable message passing across partitions. This approach is reported to be capable of processing point clouds of up to 2 million surface nodes in a single 80GB GPU for the DrivAerML [30] car dataset. In [31] a multi-scale model was proposed that uses random downsampled regional meshes, which can potentially increase scalability. In [32] a framework based on the standard graph convolutional network (GCN) was extended to deal with geometric variations and lead to significant computing cost saving with respect to MeshGraphNets. Parallel to these advances in GNNs, transformers and graph transformers have been widely studied for predicting physical fields [33]. Solutions to mitigate their scalability issues have also been explored, such as Anchored-Branched Universal Physics Transformers (AB-UPT) [33]. This method has been employed on meshes of the DrivAerML car dataset with around 140 million volume nodes using a single 80GB GPU. Other point-cloud based methods, some of which also feature the sampling of anchor nodes, have been used for unmanned aerial vehicles and launch vehicles with multiple shape variations [34–36].

Regarding the second group of geometric deep learning methods considered here, coordinate-based multi-layer perceptrons were first adopted in aerodynamic surrogate modeling by mapping spatial coordinates alongside explicit parameter vectors to local field values such as pressure coefficients [7]. This can be considered as an implicit neural representation (INR) or neural field, a neural network that encodes a continuous signal or field as a function mapping spatial coordinates to output values. Its key feature is that the signal is stored implicitly in the network's weights rather than in explicit discrete structures, such as meshes. It is considered a conditional INR because in addition to the coordinates it takes as input the geometric parameterization. In [7] this method was employed to predict surface pressure coefficients on a case featuring the NASA Common Research Model (NASA CRM) with parameterized surface deflections and around 500,000 nodes. This approach required geometries to be explicitly parameterized, limiting their generality.

A natural extension to the previous approach without assuming the availability of a geometric parameterization consists in using in a first step an implicit neural representation of the signed distance functions (SDFs) [15] to create geometry encoding codes that in a second step are used as input to the multi-layer perceptron instead of a known parameterization. We refer to this approach as INR-MLP. The approach introduced in [15] for learning the codes was then extended using techniques such as latent modulation and meta-learning [16]. Conditioning the predictor MLP via feature-wise modulation has also been used as an alternative to input concatenation. These coordinate-based methods have been further extended into more complex operator learning encoder-processor-decoder architectures such as Coral [16] and INFINITY [17], in which the networks are trained to approximate PDE solution operators by learning mappings from geometry latent codes to target latent codes that condition an implicit neural representation of the target field. In [18] a version of these approaches was employed for predicting the surface pressure distribution on 3D XRF1 wings with 6,600 nodes, featuring shape, inflow condition, and control surface deflection variations.

As described and exemplified previously, geometric deep learning methods work on raw geometric objects without requiring a predefined parameterization. Hence, they are flexible and applicable in settings involving non-parametric geometries as well as legacy data where explicit parameterizations are missing or different. Furthermore, many 3D aircraft configurations lack an explicit parameterization, so that geometric deep learning models stand out as promising powerful approaches for larger-scale industrial applications. Even if a parameterization exists,

some of these methods can still provide valuable advantages by capturing complex spatial relationships and improving prediction accuracy.

Most large-scale datasets for aerodynamic modeling focus on automotive applications rather than aircraft. Some of the available datasets for aircraft are based on the NASA Common Research Model (NASA CRM). The German Aerospace Center (DLR) created a few years ago a dataset of 149 RANS simulations with this model featuring 6 parameterized surface deflections as well as variable Mach number and angle of attack. This dataset is now publicly available [37]. The French Aerospace Lab (ONERA) created a dataset that contains 468 RANS simulations using a fixed wing-body-pylon-nacelle configuration and variable Mach number, angle of attack and Reynolds number [38]. Only recently, SHIFT-Wing [39] was developed as a collaboration between Luminary Cloud, Otto Aviation and NVIDIA. SHIFT-Wing is based on parametric variants of the NASA-CRM model and contains more than 3000 simulations at variable Mach numbers and angles of attack.

Due to the scarcity of datasets for 3D aerodynamic configurations suitable for deep learning applications most research has focused on 2D airfoils. AIRFRANS was introduced in [40] as a dataset with 1000 simulations for studying the two-dimensional incompressible steady-state RANS equations over airfoils at a subsonic regime and for different angles of attacks. In recent work, datasets with more airfoils and simulations also covering the transonic regime have been created [41,42]. The dataset UniFoil in [42] consists of 38,400 airfoils, with a total of 500,000 simulations with Mach numbers ranging from 0.1 to 0.85, and also features variable angles of attack and Reynolds number. Less research has focused on multi-element airfoils used primarily in high-lift systems. Early work from the end of the last century focused on the prediction of scalar integral quantities [13]. More recently, in [43] a dataset was created for two-dimensional airfoils with a main element and a flap. In total it consists of 43,200 simulations covering 200 airfoil shapes, with variable flap deflection angle, Mach number and angle of attack.

In this work we extend and investigate two promising geometric deep learning methods for the prediction of surface pressure distributions on non-parametric single and multi-element airfoils, given various levels of data availability. These methods are the BiStride Graph Neural Network (BSMS-GNN) and the Implicit Neural Representation of the signed distance function coupled with a Multi-Layer Perceptron (INR-MLP). To enhance both of these methods, we propose the use of area-weighted loss functions to better account for variations in node density in the meshes. Moreover, in the formulation of BSMS-GNN we introduce edge completion at the coarsest level to account for interactions between different connected components, such as flap, main element and slat in a 3-element high-lift airfoil. We provide a systematic comparison of these parameterization-agnostic methods that serves as a substantial step towards the adoption of geometric deep learning methods within the aerospace industry. We construct two tailored high-fidelity datasets solving the Reynolds-Averaged Navier Stokes (RANS) equations, aiming to strike a balance between the inherently data-hungry nature of the deep learning methods and the limited data availability encountered in real industrial applications. The first dataset features a varied set of single-element airfoils with simulations in the subsonic and transonic regime. The second dataset features high-lift multi-element airfoils with a variable flap position with simulations in the subsonic regime. The high-lift airfoils are based on the DLR-F15 model [44], and the positions of the slat and main element remain fixed and match the landing configuration. The geometric deep learning models are compared to Proper Orthogonal Decomposition coupled with Interpolation (PODI) for comparison with a well established approach and to provide a baseline for evaluating performance. This paper is organized as follows. Section 2 describes the problem statement, the baseline approach and the two geometric deep learning methods. Section 3 introduces the two test cases, and Section 4 presents the results for both test cases.

2. Methodology

2.1. Problem statement

In this work, our goal is to create surrogate models that predict surface pressure coefficient distributions for non-parametric airfoils under given flow conditions. All airfoils are assumed to belong to a set of airfoils $\mathcal{A}$. Each airfoil $S \in \mathcal{A}$ is represented by its surface mesh as $S = (X, A)$, an ordered pair of node coordinates and their adjacency (connectivity) matrix. The node coordinates are given by $X \in \mathbb{R}^{N_S \times 2}$, where N_S corresponds to the number of nodes. The i th row contains the node coordinates $\mathbf{x}_i^\top = (x_i, z_i)$ of node i . For the adjacency matrix $A \in \{0, 1\}^{N_S \times N_S}$ it holds that $A_{ij} = 1$ whenever there is a connection from node i to node j and $A_{ij} = 0$ if there is none.

From the surface mesh S , we obtain the unit normal vectors $\mathbf{v}_i^\top = (v_{x,i}, v_{z,i})$ and the surface areas Δs_i (lengths) that correspond to each node. For simplicity, *area* and *length* are used interchangeably with *length* applying in 2D and *area* extending naturally to 3D. To each node we assign the relative area weight w_i corresponding to the proportion of its area with respect to the total area of the airfoil. We consider the relative area-weight vector $\mathbf{w}$ as given by

$$\mathbf{w} = (w_1, \dots, w_{N_S})^\top \quad (1)$$

$$w_i = \frac{\Delta s_i}{\sum_{j=1}^{N_S} \Delta s_j} \quad (2)$$

The flow conditions are denoted by $\mathbf{p} \in \mathcal{P}$ and are typically the Mach number, angle of attack, and Reynolds number. The surface pressure coefficient distribution $\mathbf{y} \in \mathbb{R}^{N_S}$ obtained with the CFD solver for the flow parameters $\mathbf{p}$ and the airfoil S is assumed to be given by a function f via

$$\mathbf{y} = f(\mathbf{p}, S) \quad (3)$$

$$f : \mathcal{P} \times \mathcal{A} \rightarrow \bigcup_{S \in \mathcal{A}} \mathbb{R}^{N_S} \quad (4)$$

The goal of the surrogate models is to approximate the function f with a faster-to-evaluate function $\hat{f}_\theta$ with learnable parameters θ as

$$\hat{\mathbf{y}} = \hat{f}_\theta(\mathbf{p}, S) \approx f(\mathbf{p}, S) = \mathbf{y} \quad (5)$$

We are interested in obtaining predictions that match the pressure coefficient distributions well throughout each airfoil. To this end, we consider an area-weighted loss function $\mathcal{L}$ as the area-weighted average of pointwise loss terms given by a pointwise loss function $\ell : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$. The metric for the accuracy of the approximation $\hat{\mathbf{y}}$ of $\mathbf{y}$ throughout an airfoil is expressed by

$$\mathcal{L}((\mathbf{y}, \hat{\mathbf{y}}, \mathbf{w})) = \sum_i w_i \ell(\mathbf{y}_i, \hat{\mathbf{y}}_i) \quad (6)$$

$$\mathcal{L} : \bigcup_{S \in \mathcal{A}} (\mathbb{R}^{N_S} \times \mathbb{R}^{N_S} \times \mathbb{R}^{N_S}) \rightarrow \mathbb{R} \quad (7)$$

The models seek to minimize the expected value of the area-weighted loss function $\mathcal{L}$. This has two advantages, (1) regions with higher node density do not gain unintended significance with respect to regions with lower node density, (2) airfoils with more nodes or more area do not attain greater relevance than the others.

If we consider $\mu : \mathcal{P} \times \mathcal{A} \rightarrow \mathbb{R}_+$ the underlying probability measure of the product of the sets of parameters and airfoils, then the expected value of the loss function $\mathcal{L}$ with the surrogate $\hat{f}_\theta$ with parameters θ is given by

$$\mathbb{E}_{\mu, \theta}[\mathcal{L}] = \int_{\mathcal{P} \times \mathcal{A}} \sum_{i=1}^{N_S} w_i^S \ell(\mathbf{y}(\mathbf{p}, S)_i, \hat{f}_\theta(\mathbf{p}, S)_i) d\mu(\mathbf{p}, S) \quad (8)$$

Three datasets, $\mathcal{D}_{\text{train}}$, $\mathcal{D}_{\text{val}}$, $\mathcal{D}_{\text{test}}$, are created to train, validate and test the models. Each dataset $\mathcal{D}$ contains $M_{\mathcal{D}}$ snapshots and is of the form $\mathcal{D} = \{(\mathbf{p}_j, S_j, \mathbf{y}_j)\}_{j \in \mathcal{J}_{\mathcal{D}}}$, where $\mathcal{J}_{\mathcal{D}}$ is a set of $M_{\mathcal{D}}$ indices. Using the dataset $\mathcal{D}$, the probability measure μ is approximated with a uniform discrete

probability measure P_D over the dataset D . The expected value of the loss function is then approximated as

$$\mathbb{E}_{\mu, \theta}[\mathcal{L}] \approx \mathbb{E}_{P_D, \theta}[\mathcal{L}] = \sum_{j \in J_D} \frac{1}{N_{S_j}} \sum_{i=1}^{N_{S_j}} w_i^{S_j} \ell(\mathbf{y}(\mathbf{p}_j, S_j)_i, \hat{f}_\theta(\mathbf{p}_j, S_j)_i) \frac{1}{M_D} \quad (9)$$

During training the models seek to minimize the expected value approximation using the train dataset D_{train} and a chosen pointwise loss function ℓ_{train} by learning the parameters θ . Among the models that are trained, the best model is selected by using a test pointwise loss function ℓ_{test} and the validation dataset D_{val} . To evaluate the generalization capability of the models the pointwise loss function ℓ_{test} and the test dataset D_{test} are used. In Section 4 we explain in more detail the specific metrics that we use to evaluate the models.

2.2. Baseline approach and parametric airfoils

In this paper we compare the non-parametric geometric deep learning methods to a widely established parametric method known as Proper Orthogonal Decomposition (POD) coupled with Interpolation (PODI), which is taken to be the baseline method. We consider parametrically generated airfoils, but with possibly different parameterizations. Hence, the set of airfoils $\mathcal{A} = \bigcup_i \mathcal{A}_i$ is taken to be the union of families of parameterized airfoils. Each parametric family $\mathcal{A}_i = \{S_i(\xi) \mid \xi \in \Xi_i\}$ is parameterized by shape and position parameters $\xi = [\sigma, \eta] \in \Xi_i$. Considering $\Xi = \bigcup_i \Xi_i$, we allow the baseline parametric surrogate $\hat{f}_B$ to incorporate parameterization information and compute predictions as

$$\hat{\mathbf{y}} = \hat{f}_B(\mathbf{p}, S, \xi) \approx f(\mathbf{p}, S) \quad (10)$$

$$\hat{f}_B : \mathcal{P} \times \mathcal{A} \times \Xi \rightarrow \bigcup_{S \in \mathcal{A}} \mathbb{R}^{N_S} \quad (11)$$

The function $\hat{f}_B$ consists of different PODI sub-models for each parameterization. POD extracts orthogonal modes that capture the main patterns in the pressure coefficient distributions. Interpolating the mode coefficients for new flow and geometric parameters allows predictions at new conditions. A more detailed explanation can be found in [23]. For each PODI sub-model an interpolation to a reference mesh might be beneficial or even required to account for different number of nodes or different node distribution in the meshes.

2.3. Geometric deep learning task

The task of the geometric deep learning models is to provide predictions of the surface pressure distributions for multiple airfoils without a geometric parameterization. The models should be able to answer the following question: Given flow parameters $\mathbf{p}$ and a point (x, z) on the airfoil S what is the pressure coefficient at that point? The pressure coefficient at a point (x, z) depends on the overall shape of the airfoil, not only on the local geometry, so that making a prediction at a point requires knowledge of the complete airfoil S . The flow parameters $\mathbf{p}$ and a point (x, z) can be trivially given as input to a neural network. However, the airfoils potentially have a different number of nodes, connectivity, and distance between nodes. Hence, the main advantage of these models is that they can represent and process this geometric input.

2.4. Bi-stride multi-scale graph neural network

The model Bi-Stride Multi-Scale Graph Neural Network (BSMS-GNN) introduced in [24] is described in this section with an emphasize on the task at hand, and a modification to account for interaction between different connected components in a graph, such as the flap, main wing and slat in a 3-element airfoil. From the surface mesh $S = (X, A)$ we form an attributed graph $\mathcal{G} = (X, V, A)$ containing the node coordinates X , the node features $V \in \mathbb{R}^{N_S \times 4}$ with the node coordinates and the unit surface normals, and the adjacency matrix A .

In order to enable fast, multi-scale information propagation, the graph is successively coarsened into smaller representations. If there

are d levels of coarsening we denote the graphs by $\mathcal{G}^0, \dots, \mathcal{G}^d$ with adjacency matrices $A^0, \dots, A^d$. At each level $l = 0, \dots, d-1$ a seed node is chosen for each connected component of the graph. For each component, a breadth-first search (BFS) is conducted from the seed node to compute the geodesic distances from the initial seed to all other nodes, and then, by striding, a set of nodes $\mathcal{I}^l$ at each other level (bi-stride) of the BFS frontiers are selected and edges are formed to obtain the pooled graph. Fig. 1 illustrates the bi-stride selection on a cyclic graph and on a mesh. At the level $l+1$, the connectivity is determined by using second-order adjacency enhancement, meaning that edges are added so that pooled nodes maintain connections through paths of up to two hops. This can be expressed as

$$A^{l+1} = A^l A^l[\mathcal{I}^l, \mathcal{I}^l] \quad (12)$$

We add self-loops to the graphs to enable that each node's own features are propagated to coarser graphs with the downsampling algorithm explained later in this section. We also denote by $\mathcal{G}^{d+1}, \dots, \mathcal{G}^{2d}$ the graphs obtained during upsampling, where $\mathcal{G}^{l+1}$ has the same nodes positions and connectivity as $\mathcal{G}^{2d-(l+1)}$ for $l = d, \dots, 2d-1$.

2.4.1. Model structure

The Bi-Stride Multi-Scale Graph Neural Network (BSMS-GNN) follows an encoder-processor-decoder architecture as in MeshGraphNets. The encoder and the decoder are implemented by multi-layer perceptrons, while the processor is a multi-scale message passing graph neural network that follows a U-Net structure with bi-stride graph pooling. This is shown in Fig. 2 and explained in detail in what follows.

Encoder. First, an encoded representation of each node j is produced using an encoder multi-layer perceptron MLP_E . It takes as input the flow conditions, the node coordinates and normal and outputs a representation $\mathbf{v}_j^0$ given by

$$\mathbf{v}_j^0 = \text{MLP}_E(\mathbf{p}, \mathbf{x}_j, \mathbf{v}_j) \quad (13)$$

This layer produces a new attributed graph $\mathcal{G}^0 = (X^0, V^0, A^0)$ where V^0 is obtained after applying the encoder to all nodes.

Processor Message Passing. At level l of the graph coarsening, each node j has a representation $\mathbf{v}_j^l$ before message passing. Then, each neighbor i of node j sends it a message, as illustrated in Fig. 3a. The message is calculated using a shared message multi-layer perceptron MLP_{M_l} as

$$\mathbf{m}_{ij}^l = \text{MLP}_{M_l}(\mathbf{v}_i^l \oplus \mathbf{v}_j^l \oplus \Delta \mathbf{x}_{ij}^l \oplus |\Delta \mathbf{x}_{ij}^l|) \quad (14)$$

where $\Delta \mathbf{x}_{ij}^l$ is the displacement vector between node i and node j . Node j then sums the messages it received and creates the aggregated message $\mathbf{m}_j^l$ as expressed by

$$\mathbf{m}_j^l = \sum_{i \in \mathcal{N}^l(j)} \mathbf{m}_{ij}^l \quad (15)$$

This message aggregation is shown in Fig. 3b. A new representation $\tilde{\mathbf{v}}_j^l$ for node j is obtained by the sum of the current representation and the output of a shared update multi-layer perceptron MLP_{U_l} as

$$\tilde{\mathbf{v}}_j^l = \mathbf{v}_j^l + \text{MLP}_{U_l}(\mathbf{v}_j^l \oplus \mathbf{m}_j^l) \quad (16)$$

We refer to $\tilde{\mathbf{v}}_j^l$ as the representation of node j at level l after message passing, and illustrate this node update in Fig. 3c.

Processor Downsampling. First, at the 0th level node weights w_i^0 are provided at each node. This can correspond to the relative area weights defined previously or be all equal to one. At level l , sender edge weights $\hat{w}_{ij}^l$ from the sender node i to the receiver node j are obtained by normalizing the weight of the sender node i with respect to its number of neighbors via

$$\hat{w}_{ij}^l = \frac{w_i^l A_{ij}^l}{\sum_{j'} A_{ij'}^l} \quad (17)$$

This will assist in splitting the weighted information of the sender nodes and thus helps to spread information more evenly across the graph and

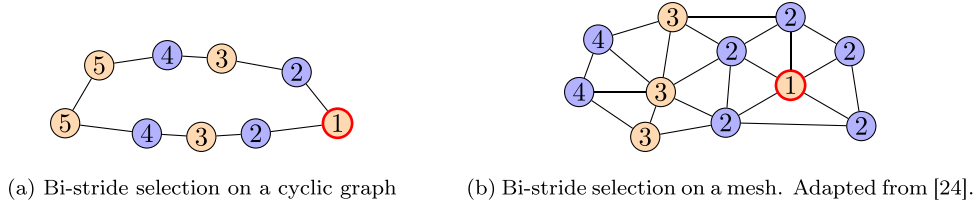


Fig. 1. Bi-stride selection. The seed node is circled in red. The selected nodes are shown in orange and unselected ones in purple. Each node shows its level in the BFS frontiers. Adapted from [24].

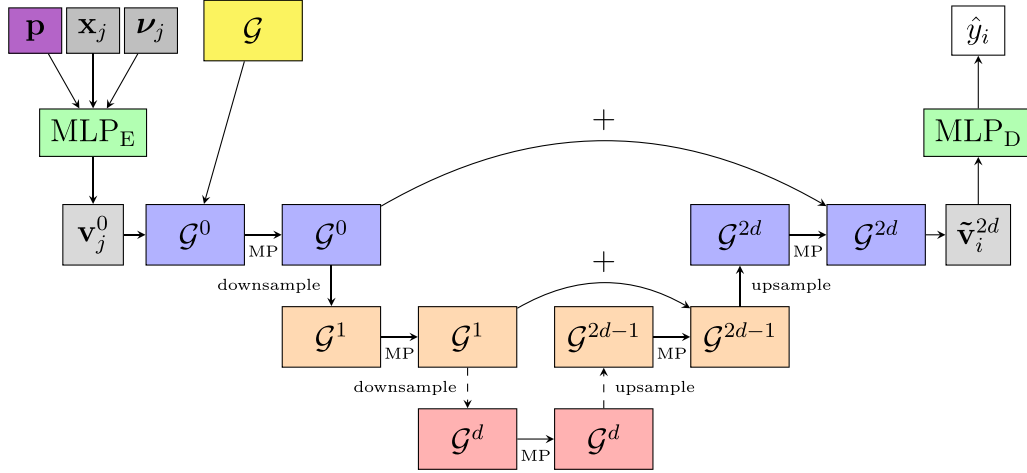


Fig. 2. Architecture of BSMS-GNN. The encoder multi-layer perceptron MLP_E needs to process all surface nodes j . Adapted from [24].

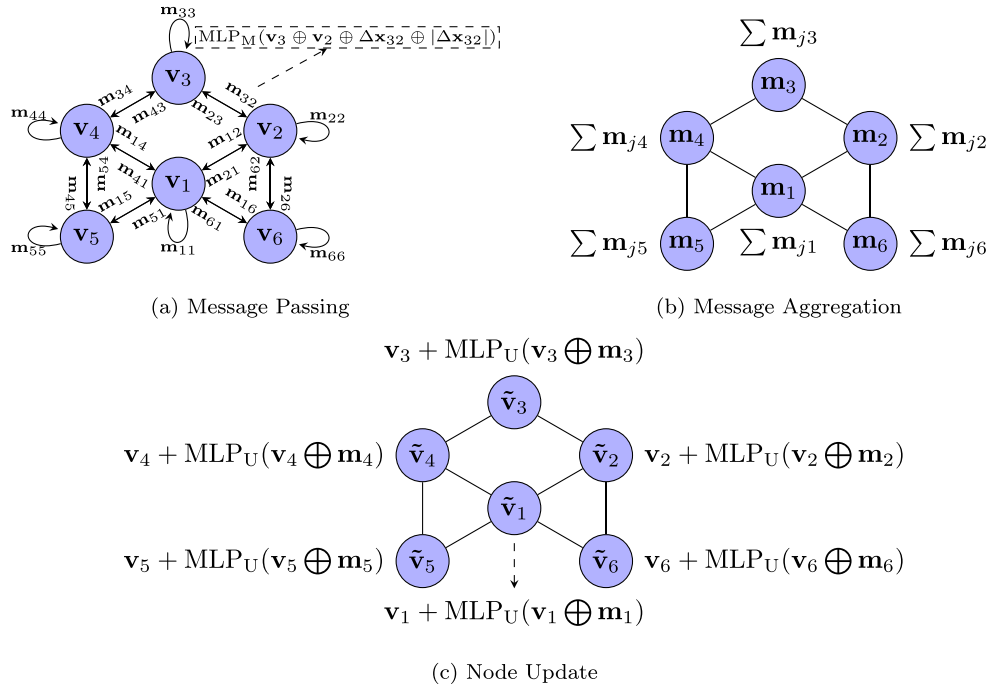


Fig. 3. Message Passing schematic representation. Adapted from [24].

prevents over-representation of high-degree nodes. Fig. 4a illustrates node and sender edge weights.

Next, contribution edge weights c_{ij}^l from node i to node j are calculated by normalizing the previously calculated sender edge weights $\hat{w}_{ij}^l$ such that the sum of incoming contribution edge weights for each

receiver node equals one. This is expressed by

$$c_{ij}^l = \frac{\hat{w}_{ij}^l}{\sum_{i'} \hat{w}_{i'j}^l} \quad (18)$$

Similarly, this ensures that each node receives information from its

neighbors on a consistent scale, regardless of how many connections it has. The weight w_j^{l+1} of node j at the next level $l+1$ is computed as the sum of the sender edge weights at the current level as given by

$$w_j^{l+1} = \sum_i \hat{w}_{ij}^l \quad (19)$$

Fig. 4b illustrates the updated node weights and the contribution edge weights.

Then, the representation $\mathbf{v}_j^{l+1}$ of node j at the next level $l+1$ before message passing is computed as a weighted sum of the representations after message passing of its neighbors at the current level l based on the contribution weights as

$$\mathbf{v}_j^{l+1} = \sum_i c_{ij}^l \tilde{\mathbf{v}}_i^l \quad (20)$$

Fig. 4c illustrates the computation of node features during downsampling. Similarly, the position of node j at the next level $l+1$ can be obtained as

$$\mathbf{x}_j^{l+1} = \sum_i c_{ij}^l \mathbf{x}_i^l \quad (21)$$

This is not shown in Fig. 4c for simplicity. It is also possible to keep the node positions fixed as originally introduced in [24].

Edge Completion at Coarsest Level. In contrast to [24], at the coarsest level d we introduce edges between all nodes in order to exchange messages between all connected components of the graph. If there is more than one connected component, this is essential to model the effect between the connected components.

Processor Upsampling. After message passing at the coarsest level d , the upsampling part of the processor begins. At level $l+1$ the representation of node i before message passing $\mathbf{v}_i^{l+1}$ is computed as the sum of the representations at level l after message passing of its neighbors with the contribution weights from node i to its neighbors. This is expressed by

$$\mathbf{v}_i^{l+1} = \sum_j c_{ij}^{2d-(l+1)} \tilde{\mathbf{v}}_j^l \quad (22)$$

Fig. 4d illustrates the computation of node features during upsampling. Next, message passing is performed as done in the downsampling section and summed with the representation from level $2d-(l+1)$ to obtain $\tilde{\mathbf{v}}_i^{l+1}$ as

$$\tilde{\mathbf{v}}_i^{l+1} = \tilde{\mathbf{v}}_i^{2d-(l+1)} + \left(\mathbf{v}_i^{l+1} + \text{MLP}_{U_{l+1}}(\mathbf{v}_i^{l+1} \oplus \mathbf{m}_i^{l+1}) \right) \quad (23)$$

The process repeats until reaching the level $2d$ with representations $\tilde{\mathbf{v}}_i^{2d}$.

Decoder. After the processor, a decoder multi-layer perceptron MLP_D is used to obtain a prediction of the pressure coefficient for each node i based on the representation $\tilde{\mathbf{v}}_i^{2d}$ as expressed by

$$\hat{y}_i = \text{MLP}_D(\tilde{\mathbf{v}}_i^{2d}) \quad (24)$$

The prediction of the surface pressure distribution is computed as

$$\hat{\mathbf{y}} = [\text{MLP}_D(\tilde{\mathbf{v}}_1^{2d}), \dots, \text{MLP}_D(\tilde{\mathbf{v}}_n^{2d})]^T \quad (25)$$

2.5. Implicit neural representation of the signed distance function coupled with a multi-layer perceptron

One model that has been explored when shape and position parameterizations are available consists of a multi-layer perceptron for the nodewise prediction of the pressure coefficient [7]. It uses as input the flow conditions $\mathbf{p}$, the node coordinates $\mathbf{x}_i$ and normal $\mathbf{v}_i$, together with the shape and position parameterization ξ . If such parameterization is not available, a natural extension of that model, is to first learn to parameterize airfoils, and then follow the same approach with the extracted representation $\mathbf{z}$ instead of the explicit parameterization ξ . The prediction for node i for an airfoil with extracted representation $\mathbf{z}$ and flow parameters $\mathbf{p}$ is computed as

$$\hat{y}_i = \text{MLP}(\mathbf{p}, \mathbf{x}_i, \mathbf{v}_i, \mathbf{z}) = \hat{f}_{\theta_p}(\mathbf{p}, \mathbf{x}_i, \mathbf{v}_i, \mathbf{z}) \quad (26)$$

where we denote the parameters of this predictor MLP by θ_p . To learn the representation $\mathbf{z}$ of airfoils we use an approach based on an implicit neural representation (INR) of the signed distance function (SDF) [17]. This approach captures the airfoil's geometry by learning a code $\mathbf{z}$ that represents it. This code $\mathbf{z}$ allows, coupled with a shared learned neural network, the prediction of the distance of any point to the airfoil's surface, with the sign indicating whether the point is inside or outside the surface, and thus serves as a compact representation of the airfoil's geometry.

2.5.1. Implicit neural representation of the signed distance function

Each configuration is associated with a domain Ω_j with a volume mesh $\mathcal{M}_j$, with the airfoil surface defined by $\partial\Omega_j$ that is discretized into the surface mesh S_j . We consider a bounded simply connected region $R_j \supset \partial\Omega_j$ containing the airfoil surface and its interior.

The signed distance function $u_j : R_j \rightarrow \mathbb{R}$ is learned by an INR $f_{\theta_u, \phi_{u_j}}$ with a multi-layer perceptron structure. The parameters θ_u are shared for all airfoils and their corresponding u_j , while the modulation parameters ϕ_{u_j} are specific to each individual airfoil. We denote by d the dimension of the latent representations $\mathbf{z}_{u_j} \in \mathbb{R}^d$. If the network $f_{\theta_u, \phi_{u_j}}$ consists of L hidden layers, each with n hidden nodes, the modulation parameters ϕ_{u_j} are determined by using a linear hypernetwork h_u , as

$$\phi_{u_j} = h_{\psi_u}(\mathbf{z}_{u_j}) = \left\{ V_u^l \mathbf{z}_{u_j} + \mathbf{c}_u^l \right\}_{l=0}^{L-1} \quad (27)$$

where each $V_u^l \in \mathbb{R}^{n,d}$ and $\mathbf{c}_u^l \in \mathbb{R}^n$. All the parameters of h_{ψ_u} are denoted by the vector ψ_u , shared for all airfoils. h_{ψ_u} is called a hypernetwork, because it is a network that generates parameters for another network, namely $f_{\theta_u, \phi_{u_j}}$. The latent codes $\mathbf{z}_{u_j}$ are learned through a procedure known as auto-decoding. This is explained in the next section that covers the training and inference algorithm of all networks involved. Given modulation parameters ϕ_{u_j} the network $f_{\theta_u, \phi_{u_j}}$ first pre-processes the query point $\mathbf{q}$ using a Fourier encoding vector $\hat{\mathbf{q}}$ as

$$\mathbf{B} \sim \mathcal{N}(0, \sigma^2 \mathbf{I}) \quad (28)$$

$$\hat{\mathbf{q}} = \sin(2\pi \mathbf{B} \mathbf{q}) \oplus \cos(2\pi \mathbf{B} \mathbf{q}) \oplus \mathbf{q} \quad (29)$$

where $\mathbf{B} \in \mathbb{R}^{F,2}$ is a matrix with independent entries drawn from a normal distribution with zero mean and a standard deviation $\sigma \in \mathbb{R}_+$ and $\hat{\mathbf{q}} \in \mathbb{R}^{2F+2}$. The rest of the network proceeds as a standard MLP, but at each hidden layer the corresponding modulation parameters are added before applying the non-linearity. The modulated MLP is illustrated in Fig. 5 and summarized as

$$\eta^0 = \hat{\mathbf{q}} \quad (30)$$

$$\chi^l(\eta^l) = \sigma(W^l \eta^l + \mathbf{b}^l + (\phi_{u_j})_l) \quad (31)$$

$$f_{\theta_u, \phi_{u_j}}(\mathbf{q}) = W^L(\chi^{L-1} \circ \chi^{L-2} \circ \dots \circ \chi^0(\hat{\mathbf{q}})) + \mathbf{b}^L \quad (32)$$

2.5.2. Training and inference algorithms

Three datasets, $\mathcal{D}_{\text{train}}^{\text{INR}}$, $\mathcal{D}_{\text{val}}^{\text{INR}}$, $\mathcal{D}_{\text{test}}^{\text{INR}}$ are created to train, validate and test the signed distance function INR submodel. For each airfoil S_j , n_j query points $\mathbf{q}_i^j \in \mathbb{R}^2$ are sampled from the region R_j and the corresponding signed distance value $u_j(\mathbf{q}_i^j) \in \mathbb{R}$ is computed. In practice, various points are sampled close to the surface S_j in addition to points uniformly sampled from the region R_j , in order to obtain an accurate representation of the shape. The train dataset for the INR is given by

$$\mathcal{D}_{\text{train}}^{\text{INR}} = \{ \{ (\mathbf{q}_i^j, u_j(\mathbf{q}_i^j)) \}_{i=1}^{n_j} \}_{j \in \mathcal{J}_{\text{D}_{\text{train}}}} \quad (33)$$

and the other datasets are generated analogously. The loss function $\mathcal{L}_u$ corresponds to the mean squared error, so that for a prediction $\hat{\mathbf{u}}$ of the signed distances $\mathbf{u}$ for n points the loss is given by

$$\mathcal{L}_u(\mathbf{u}, \hat{\mathbf{u}}) = \frac{1}{n} \|\hat{\mathbf{u}} - \mathbf{u}\|_2^2 \quad (34)$$

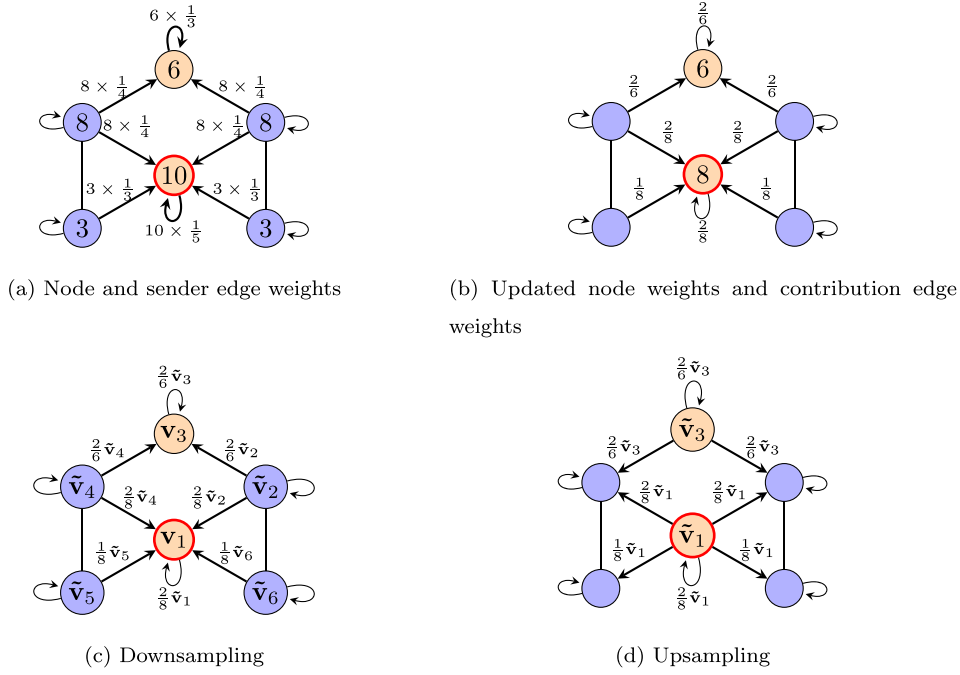


Fig. 4. Downsampling and upsampling schematic example. Node position remains fixed for simplicity in the illustration. Adapted from [24].

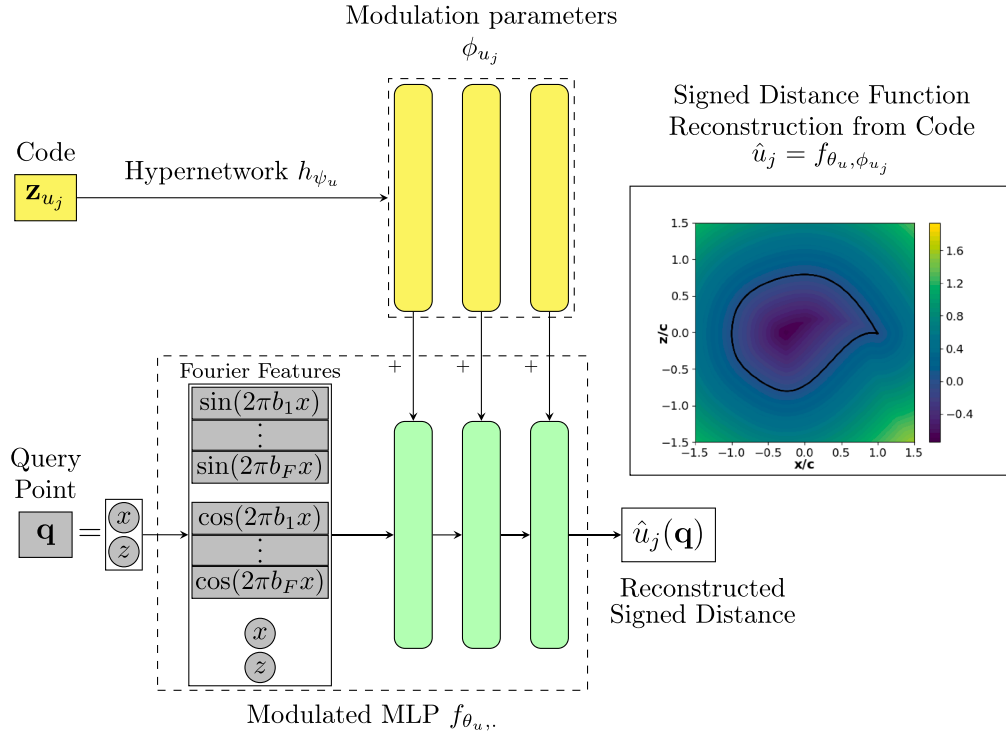


Fig. 5. Modulated Multi-Layer Perceptron (MLP) with Fourier features as an implicit neural representation (INR) of the signed distance function (SDF). The MLP shows 3 hidden layers as an example.

The INR is trained using a second-order meta-learning algorithm based on CAVIA [45] shown in Algorithm 1. It consists of an interleaved training procedure. In the inner loop only the latent representations $\mathbf{z}_{u_j}$ are updated using a few K gradient inner steps, as show in lines 9–15. Then, in the outer loop the shared parameters θ_u of the multi-layer perceptron and the shared parameters ψ_u of the linear hypernetwork are updated using Adam, as shown in lines 16–19. This requires back-propagating gradients through the K inner steps and computing

gradients of gradients. Using only a few gradient steps in the inner loop reduces computational cost and prevents the INR from memorizing the training samples into individual codes. Overall, the training algorithm allows to optimize the shared parameters θ_u and ψ_u , in a way that enables the quick computation of latent representations during inference time [17,45].

Once the INR is trained, the predictor MLP can be trained as usual. The complete inference procedure for INR-MLP is illustrated in Fig. 6.

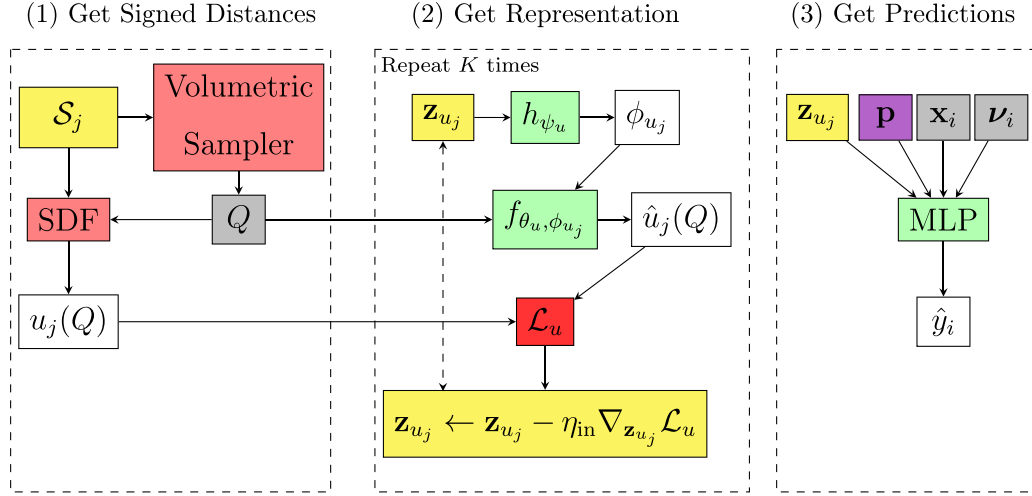


Fig. 6. Inference procedure of INR-MLP. For the first iteration in (2) the representation is initialized as a vector of zeroes. The multi-layer perceptron MLP does **not** need to process all surface nodes at once.

Algorithm 1 Training algorithm of INR (h_{ψ_u} and f_{θ_u}).

```

1: Input  $E, B, n, K, \eta_{out}, \eta_{in}$ 
2: for  $e = 1, \dots, E$  do
3:    $\mathcal{J} \leftarrow \mathcal{D}_{train}$ 
4:   while  $\mathcal{J} \neq \emptyset$  do
5:     Sample a set  $\mathcal{J}_B \subset \mathcal{J}$  of  $B$  train airfoil indices
6:     Sample a set of  $I_j$  of  $n$  point indices for  $j$  in  $\mathcal{J}_B$ 
7:      $\mathbf{u}_j \leftarrow (u_j(\mathbf{q}_{i_1}^j), \dots, u_j(\mathbf{q}_{i_n}^j))$  for  $j$  in  $\mathcal{J}_B$  where  $I_j = (i_1, \dots, i_n)$ 
8:      $\mathbf{z}_{u_j} \leftarrow \mathbf{0}$  for  $j$  in  $\mathcal{J}_B$ 
9:     for  $k = 1, \dots, K$  do
10:      for  $j \in \mathcal{J}_B$  do ▷ Done in parallel
11:         $\phi_{u_j} \leftarrow h_{\psi_u}(\mathbf{z}_{u_j})$ 
12:         $\hat{\mathbf{u}}_j \leftarrow (f_{\theta_u, \phi_{u_j}}(\mathbf{q}_{i_1}^j), \dots, f_{\theta_u, \phi_{u_j}}(\mathbf{q}_{i_n}^j))$ 
13:         $\mathbf{z}_{u_j} \leftarrow \mathbf{z}_{u_j} - \eta_{in} \nabla_{\mathbf{z}_{u_j}} \mathcal{L}_u(\mathbf{u}_j, \hat{\mathbf{u}}_j)$  ▷ Stochastic gradient descent
14:      end for
15:    end for
16:     $\phi_{u_j} \leftarrow h_{\psi_u}(\mathbf{z}_{u_j})$  for  $j$  in  $\mathcal{J}_B$ 
17:     $\hat{\mathbf{u}}_j \leftarrow (f_{\theta_u, \phi_{u_j}}(\mathbf{q}_{i_1}^j), \dots, f_{\theta_u, \phi_{u_j}}(\mathbf{q}_{i_n}^j))$  for  $j$  in  $\mathcal{J}_B$ 
18:     $\theta_u \leftarrow \theta_u - \eta_{out} \frac{1}{|\mathcal{J}_B|} \sum_{j \in \mathcal{J}_B} \nabla_{\theta_u} \mathcal{L}_u(\mathbf{u}_j, \hat{\mathbf{u}}_j)$  ▷ In practice with Adam
19:     $\psi_u \leftarrow \psi_u - \eta_{out} \frac{1}{|\mathcal{J}_B|} \sum_{j \in \mathcal{J}_B} \nabla_{\psi_u} \mathcal{L}_u(\mathbf{u}_j, \hat{\mathbf{u}}_j)$  ▷ In practice with Adam
20:     $\mathcal{J} \leftarrow \mathcal{J} \setminus \mathcal{J}_B$ 
21:  end while
22: end for

```

First, query points are sampled around the airfoil S_j and the signed distances are computed. Next, the latent representation or code $\mathbf{z}_{u_j}$ is obtained by using K steps of gradient descent exclusively on $\mathbf{z}_{u_j}$, with the fixed learned shared parameters θ_u and ψ_u . Then, the prediction at some surface point with coordinates $\mathbf{x}_i$ and normal $\mathbf{v}_i$ is computed by the predictor MLP by concatenating the latent representation $\mathbf{z}_{u_j}$ and the flow parameters $\mathbf{p}$ to the input.

3. Test cases

The DLR's Surrogate Modeling for AeRo data Toolbox in python (SMARTy) [46] is used throughout for the generation of the dataset and the creation of the surrogate models.

3.1. Generic airfoils dataset

The first test case consists of two-dimensional airfoils that are variants of the transonic airfoil RAE2822. We refer to this dataset as the Generic Airfoils Dataset. The dataset consists of pressure coefficient distributions at various flow conditions for different airfoils. The flow conditions are specified by the Mach number M , the angle of attack α , and the Reynolds number Re .

3.1.1. Geometric parameterization

The airfoils are produced using either a Class-Shape Transformation (CST) or a Non-Uniform Rational Basis Spline (NURBS) parameterization. CST geometries are obtained with 9 coefficients, while NURBS geometries are generated with 19 (free) coefficients.

3.1.2. Software

The airfoils are created using the Airfoil Parameterization Toolbox. The DLR-CFD Software TAU is used as the flowsolver, that employs a vertex-center scheme. The Reynolds Averaged Navier Stokes Equations (RANS) with the negative Spalart-Allmaras turbulence model (SA-neg) are solved. All simulations run until a density residual below 1×10^{-8} is reached or a maximum of 100,000 iterations is reached. Simulations that do not reach the target density residual are considered not converged and are discarded.

3.1.3. Dataset description

Each sample in the dataset consists of the flow conditions, the airfoil geometric parameters, the surface mesh and the corresponding surface pressure coefficient distribution. The dataset $\mathcal{D}$ with 1198 samples is given by

$$\mathcal{D}^{CST} = \{(M_i, \alpha_i, Re_i, \xi_i^{CST}, S_i, \mathbf{y}_i) : 1 \leq i \leq 598\} \quad (35)$$

$$\mathcal{D}^{NURBS} = \{(M_i, \alpha_i, Re_i, \xi_i^{NURBS}, S_i, \mathbf{y}_i) : 1 \leq i \leq 600\} \quad (36)$$

$$\mathcal{D} = \mathcal{D}^{CST} \cup \mathcal{D}^{NURBS} \quad (37)$$

The range of the flow parameters is shown in Table 1, while Fig. 7 shows sample airfoils for the CST and NURBS parameterizations. For each parameterization, 600 combinations of flow and geometric parameters are created with a Sobol sequence within their bounds. For CST, out of the 600 combinations, 598 simulations were successfully generated, while for NURBS all 600 combinations lead to converged simulations. As seen in Fig. 8, this results in the same combinations of Mach number and angle of attack for both parameterizations, except for the 2 discarded simulations for CST. Although all airfoils were generated from

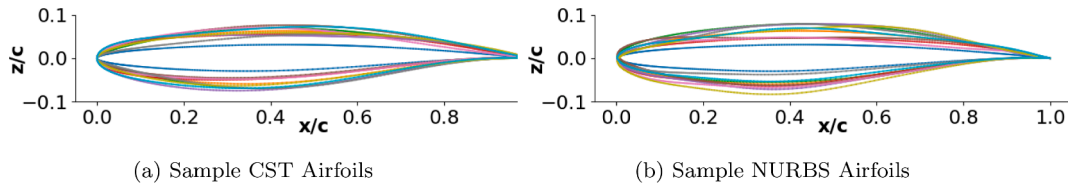


Fig. 7. Sample Generic Airfoils. 10 airfoils are shown for each parameterization.

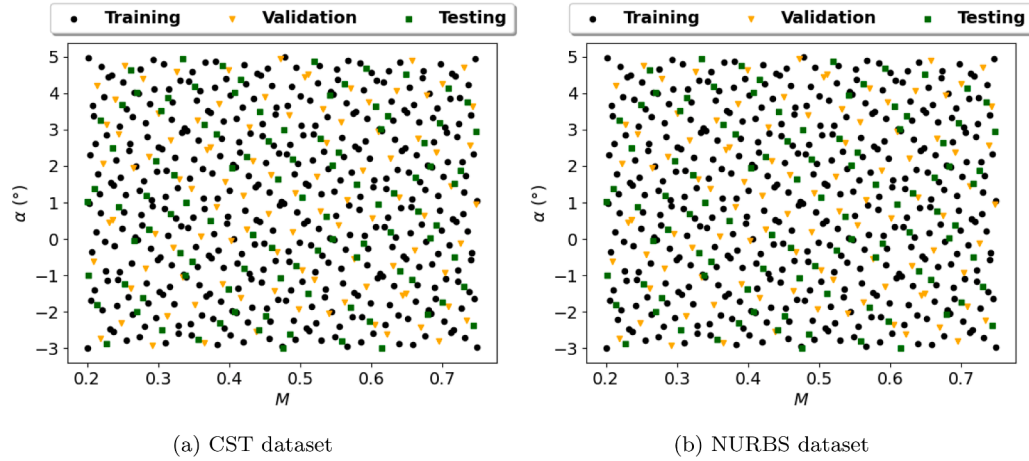


Fig. 8. Sampling plan shown in the Mach-Alpha plane for the full data regime for the Generic Airfoils Dataset. Each point shown corresponds to a different airfoil and Reynolds number.

Table 1

Range of the parameters for the generic airfoils dataset.

Parameter	Notation	Minimum	Maximum
Mach number	M	0.20	0.75
Angle of attack	α	-3.0°	5.0°
Reynolds number	Re	1.0×10^6	6.0×10^6
Geometric coefficient j	ξ_j	$\min(\xi_j)$	$\max(\xi_j)$

Table 2

Number of training samples for each data regime for the Generic Airfoils Dataset. The validation and test datasets remain constant with 200 samples, 100 from each parameterization.

Dataset	R1	R2	R3	R4
Train CST	398	300	200	100
Train NURBS	400	300	200	100
Train Total	798	600	400	200

the RAE2822 baseline, the resulting dataset exhibits a diverse range of airfoil shapes. The airfoils generated with the CST parameterization are the same ones as in the AIAA surrogate model benchmark data available online [37].

3.1.4. Dataset partitioning

Four different splits of the data are created to evaluate the performance of the models with different training sizes. In the full-data regime the train dataset consists of 798 samples, 398 samples from the CST dataset and 400 from the NURBS dataset. The validation and test datasets remain constant with 200 samples, 100 from each parameterization. Table 2 details the number of training samples per parameterization on each data regime. Fig. 8 shows the sampling plan shown in the Mach-Alpha plane for the full-data regime.

3.2. DLR-F15 multi-element airfoils dataset

The second test case features a two-dimensional multi-element airfoil based on the DLR-F15 model with a fixed geometry but a changing configuration. This model was set up in 2005 by the German Aerospace Center (DLR) for various purposes in high-lift research. It has been used for investigations of wind-tunnel side effects and corrections, noise-reduced high-lift systems, flow-separation control, sensitivities on geometrics, and the sensitivity on flow conditions [44]. The aim of this technology is to reduce the flight speed during the take-off, climb, approach and landing flight phases in proximity to the ground. This is achieved by increasing the maximum lift coefficient, which enables the generation of a required lift force at a lower speed [47].

The reference configuration is a three-element configuration named F15-3eRef with a slat deployed at 28° and a flap deployed at 35° depicted in Fig. 9. This wing section is taken directly from the reference 3D wing and corresponds to the landing configuration. The mesh consists of 132,333 nodes, of which 924 correspond to surface nodes.

Here we create a dataset that consists of pressure coefficient distributions at various flow conditions for distinct configurations obtained by modifying the position of the flap of the F15-3eRef landing configuration, while keeping the slat and main element at fixed positions. The flow conditions are specified by the Mach number M and the angle of attack α . The Reynolds number is fixed for all simulations at 7×10^6 . We refer to this dataset as the DLR-F15 Multi-Element Airfoils Dataset.

For the results that we present next, we consider only the full-data regime and for each model we select the network with the best validation MPSAWE. In Fig. 16 we plot the MPSAWE of BSMS-GNN and INR-MLP in the $M\alpha$ plane and the $\delta - \text{ovl}$ plane for the test samples.

3.2.1. Configuration parameterization

The position of the flap can be described by its position relative to the main element [47]. From an aerodynamics standpoint it is most advantageous to use the deflection angle together with the characteristic of the slot as specified by the overlap and gap definitions. Fig. 10 illustrates the parameterization for an exemplary position of the flap. The

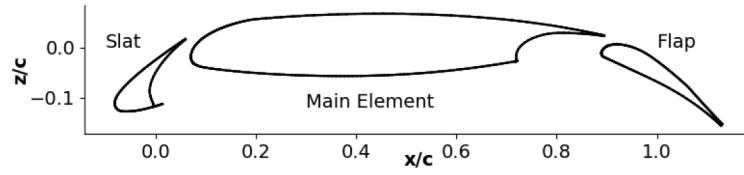
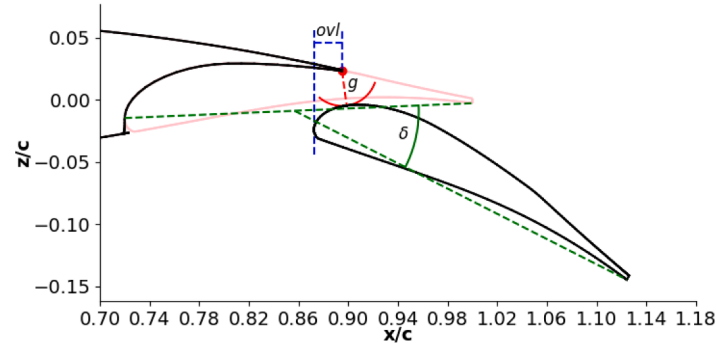
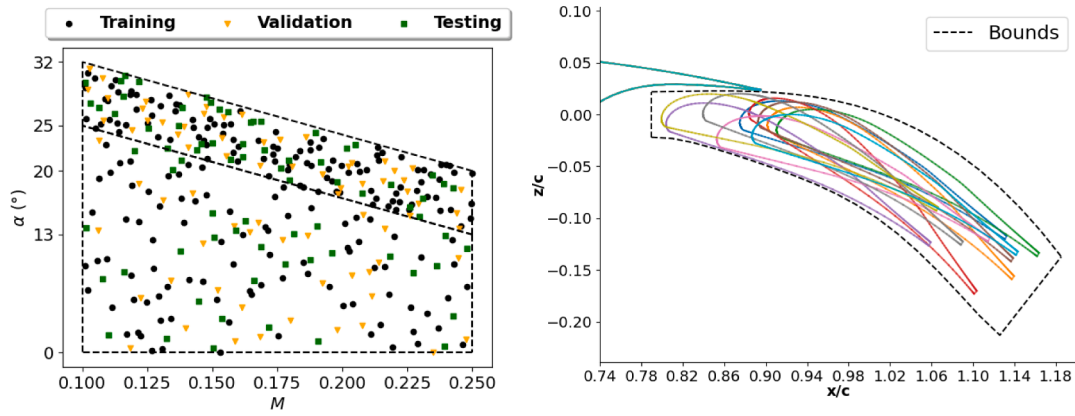


Fig. 9. F15-3eRef landing configuration.

Fig. 10. Flap parameterization illustrating the deflection angle δ , the overlap ovl and the gap g . Parameters do **not** correspond to the F15-3eRef landing configuration.(a) Samples in the $M\alpha$ -plane. Each point corresponds to a different flap position.

(b) Sample flap positions. Only 10 positions are shown for clarity.

Fig. 11. Sampling plan in the full-data regime.

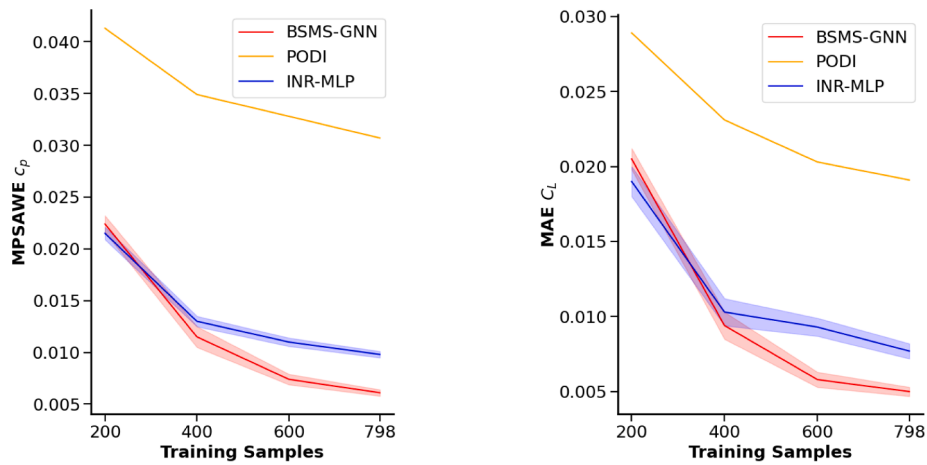
(a) MPSAWC c_p vs Training Samples(b) MAE C_L vs Training Samples

Fig. 12. Error metrics by train size for the Generic Airfoils Dataset. Curves are linear interpolations between measured points for visualization purposes.

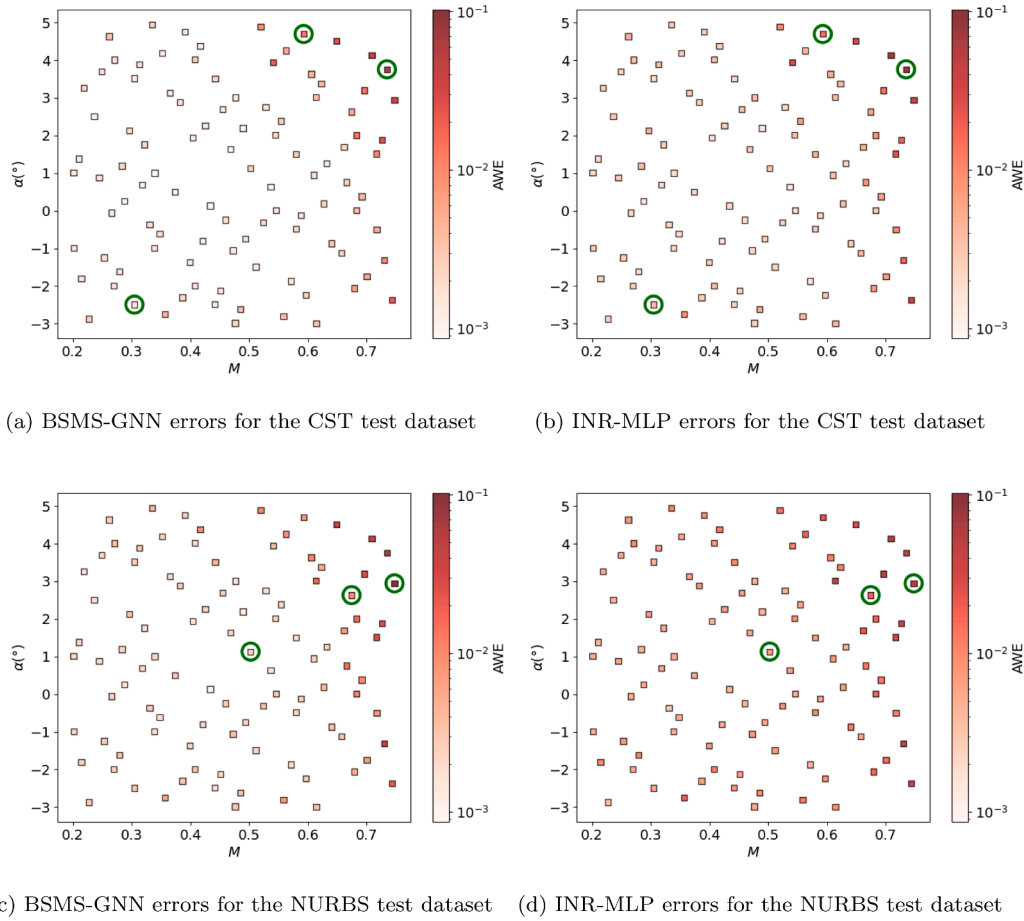


Fig. 13. Area-weighted error of BSMS-GNN and INR-MLP for each sample shown in the $M\alpha$ plane. Selected test samples are circled (green).

deflection angle δ corresponds to the angle between the chord line of the deployed flap and the chord line of the retracted flap. The overlap ovl is defined as the distance between the vertical projection of the trailing edge of the main element to the leading edge of the flap at the vertical tangent. It is positive if there is real overlap and negative otherwise. The gap g is the minimum distance of the trailing edge of the main element to the flap. The overlap and the gap are reported as the percentage of the chord length of the airfoil with the slat and flap retracted.

3.2.2. Software

The DLR's parametric grid generation system MegaCADs [48] is employed to generate the different meshes via mesh deformation from the reference mesh. The CFD Software by ONERA, DLR and Airbus (CODA) is used as the flowsolver. CODA is the CFD software being developed as part of a collaboration between the French Aerospace Lab ONERA, DLR, Airbus, and their European research partners. CODA is jointly owned by ONERA, DLR and Airbus. CODA is a cell-centered solver. The Reynolds Averaged Navier Stokes Equations (RANS) with the negative Spalart-Allmaras turbulence model (SA-neg) are solved. All simulations run until a density residual below 1×10^{-8} is reached or a maximum of 1500 iterations is reached. Simulations that do not reach the target density residual are considered not converged and are discarded.

3.3. Postprocessing of CFD simulations

First, the CFD solution from the cell-centered solver is interpolated from the center of the surface elements to the surface nodes. Next, since peaks at the trailing edges are known to be an artifact from the CFD-solver, the values at the upper and lower trailing edge points are substituted with the result of a linear interpolation using their two nearest

neighbors at the upper and lower side of the airfoil's element, respectively. Then, all inner nodes at the trailing edges are removed from the solution. The post-processed surface solution consists of $N = 831$ nodes. Note that for calculating the lift coefficient, linear interpolations at the trailing edges are used to get pressure coefficient values at the inner nodes in the trailing edges before performing the integration.

3.3.1. Dataset description

Each sample in the dataset consists of the flow conditions, the flap configuration parameters, the surface mesh and the corresponding surface pressure coefficient distribution. The dataset $\mathcal{D}$ with 348 samples is given by

$$\mathcal{D} = \{(M_i, \alpha_i, \delta_i, ovl_i, g_i, S_i, \mathbf{y}_i) : 1 \leq i \leq 348\} \quad (38)$$

Samples are created jointly based on 5 design parameters, which are the flow conditions M and α and the flap parameters δ, ovl, g . A minimum and maximum value are selected for each of the 5 design parameters as shown in Table 3. The $M\alpha$ region of interest is divided into two convex regions as displayed in Fig. 11a. The first $M\alpha$ -region corresponds to lower angles of attack for each Mach number, whereas the second one has higher angles of attack for each Mach number. Not all combinations of (δ, ovl, g) lead to geometrically feasible configurations. In addition, not all combinations describe configurations that are favorable from an aerodynamic point of view, as they can lead to increased drag, flow separation, and reduced lift generation. In order to restrict the flap positions to more aerodynamically favorable positions a region in the xz -plane is defined within which the flap should lie. Fig. 11b shows the bounds of this region. For some of the geometrically feasible configurations with the flap lying within the designated region the mesh deformation process still fails.

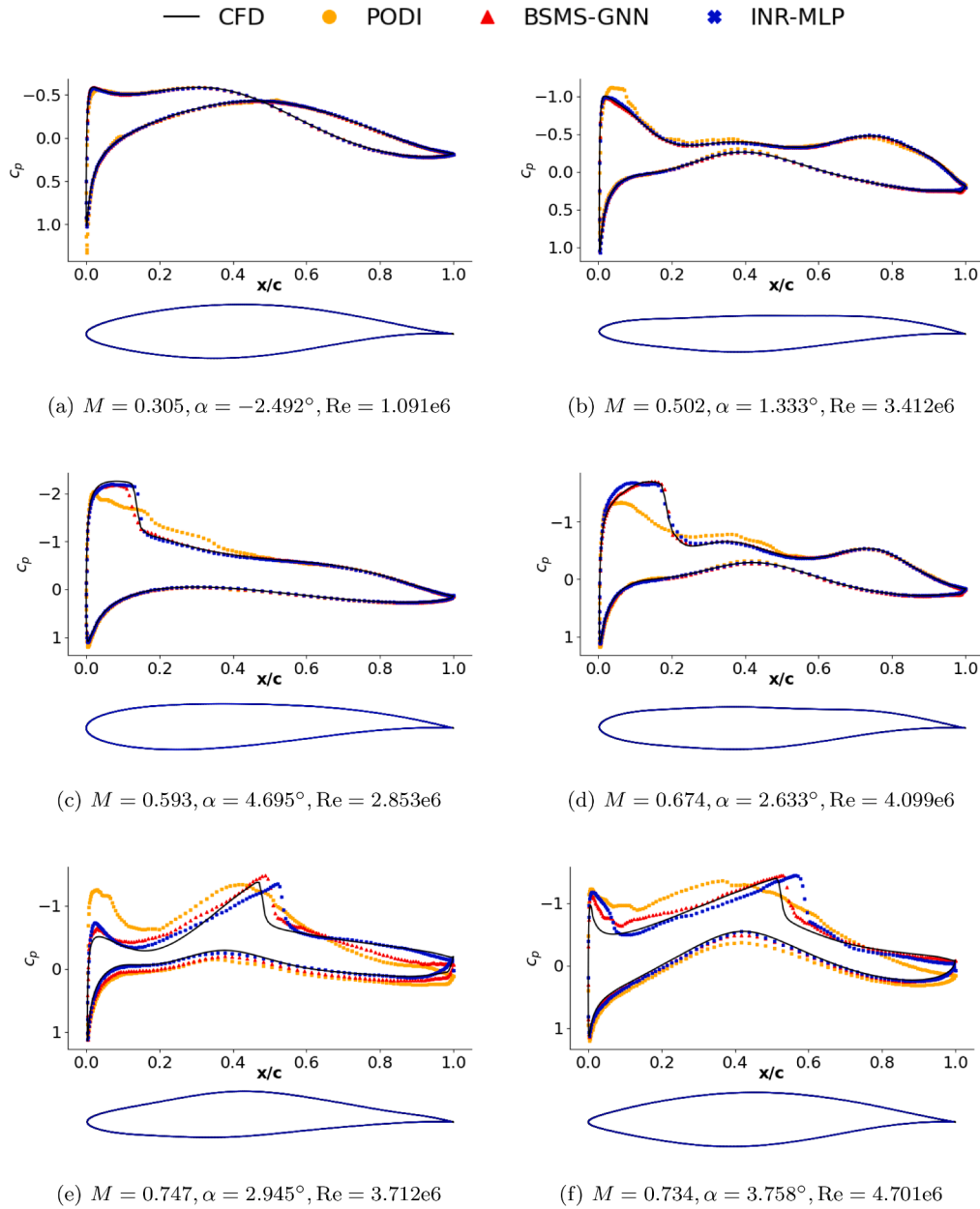


Fig. 14. Surface pressure distribution for selected test samples of the Generic Airfoils Dataset. The reconstruction of the airfoil from the code obtained with INR-MLP is shown in blue, overlaid on the reference airfoil in black.

For the lower $M\alpha$ -region, a Halton sequence of 454 ($M, \alpha, \delta, ovl, g$)-samples is created, of which 165 fulfill the requirement of being geometrically feasible and the flap lying in the designated region. Of these samples, 154 meshes were successfully generated and 152 simulations converged. For the upper $M\alpha$ -region, a Sobol sequence of 606 ($M, \alpha, \delta, ovl, g$)-samples is created, of which 213 fulfill the requirement of being geometrically feasible and the flap lying in the designated region. Of these samples, 196 meshes were successfully generated and 196 simulations converged. Thus in total there are 348 samples in the dataset, 152 in the lower $M\alpha$ -region and 196 in the upper one.

3.3.2. Dataset partitioning and data regimes

Four different splits of the data are created to evaluate the performance of the models with different training sizes. In the full-data regime the train dataset consists of 200 samples. From the lower $M\alpha$ -region the first 88 samples are selected and from the higher one the first 112 samples. The validation dataset consists of 74 samples, the next 32 from the

Table 3

Range of the parameters for the DLR-F15 multi-element airfoils dataset.

Parameter	Notation	Minimum	Maximum
Mach number	M	0.10	0.25
Angle of attack	α	0.0°	32.0°
Deflection angle	δ	10.0°	45.0°
Overlap	ovl	-2%	10%
Gap	g	0.5%	4.0%

lower region, and the next 42 of the higher one. The test dataset consists of 74 samples too, the last 32 from the lower region, and the last 42 of the higher one. This split corresponding to the full-data regime is the one shown in Fig. 11a. For the data regimes, the number of training samples is reduced while the validation and test datasets remain fixed.

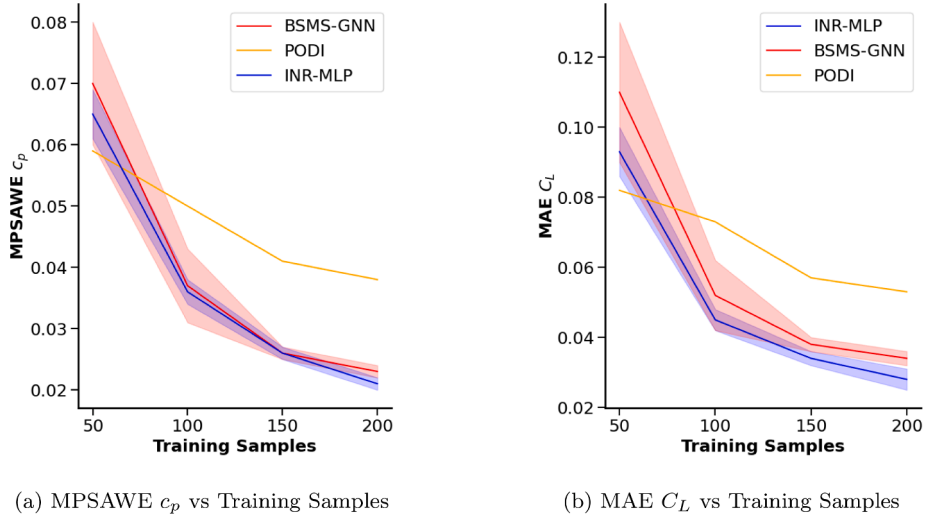


Fig. 15. Error metrics by train size for the DLR-F15 Multi-Element Airfoils. Lines are linear interpolations between measured points for visualization purposes.

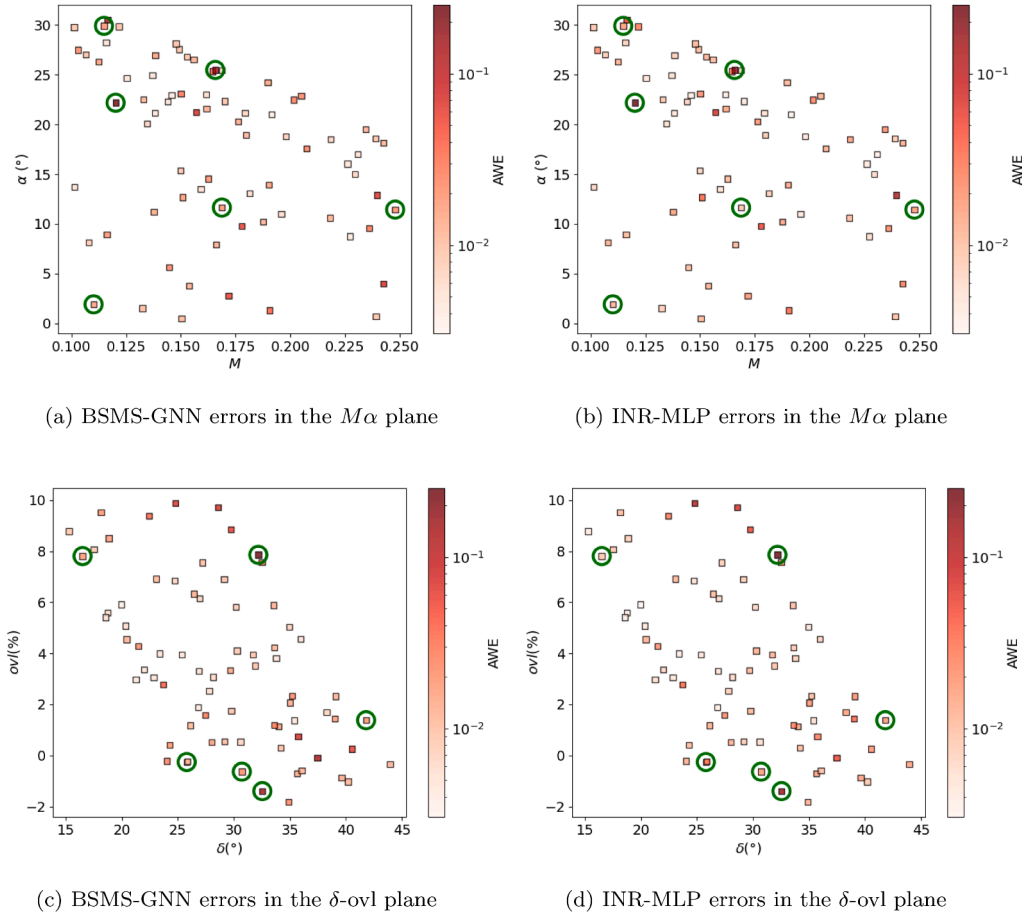


Fig. 16. Area-weighted error of BSMS-GNN and INR-MLP for each sample. Selected test samples are circled (green).

Table 4 details the number of training samples per region on each data regime.

4. Results

In this section we present the results for the Generic Airfoils Dataset and the DLR-F15 Multi-Element Airfoils Dataset for the prediction of surface pressure distributions using PODI, BSMS-GNN and INR-MLP. Sup-

plementary visualizations regarding the codes learned with the implicit neural representation of INR-MLP are presented in B. The performance of the models is assessed based on the predictions on test samples that are not seen during training and validation of the models. We consider four different regimes according to the number of training samples. The model hyperparameters are optimized based on the performance on the validation samples when trained with the training samples according to the full-data regime. Then, for each the training sizes, we train 16 net-

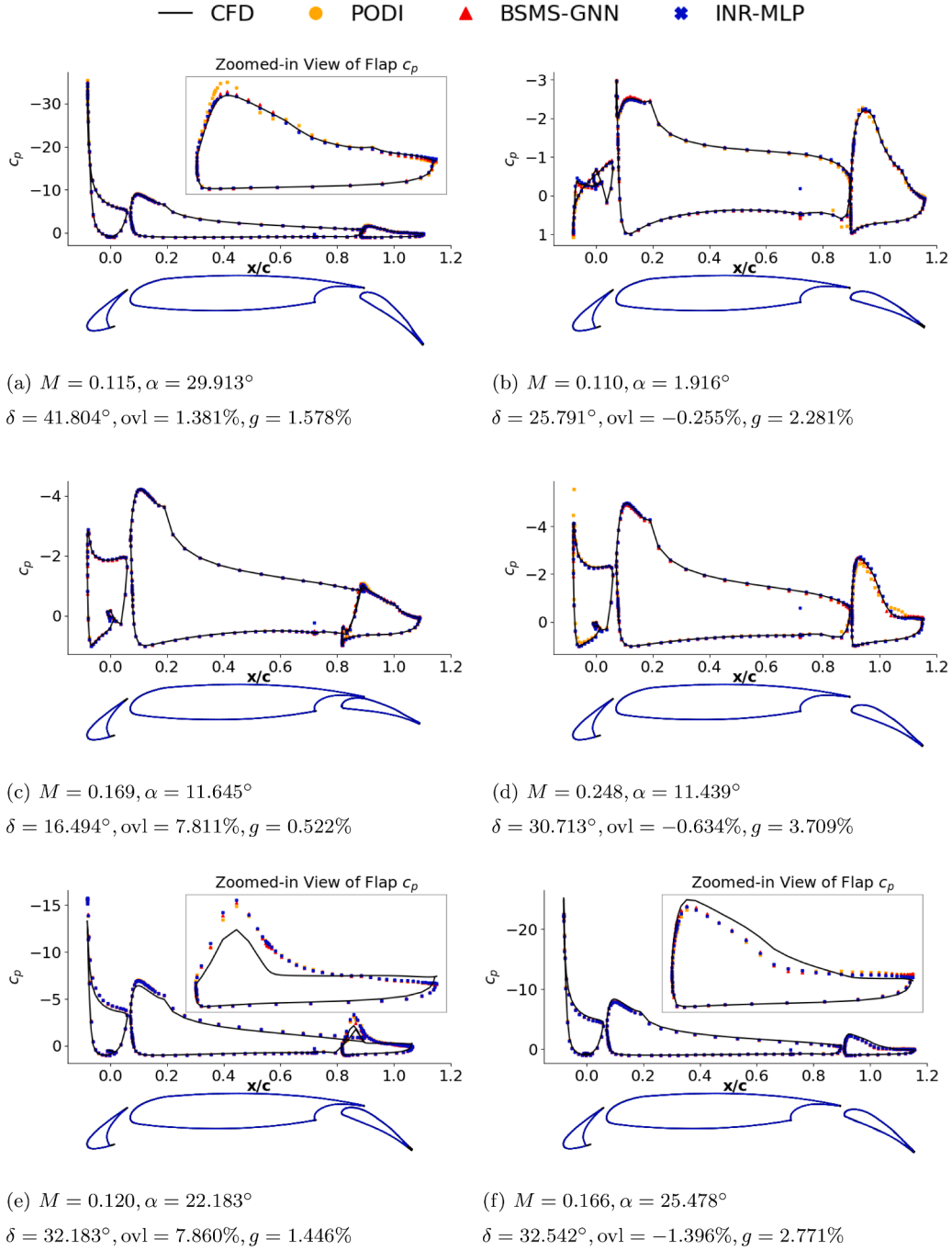


Fig. 17. Surface pressure distribution for selected test samples of the DLR-F15 Multi-Element Airfoils Dataset. The reconstruction of the airfoil from the code obtained with INR-MLP is shown in blue, overlaid on the reference airfoil in black.

works with different random seeds and report the average of the errors together with their standard deviations. The optimized hyperparameters are shown in the appendix in [Tables A.1–A.4](#). The error metrics that we use are described next.

4.1. Global error metrics

To evaluate the performance of the models in the prediction of the pressure coefficient (c_p) distributions we use the mean per-sample area-weighted error (MPSAWE). One airfoil S with M components is described by a union of curves $C = C_1 \cup \dots \cup C_M$ with total length $L = \sum_k^M \text{length}(C_k)$. The area-weighted error (AWE) approximates the mean

absolute error along the airfoil as

$$\text{AWE} := \frac{1}{\sum_{i=1}^N \Delta s_i} \sum_{i=1}^N |e_i| \Delta s_i \approx \frac{1}{L} \sum_{k=1}^M \int_{C_k} |e(s)| ds \quad (39)$$

where the error $e_i = |c_{p,i}^{\text{Pred}} - c_{p,i}^{\text{CFD}}|$ is the absolute error at node i of the prediction $c_{p,i}^{\text{Pred}}$ with respect to the CFD value $c_{p,i}^{\text{CFD}}$, and Δs_i is the length of the segment corresponding to the node.

The area-weighted error corresponds to an approximation of the expected absolute error of the predicted pressure coefficient when sampling a point uniformly at random along the airfoil surface, thus yielding a spatially fair assessment of the model accuracy across the entire contour. Unlike the standard MAE, which averages errors uniformly across

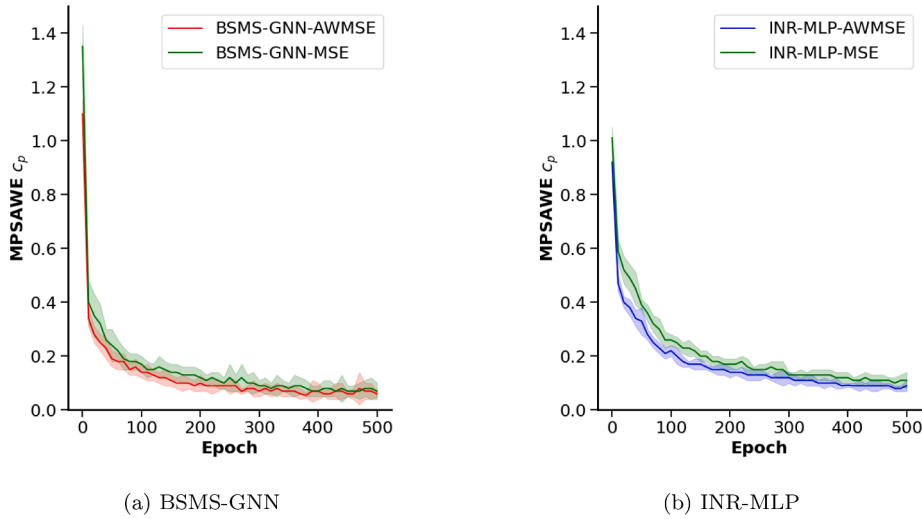


Fig. 18. Error history on validation samples for the first 500 epochs. The metric is obtained every ten epochs, the average is shown at each epoch with its standard deviation.

Table 4

Number of training samples for each data regime for the DLR-F15 Multi-Element Airfoils Dataset. The validation and test datasets remain constant with 74 samples, 32 from the lower region and 42 from the upper region.

Dataset	R1	R2	R3	R4
Train 1	88	66	44	22
Train 2	112	84	56	28
Train Total	200	150	100	50

points, this metric weights each error by the length of the segment it lies on, hence accounting for irregular spacing and varying element sizes. This metric is relatively insensitive to mesh density variations, regions with more density of nodes do not disproportionately influence the results, making the measure more robust to re-meshing and non-uniform point distributions. This is in particular useful for the DLR-F15 Multi-Element Airfoils Dataset where the meshes feature a high mesh density near the sealing lips of the slat and main element. In addition, the point distribution varies significantly for each sample, resulting in different areas of high mesh density.

The mean per-sample area-weighted error computed for a dataset $\mathcal{D}$ of samples is obtained by taking the mean of the area weighted error (AWE) of the samples, as described by

$$\text{MPSAWC}_p := \frac{1}{|\mathcal{D}|} \sum_{j=1}^{|\mathcal{D}|} \text{AWE}_j \quad (40)$$

We use the term *per-sample* to emphasize that the area-weighting is done for each airfoil (sample) to calculate its area weighted error before computing the mean of these values. In this manner, each sample is equally important towards the total error.

We are also interested in evaluating how well the lift coefficients obtained by integration of the predicted pressure distributions approximate the lift coefficient obtained with the CFD distribution. To this end, we consider the absolute error (AE) of the lift coefficient prediction for an individual sample and the mean absolute error (MAE) for multiple samples.

4.2. Local error metrics

While the previous global errors in the pressure distributions provide an overall measure of the performance of the models, the prediction of shocks in the transonic regime is of particular importance. These localized discontinuities heavily influence aerodynamic performance, thus it is essential to assess whether the models can accurately capture their presence and location. We adopt an approach for quantifying the prediction accuracy regarding shock locations based on a method introduced in [49].

For an airfoil S , we consider the pressure coefficient over its upper surface to be given by a function $cp: [0, 1] \rightarrow \mathbb{R}$. The domain is assumed to be discretized with the x -coordinates of the surface nodes of the upper surface $x_1 < x_2 < \dots < x_{N_U}$. We start with the set of candidates for shock regions $C_S = \{x_1, x_2, \dots, x_{N_U}\}$ and determine the location x_m at which the pressure coefficient derivative is maximum

$$x_m = \arg \max_{z \in C_S} \frac{d(cp)}{dx}(z) \quad (41)$$

Then, we find the first location $x_s \in C_S$ where the pressure coefficient derivative exceeds a certain proportion γ_1 of the maximum derivative, and that lies within a distance $\epsilon_{\max}$ of x_m .

$$\frac{d(cp)}{dx}(x_s) > \gamma_1 \frac{d(cp)}{dx}(x_m) \quad (42)$$

$$0 \leq x_m - x_s \leq \epsilon_{\max} \quad (43)$$

This is the candidate to be the first shock location found, and the second condition is used for the method to identify a candidate shock position that is near the point of maximum gradient.

The candidate shock end location $x_e \in C_S$ is determined, by finding the first location after the location x_m of maximum derivative where the derivative goes below a proportion γ_2 of the maximum, lies within a distance $w_{\max}$ of x_s , and such that the jump in the pressure coefficient from the shock location x_s to x_e exceeds a minimum value $\Delta cp_{\min}$.

$$x_m < x_e \quad (44)$$

$$\frac{d(cp)}{dx}(x_e) \leq \gamma_2 \frac{d(cp)}{dx}(x_m) \quad (45)$$

$$0 < x_e - x_s \leq w_{\max} \quad (46)$$

$$cp(x_e) - cp(x_s) > \Delta cp_{\min} \quad (47)$$

If such a point x_e is found, x_s is taken as the shock location. The third and fourth conditions are added with respect to [49] to avoid

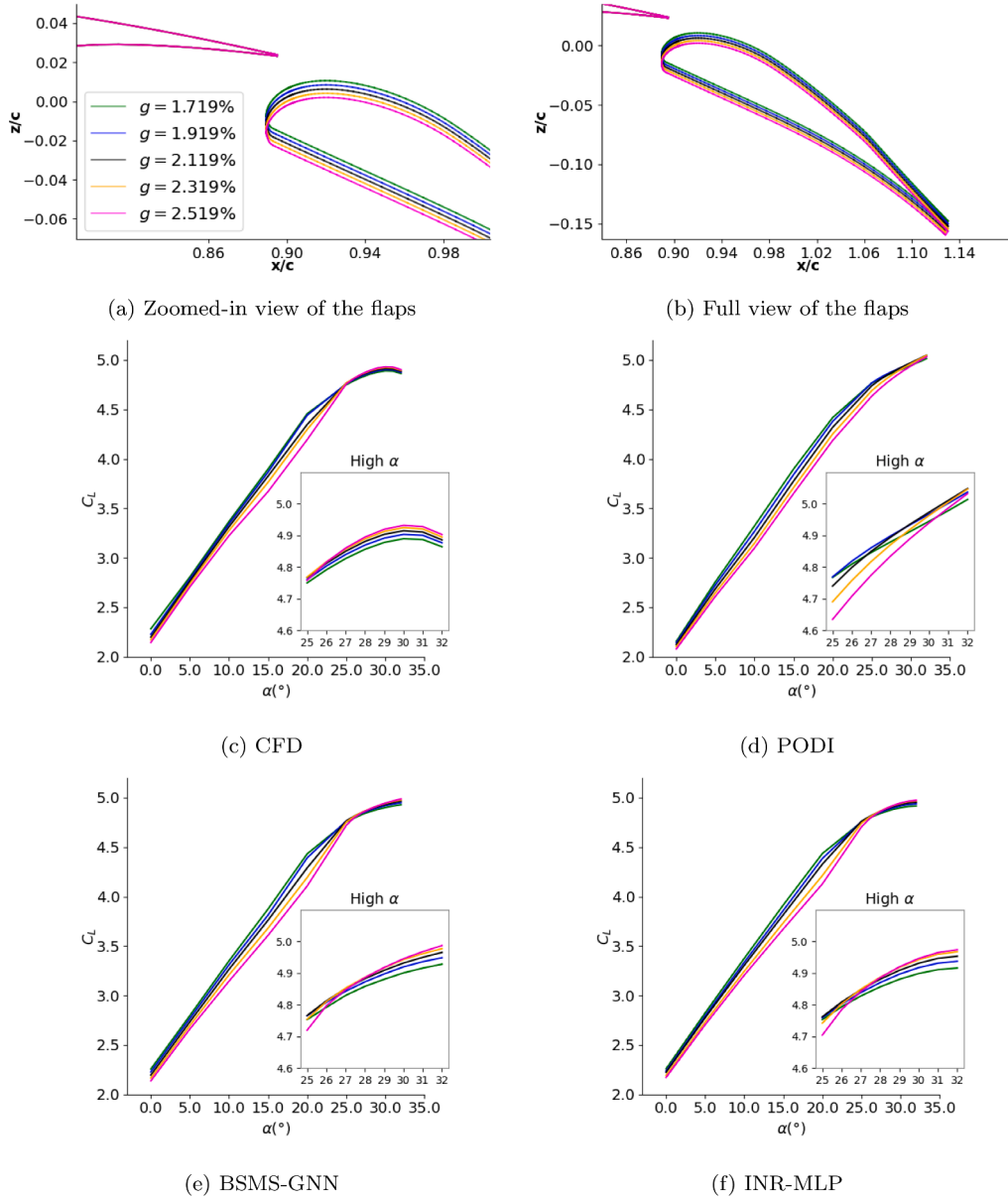


Fig. 19. Predictions of lift coefficient vs angle of attack for various gaps. The deflection angle δ and the overlap ovl correspond to the F15-3eRef landing configuration.

assigning shocks when the candidate shock location and the candidate shock end location are too far away or the variation in pressure coefficient is not significant enough for a shock to actually be present.

If a shock is found we remove from the candidates all locations that are within a distance $\delta_{\min}$ from the shock location x_s , assuming that other shocks should not be too close to the shock found. We set $C_S \leftarrow C_S \setminus [x_s - \delta_{\min}, x_s + \delta_{\min}]$. Otherwise, if no shock is found, we remove x_m from C_S , setting $C_S \leftarrow C_S \setminus \{x_m\}$, as it likely corresponds to an isolated fluctuation rather than a sustained increase in pressure coefficient. We repeat the process until either $n_{\max}$ shocks have been found, C_S is empty, or $\max_{z \in C_S} \frac{d(\text{cp})}{dx}(z) < \tau_{\min}$.

Suppose that for the reference solution the shock locations are $X^R = \{x_1^R, \dots, x_{N_r}^R\}$ and for the predicted solution correspond to $X^P = \{x_1^P, \dots, x_{N_p}^P\}$. Let $N = \max(N_r, N_p)$ and consider assignment costs c_{ij} for

assigning the predicted shock j to shock the reference shock i as given by

$$c_{ij} := \begin{cases} |x_j^P - x_i^R|, & \text{if } 1 \leq i \leq N_r, 1 \leq j \leq N_p \\ 1, & \text{otherwise} \end{cases} \quad (48)$$

We find an assignment $\pi^* \in \{0, 1\}^{N \times N}$ that solves the following minimization problem which seeks to minimize the sum of the distances between the assigned pairs of shock locations

$$\min_{\pi \in \{0, 1\}^{N \times N}} \sum_{i=1}^N \sum_{j=1}^N c_{ij} \pi_{ij} \quad (49)$$

$$\text{s.t. } \sum_j \pi_{ij} = 1, \sum_i \pi_{ij} = 1 \quad (50)$$

We denote by $j^*(i)$ the matching predicted shock to the reference shock i if $\pi_{ij} = 1$ and $1 \leq i \leq N_r$. The shock location error e_i^{sl} for the

Table 5

Error metrics by train size for the Generic Airfoils Dataset. The best model according to the MPSAWE of pressure distributions is highlighted in **bold**, while the best model based according to the MAE of the lift coefficients is underlined.

$ D_{tr} \downarrow$	Model	MPSAWE c_p	MAE C_L	M. E ^{sl}
798	PODI	0.0307	0.0191	0.81
	BSMS-GNN	0.0061 $\pm$ 0.0003	0.0050 $\pm$ 0.0003	0.04 $\pm$ 0.02
	INR-MLP	0.0098 $\pm$ 0.0003	0.0077 $\pm$ 0.0005	0.16 $\pm$ 0.06
600	PODI	0.0328	0.0203	0.83
	BSMS-GNN	0.0074 $\pm$ 0.0005	0.0058 $\pm$ 0.0005	0.05 $\pm$ 0.02
	INR-MLP	0.0110 $\pm$ 0.0004	0.0093 $\pm$ 0.0006	0.17 $\pm$ 0.05
400	PODI	0.0349	0.0231	0.81
	BSMS-GNN	0.012 $\pm$ 0.001	0.0094 $\pm$ 0.0009	0.10 $\pm$ 0.04
	<u>INR-MLP</u>	0.0130 $\pm$ 0.0005	0.0103 $\pm$ 0.0009	0.21 $\pm$ 0.06
200	PODI	0.0413	0.0289	0.78
	BSMS-GNN	0.0224 $\pm$ 0.0008	0.0205 $\pm$ 0.0007	0.18 $\pm$ 0.05
	<u>INR-MLP</u>	0.0215 $\pm$ 0.0006	0.019 $\pm$ 0.001	0.23 $\pm$ 0.07

reference shock i is computed as

$$e_i^{\text{sl}} := \begin{cases} |x_{f^*(i)}^{\text{P}} - x_i^{\text{R}}|, & \text{if a matching shock is found} \\ 1, & \text{if no matching shock is found} \end{cases} \quad (51)$$

For a sample the total shock location error E^{sl} is computed by taking the mean of the corresponding errors

$$E^{\text{sl}} := \frac{\sum_{i=1}^{N_r} e_i^{\text{sl}}}{N_r} \quad (52)$$

The specific values of the parameters for the Generic Airfoils Dataset are shown in the appendix in Table C.1.

4.3. Generic airfoils dataset

The error metrics on the test samples for the Generic Airfoils Dataset are presented in Table 5, and illustrated in Fig. 12. BSMS-GNN attains the best performance in the full-data regime with 798 training samples. It provides an improvement of 80.1% in terms of the MPSAWE with respect to PODI. INR-MLP also outperforms PODI, with an error reduction of 68.1%. In terms of the MAE of the lift coefficient the improvements of BSMS-GNN and INR-MLP correspond to 73.8% and 60.0%, respectively. For the training size of 600 samples BSMS-GNN also stands out as the best model, while for the scarcest regimes with 400 and 200 training samples both BSMS-GNN and INR-MLP achieve the best performance. It is noteworthy that in all data regimes both deep learning models outperform PODI by at least 45% in terms of the MPSAWE. In terms of the shock location error, BSMS-GNN consistently outperforms the other methods across all data regimes.

For the results that we present next, we consider only the full-data regime and for each model we select the network with the best validation MPSAWE. In Fig. 13 we plot the MPSAWE of BSMS-GNN and INR-MLP in the $M\alpha$ plane for the test samples, with each point colored according to the error value, thereby providing a spatial representation of the error distribution. This is done separately for the CST airfoils and the NURBS airfoils. It can be observed that the highest errors are obtained in the region of higher Mach numbers near the boundary of the sampling region, and in particular in combination with a high angle of attack. The errors are higher for NURBS airfoils. For instance, the MPSAWE of BSMS-GNN for NURBS airfoils corresponds to 0.0073, which is 55.3% higher than the corresponding value of 0.0047 for the CST airfoils.

We select four test samples to illustrate the predictions of the models at different flow conditions and airfoil shapes, as well as two additional challenging samples. These six samples are circled in Fig. 13. Fig. 14

shows the predictions of the pressure coefficients for these six samples. The reconstruction of the airfoils from the code obtained with INR-MLP are shown in blue overlaid on the reference airfoils in black. The reconstructed airfoils closely match the reference airfoils.

The first sample shown in Fig. 14a features a subsonic sample with a Mach number of 0.305 and an angle of attack of -2.492° . The predictions of the deep learning methods agree well with the CFD results throughout the airfoil, while near the leading edge, PODI overshoots the pressure increase at the upper surface and slightly undershoots the suction peak at the lower side. Even though PODI obtains the highest AWE (area-weighted error), it provides the best lift coefficient estimate. Fig. 14b shows the second subsonic sample with a Mach number of 0.502 and an angle of attack of 1.333° . PODI overshoots the suction pressure peak, while BSMS-GNN and INR-MLP predict it more accurately. In this case BSMS-GNN gives the best prediction of the lift coefficient, followed by PODI that still yields a more accurate prediction than INR-MLP. The third sample shown in Fig. 14c features a transonic sample with a Mach number of 0.593 and an angle of attack of 4.695° . INR-MLP gives the best prediction of the shock location, with a location $1.2\% c$ downstream, followed by BSMS-GNN that yields a shock $1.8\% c$ upstream. PODI gives an step-like unphysical behavior, does not follow the pressure distribution accurately at the upper surface and yields the worst prediction of the lift coefficient. It is assigned an error of $100\% c$ in the shock location, since no shock is predicted. The fourth sample shown in Fig. 14d features another transonic sample with a Mach number of 0.674 and an angle of attack of 2.633° . In contrast to the previous sample, BSMS-GNN gives a more accurate prediction of the shock and the pressure plateau than INR-MLP. BSMS-GNN achieves an error in the shock location of $0.5\% c$, while INR-MLP attains an error of $0.9\% c$. As in the previous case, PODI completely misses the shock and obtains the highest error in the prediction of the lift coefficient. Fig. 14e shows the first of the selected challenging samples. It features a Mach number of 0.747 and an angle of attack of 2.945° for an uncommon and aerodynamically unfavorable airfoil shape. All methods overpredict at the suction pressure peak. Both deep learning models yield the shock location too far downstream, while PODI does not yield a shock. For BSMS-GNN and INR-MLP the errors at the shock correspond to $2.3\% c$ and $5.2\% c$, respectively. The second challenging sample has a Mach number of 0.734 and an angle of attack of 3.758° and is shown in Fig. 14f. The same behavior as in the previous case is observed for the models. Even though none of the predictions closely matches the reference solution, the deep learning models still outperform PODI. BSMS-GNN outperforms in both cases PODI in terms of the MPSAWE and the MAE by more than 50%. Moreover, both deep learning methods retain specific physical features such as shocks while PODI predicts a smeared pressure distribution. For BSMS-GNN

Table 6

Error metrics by train size for the DLR-F15 Multi-Element Airfoils Dataset. The best model according to the MPSAWE of pressure distributions is highlighted in **bold**, while the best model based according to the MAE of the lift coefficients is underlined.

$ D_{tr} \downarrow$	Model	MPSAWE c_p	MAE C_L
200	PODI	0.038	0.053
	BSMS-GNN	0.023 ± 0.001	0.034 ± 0.002
	<u>INR-MLP</u>	0.021 ± 0.001	0.028 ± 0.003
150	PODI	0.041	0.057
	BSMS-GNN	0.026 ± 0.001	0.038 ± 0.002
	<u>INR-MLP</u>	0.026 ± 0.001	0.034 ± 0.002
100	PODI	0.050	0.073
	BSMS-GNN	0.037 ± 0.006	0.05 ± 0.01
	<u>INR-MLP</u>	0.036 ± 0.002	0.045 ± 0.003
50	PODI	0.059	0.082
	BSMS-GNN	0.07 ± 0.01	0.11 ± 0.02
	INR-MLP	0.065 ± 0.004	0.093 ± 0.007

and INR-MLP the errors at the shock correspond to 1.7% c and 5.8% c , respectively.

4.4. DLR-F15 multi-element airfoils dataset

For the DLR-F15 Multi-Element Airfoils Dataset we present first the results on the test dataset. Then, we present additional results on airfoils that differ only by the gap to examine the effect of the gap size.

4.4.1. Performance on the test dataset

The error metrics on the test samples of the DLR-F15 Multi-Element Airfoils Dataset are presented in Table 6, and illustrated in Fig. 15. INR-MLP attains the best performance in the full-data regime with 200 samples. It provides an improvement of 44.7% in terms of the MPSAWE with respect to PODI. BSMS-GNN also outperforms PODI, with an error reduction of 39.5%. In terms of the MAE of the lift coefficient the improvements of INR-MLP and BSMS-GNN correspond to 47.2% and 35.8%, respectively. For the training size of 150 samples INR-MLP and BSMS-GNN are the best models in terms of the MPSAWE, while INR-MLP is the best model in terms of the MAE of the lift coefficient. For the regime with 100 training samples both BSMS-GNN and INR-MLP achieve the best performance. For the scarcest regime with only 50 samples PODI achieves the best performance. The amount of data could be insufficient for the deep learning models to operate and generalize properly to airfoils that become more different than the ones seen during training, without the explicit parameterization that only PODI is allowed to use. Moreover, the hyperparameters of the deep learning models were optimized using the full-data regime, which can result in models with many hyperparameters leading to increased overfitting in scarcer regimes.

We select four test samples to illustrate the predictions of the models at different flow conditions and flap positions, as well as two of the most challenging samples. These six samples are circled in Fig. 16. Fig. 17 shows the predictions of the pressure coefficients for the first four samples.

The first sample shown in Fig. 17a features a Mach number of 0.115 and an angle of attack of 29.913°. At the slat, PODI overpredicts the suction pressure peak. At the main element, INR-MLP underpredicts at the lower surface at the corner of the sealing lip. At the flap, PODI also overpredicts the suction pressure peak, while BSMS-GNN and INR-MLP give more accurate predictions as shown in the zoomed-in view. Overall, PODI overpredicts the lift coefficient and attains the highest absolute error. However, compared to the results of the Generic Airfoils Dataset all methods follow significantly closer the reference in terms of the physical phenomena they capture and relative to the range of the respective pressure coefficients. This also holds for most of the samples of the dataset

and can be attributed to the fact that subsonic flows are investigated and no shocks occur. This results in more suitable predictions for engineering applications, despite the magnitude of the errors being higher. The second sample shown in Fig. 17b has a Mach number of 0.110 and an angle of attack of 1.916°. At the slat, PODI overpredicts the pressure peak at the leading edge. BSMS-GNN overpredicts at the upper surface towards the trailing edge. INR-MLP overpredicts at the lower surface near the leading edge. At the main element, BSMS-GNN overpredicts near the leading edge. INR-MLP faces the same issue at the sealing lip as described in the previous case. PODI overpredicts at the lower surface near the trailing edge. At the flap, PODI overpredicts the suction pressure peak and then directly underpredicts, while BSMS-GNN gives the most accurate prediction. Fig. 17c shows the third sample with a Mach number of 0.169 and an angle of attack of 11.645°. At the slat, all predictions follow closely the CFD data. At the main element, INR-MLP underpredicts at the corner of the sealing lip. At the flap, PODI underpredicts the first suction pressure peak and overpredicts the second one, while BSMS-GNN overpredicts both. INR-MLP gives a more accurate prediction. The fourth sample shown in Fig. 17d has a Mach number of 0.248 and an angle of attack of 11.439°. At the slat, PODI overpredicts at the suction pressure peak and underpredicts at the lower surface near the leading edge. At the main element, PODI underpredicts at the lower surface near the trailing edge and INR-MLP underpredicts at the corner of the sealing lip. At the flap, PODI underpredicts the suction pressure peak and then does not capture correctly the pressure plateau, indicative of flow separation. The deep learning models provide more accurate predictions in this case.

Fig. 17e shows the first of the selected challenging samples. It features a Mach number of 0.120 and an angle of attack of 22.183°. The flap has an overlap of 7.860% and an angle of 32.183° resulting in an untypical configuration, not beneficial from an aerodynamics standpoint, that is near the boundary of the sampling space in the δ – owl plane. At the three components, all models overpredict at the upper surface. At the main element, the models overpredict the pressure decrease at the lower surface near the trailing edge. Overall, the models overpredict the lift coefficient. The second challenging sample has a Mach number of 0.166 and an angle of attack of 25.478°. The flap has an overlap of –1.396% and an angle of 32.542°. At the slat and main element, all models underpredict at the upper surface. At the flap, they also underpredict at the upper surface, except near the trailing edge, where they overpredict as shown in the zoomed-in view. Overall, the models underpredict the lift coefficient.

4.5. Effect of area-weighted loss on the mean per-sample area-weighted error

To evaluate the effect of using an area-weighted loss function instead of an unweighted loss function, we conducted an ablation study isolating its effect on the mean per-sample area-weighted error (MPSAWE) on the test samples. For both deep learning models, BSMS-GNN and INR-MLP, we compare the errors attained using the area-weighted mean squared error loss $\mathcal{L}_{AWMSE}$ and the (unweighted) mean squared error loss $\mathcal{L}_{MSE}$. We performed the experiments using the full-data regime and trained 16 networks for each loss function and report the average of the errors together with their standard deviations. For each run we selected the weights of the model from the epoch with best MPSAWE on the validation samples. Table 7 shows the MPSAWE of the pressure coefficients and the mean absolute error of the lift coefficients for the test samples. For BSMS-GNN, the MPSAWE is the same for both loss functions, while for INR-MLP there is a 12.5% improvement using the area-weighted loss function. INR-MLP treats each node independently, so that the error at each location is only directly affected by the prediction at that location without direct influence of how the network processes other nodes. Therefore, it is likely, that area weighting helps to achieve less error in coarse large areas by emphasizing the gradients in such regions, that would otherwise be under-represented. In contrast,

Table 7

Error metrics under different loss functions in the full-data regime for the DLR-F15 Multi-Element Airfoils Dataset.

Model	Loss	MPSAWE c_p	MAE C_L
BSMS-GNN	$\mathcal{L}_{\text{AWMSE}}$	0.023 ± 0.001	0.034 ± 0.002
	$\mathcal{L}_{\text{MSE}}$	0.023 ± 0.001	0.033 ± 0.002
INR-MLP	$\mathcal{L}_{\text{AWMSE}}$	0.021 ± 0.001	0.028 ± 0.003
	$\mathcal{L}_{\text{MSE}}$	0.024 ± 0.001	0.030 ± 0.002

for BSMS-GNN, the prediction at each node is directly influenced by the processed representations of all other nodes, so that an accurate prediction at one node likely relies on accurate representations of all other nodes, potentially making area-weighting less impactful. Fig. 18 shows the mean MPSAWE and its standard deviation at each epoch for the first 500 epochs. It shows for both models that the area-weighted loss function leads to a faster improvement in the validation MPSAWE during the first epochs. For 3D cases with millions of nodes and larger variations in mesh-density, this can potentially lead to significant time savings.

4.5.1. Effect of the gap size on the lift coefficient

To present a more comprehensive analysis, an additional $g\alpha$ -dataset is created by varying only the gap and angle of attack while keeping the remaining parameters constant. All simulations are performed at a Mach number of $M = 0.120$. The flap parameters of the F15-3eRef landing configuration are used as the starting point. The dataset is created by changing the gap of the flap in steps of 0.2% to obtain a set of gaps G and performing simulations for the fixed set of angles of attack Θ

$$G := \{1.719\%, 1.919\%, 2.119\%, 2.319\%, 2.519\%\} \quad (53)$$

$$\theta := \{0^\circ, 5^\circ, 10^\circ, 15^\circ, 20^\circ, 25^\circ, 26^\circ, 27^\circ, 28^\circ, 29^\circ, 30^\circ, 31^\circ, 32^\circ\} \quad (54)$$

Fig. 19 shows the lift coefficient vs angle of attack curves for CFD and the data-driven models. The position of the flaps is illustrated at the top of the figure. As seen in Fig. 19c the lift coefficient decreases with increasing gap for angles of attack below 20° . At angles of attack above 25° the trend reverses, the lift coefficient increases with increasing gap. The reduction in the lift coefficient at lower angles of attack is likely attributable to earlier flow separation caused by the larger flap gap, whereas at higher angles when flow reattachment occurs the lift reduction is mitigated or reversed. All models are able to predict the correct trend at lower angles of attack. However, PODI struggles at higher angles of attack, while BSMS-GNN and INR-MLP are able to reproduce the trend at higher angles. For these samples PODI obtains a MAE of 0.061, while BSMS-GNN and INR-MLP attain values of 0.028 and 0.020, respectively. Also, in terms of the Spearman rank correlation coefficient [50] that measures how strongly two sets of ranks are correlated, PODI with 0.941, performs worse than BSMS-GNN and INR-MLP that achieve coefficients of 0.972 and 0.977, respectively.

We also observe that the CFD lift curves enter a more nonlinear region above 25° , reach a maximum lift coefficient around 30° and then decrease at higher angles. None of the models is able to match this trend as seen in the zoomed-in views for higher angles, presumably due to the fact that these samples are outside of the sampling domain in the Ma plane in a nonlinear region.

4.6. Computational cost

DLR's High Performance Computing System CARA was used to perform simulations with TAU for the Generic Airfoils Dataset while the DLR's large-scale research facility Computer for Advanced Research in Aerospace (CARO) was used for performing numerical simulations with CODA for the DLR-F15 Multi-Element Airfoils Dataset. DLR's High Performance Computing System CARA was used to train all deep learning models with an NVIDIA A100 80GB GPU. Tables 8 and 9 show the training time and inference time for each model in the full-data regime. The

Table 8

Computational cost for the generic airfoils dataset in the full-data regime. Mesh interpolation, creation of graphs and SDF fields is not included in the times.

Model	Training Time (h)	Inference Time (ms)
CFD	—	720000
PODI	2×10^{-4}	2
BSMS-GNN	3.2	10
INR-MLP (INR + MLP)	9.9 (9.4 + 0.5)	204 (200 + 4)

Table 9

Computational cost for DLR-F15 multi-element airfoils dataset in the full-data regime. Mesh interpolation, creation of graphs and SDF fields is not included in the times.

Model	Training Time(h)	Inference Time (ms)
CFD	—	59100
PODI	3×10^{-4}	2
BSMS-GNN	1.5	20
INR-MLP (INR + MLP)	3.1 (2.6 + 0.5)	305 (300 + 5)

reported times do not include mesh interpolation, as well as creation of graphs and SDF fields, as these are considered pre-processing and post-processing steps whose necessity and computational cost can vary widely across deployment scenarios.

Table 8 shows the computational cost for the Generic Airfoils Dataset. PODI takes only around one second to train, while BSMS-GNN and INR-MLP take 3.2 hours and 9.9 hours, respectively. The inference time for all models is below one second and is thus negligible with respect to the 2 hours required by the CFD solver.

Table 9 shows the computational cost for the DLR-15 Multi-Element Airfoil Dataset. PODI takes only around one second to train, while BSMS-GNN and INR-MLP take 1.5 hours and 3.1 hours, respectively. The average inference time for the CFD solver is around 10 minutes. The inference time for all models is below one second is also negligible with respect to the 10 minutes required by the CFD solver.

For both datasets, all surrogate models offer substantial speed-ups over high-fidelity simulations during inference, but their relative advantage depend on the specific accuracy-efficiency trade-off relevant to the target application. Considering both the cost of generating the dataset and the training time, the selection and applicability of the surrogate model should be guided by the specific constraints and objectives of the intended application.

5. Conclusions

In this paper, we investigated the use of geometric deep learning methods for the prediction of surface pressure distributions of non-parametric airfoils under varying levels of data availability. For this investigation we created two tailored high-fidelity datasets solving the Reynolds-Averaged Navier Stokes equations. The first dataset, referred to as Generic Airfoils Dataset, features pressure coefficient distributions in the subsonic and transonic regime for single-element airfoils, with the angle of attack, Mach number and Reynolds number as variable global flow parameters. The second dataset, referred to as DLR-F15 Multi-Element Airfoils Dataset, features pressure coefficient distributions in the subsonic regime for multi-element airfoils with a fixed slat and main element but a variable flap position, with the angle of attack and Mach number as variable global flow parameters. Two geometric deep learning approaches were adapted and employed, Bi-Stride Multi-Scale Graph Neural Network (BSMS-GNN) and Implicit Neural Representation of the signed distance function coupled with a Multi-Layer Perceptron (INR-MLP). To enhance both of these methods, we used area-weighted loss functions to better account for variations in node density in the meshes. Moreover, in the formulation of BSMS-GNN we introduced edge completion at the coarsest level to account for interactions between

different connected components, such as flap, main element and slat in a 3-element high-lift airfoil. These models were compared to Proper Orthogonal Decomposition coupled with Interpolation (PODI), which was allowed to use an explicit parameterization of the airfoils.

The results show that both geometric deep learning methods are able to consistently outperform the classical PODI across various experiments done for both datasets, even in the absence of an explicit geometric parameterization. Moreover, the use of an area-weighted loss function enhanced area-weighted performance and lead to faster performance gains in the early training epochs for the deep learning methods. PODI performed best only for the scarce-data regime of the DLR-F15 Multi-Element Airfoils Dataset. It is likely that the geometric deep learning methods are more susceptible to performance degradation due to out-of-distribution airfoils compared to PODI, which is allowed to use the geometric parameterization. Both geometric deep learning methods capture non-linear behavior better than PODI. They match shocks more accurately as well as flow separation. BSMS-GNN matches shocks more accurately based on the shock location error. For the Generic Airfoils Dataset higher errors were found at the transonic region with higher angles of attack. For the DLR-F15 Multi-Element Dataset higher errors were found corresponding to flap positions towards the boundary of the domain, such as those featuring a large overlap and a relatively high deflection angle. For the DLR-F15 Multi-Element Airfoils Dataset all methods capture the trend of the effect of the gap on the lift coefficient with respect to the landing configuration at lower angles of attack. The geometric deep learning methods capture better the trend at higher angles of attack when flow detaches and reattaches. All models are able to give predictions in nearly real-time. However, training the geometric deep learning methods requires hours with BSMS-GNN being slightly faster than the INR-MLP, while PODI can be trained also nearly real-time. For INR-MLP most of the training time corresponded to training the network that learns to generate the codes for the airfoils.

Overall, both geometric deep learning methods are suitable approaches as surrogates for the prediction of surface pressure distributions of non-parametric airfoils, either single-element or multi-element. In this paper, we dealt with 2D airfoils so that applying and scaling the methodologies for 3D aircraft remains for future work. The main challenges are the larger size of the meshes and the presence of more components, as well as the limited data availability in industrial cases. In the case of 3D high lift cases mid and long-range interactions between the slat, main element and flap need to be accounted for when scaling the graph approaches, ensuring communication between elements. The use of random downsampling to select anchor or regional nodes as proposed recently can be explored to enhance the scalability of the models as well as the robustness and generalizability of such models. BSMS-GNN can require the partition of the domain as in XMeshGraphNets [29]. The use of the implicit neural representation of the signed distance function might also require partitioning of the domain, or a targeted selection of query nodes around the region where the slat and flap change. Future work may also explore shock-aware loss weighting strategies to better capture discontinuities and improve performance in transonic flows.

CRedit authorship contribution statement

Derrick Hines: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization; **Philipp Bekemeyer:** Writing – review & editing, Supervision, Software, Resources, Project administration, Funding acquisition, Data curation, Conceptualization.

Data availability

Data will be made available on request.

Declaration of competing interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The authors gratefully acknowledge the scientific support and HPC resources provided by the German Aerospace Center (DLR). The HPC system CARA is partially funded by ‘Saxon State Ministry for Economic Affairs, Labor and Transport’ and ‘Federal Ministry for Economic Affairs and Climate Action’. The HPC system CARO is partially funded by “Ministry of Science and Culture of Lower Saxony” and “Federal Ministry for Economic Affairs and Climate Action”.

CODA is the computational fluid dynamics (CFD) software being developed as part of a collaboration between the French Aerospace Lab ONERA, the German Aerospace Center (DLR), Airbus, and their European research partners. CODA is jointly owned by ONERA, DLR and Airbus.

Appendix A. Model hyperparameters

Table A.1
PODI hyperparameters.

Hyperparameter	CST	NURBS	DLR-F15
Relative information content	100.0%	100.0%	100.0%
Interpolator	TPS	TPS	TPS
Augmentation	Linear	Linear	Quadratic
Scale	True	True	True
Interpolate to reference	False	False	True

Table A.2
BSMS-GNN hyperparameters.

Hyperparameter	Generic Airfoils	DLR-F15
Activation	ELU	ELU
MLP’s hidden dimension	128	128
Latent dimension	64	64
MLP _E input dimension	7	6
MLP _E hidden layers	2	2
Graph coarsening levels	4	8
Edge completion at coarsest level	True	True
MLP _M hidden layers	1	1
MLP _U hidden layers	1	1
MLP _D hidden layers	2	2
Weights for downsampling	Uniform	Uniform
Weights for loss	Area weights	Area weights
Epochs	10000	10000
Batch size	20	20
Learning Rate Scheduler	Exponential	Exponential
Initial LR	8.4×10^{-4}	6.3×10^{-4}
Final LR	5.0×10^{-6}	5.0×10^{-6}
Learnable parameters	275,969	476,033

Table A.3
INR hyperparameters.

Hyperparameter	Generic Airfoils	DLR-F15
Activation	ReLU	ReLU
Input dimension	2	2
Code dimension	32	16
Fourier encoding frequencies F	64	64
Fourier encoding standard deviation	1.0	1.0
Hidden dimension	128	128
Hidden layers	5	5
Epochs	7500	75000
Batch size	32	32
Sampled points	5000	5000
Outer learning rate	1.0×10^{-4}	1.0×10^{-4}
Inner steps	3	3
Inner learning rate	1.0×10^{-2}	1.0×10^{-2}
Learnable parameters	120,577	110,337
Sampled uniform points	5000	5000
Sampled points near surface	30,720	49,860

Table A.4
MLP hyperparameters.

Hyperparameter	Generic Airfoils	DLR-F15
Activation	ELU	ELU
Input dimension	39	22
Hidden layers	10	11
Hidden dimension	128	128
Residual frequency	2	3
Weights for loss	Area weights	Area weights
Epochs	10000	10000
Batch size	50	10
Learning Rate Scheduler	Exponential	Exponential
Initial LR	9.0×10^{-4}	5.7×10^{-4}
Final LR	5.0×10^{-6}	5.0×10^{-6}
Learnable parameters	208,513	204,161

Appendix B. Supplementary information for INR-MLP

This section include supplementary information for INR-MLP. Visualizations are provided for both datasets in the next subsections.

B.1. Generic airfoils dataset

Fig. B.1 shows a two-dimensional t-SNE representation of the latent representations learned for the training samples of the Generic Airfoils Dataset. Even though the airfoils were derived from two different parameterizations, the implicit neural representation seems to organize them in a latent space in which they appear to be organized consistently under a single parameterization.

B.2. DLR-F15 multi-element airfoils dataset

Fig. B.2 shows a two-dimensional t-SNE representation of the latent representations learned for the training samples of the DLR-F15 Multi-Element Airfoils Dataset. The samples are colored based on the flap deflection angle δ .

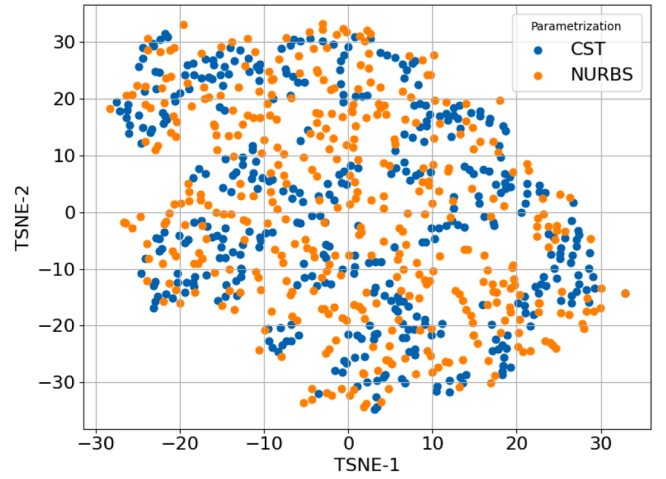


Fig. B.1. t-SNE representation of the 32-dimensional latent representations of the training samples of the Generic Airfoils Dataset.

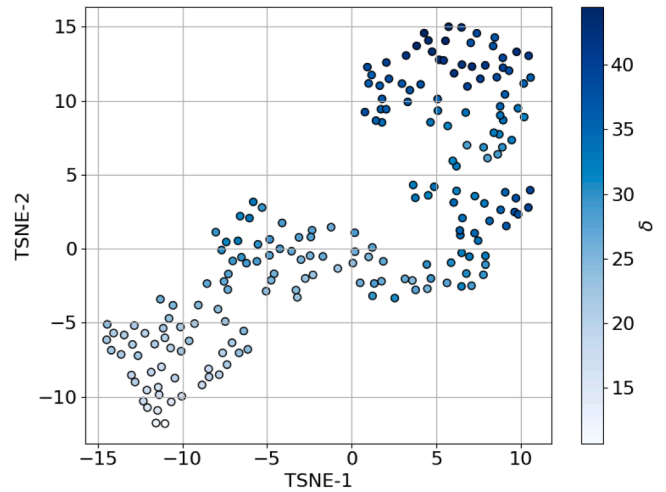


Fig. B.2. t-SNE representation of the 16-dimensional latent of the training samples of the DLR-F15 Multi-Element Airfoils Dataset. Samples are colored based on their flap deflection angle δ .

flexion angle δ . Samples with low angles and samples with high angles appear to occupy distinct regions of the plot.

Fig. B.3 shows the signed distance field associated to a test airfoil, as well as the decoded field extracted from the latent code. It also shows the aerodynamic parameterization and the latent code.

Fig. B.4 shows the norm of the gradient. As expected, the norm of the gradient is higher in the region closer to the flap, since only the position of the flap changes in the dataset.

Fig. B.5 shows the partial derivatives of the predicted signed distance function with respect to the code components, illustrating how sensitive the prediction is to changes at each code component.

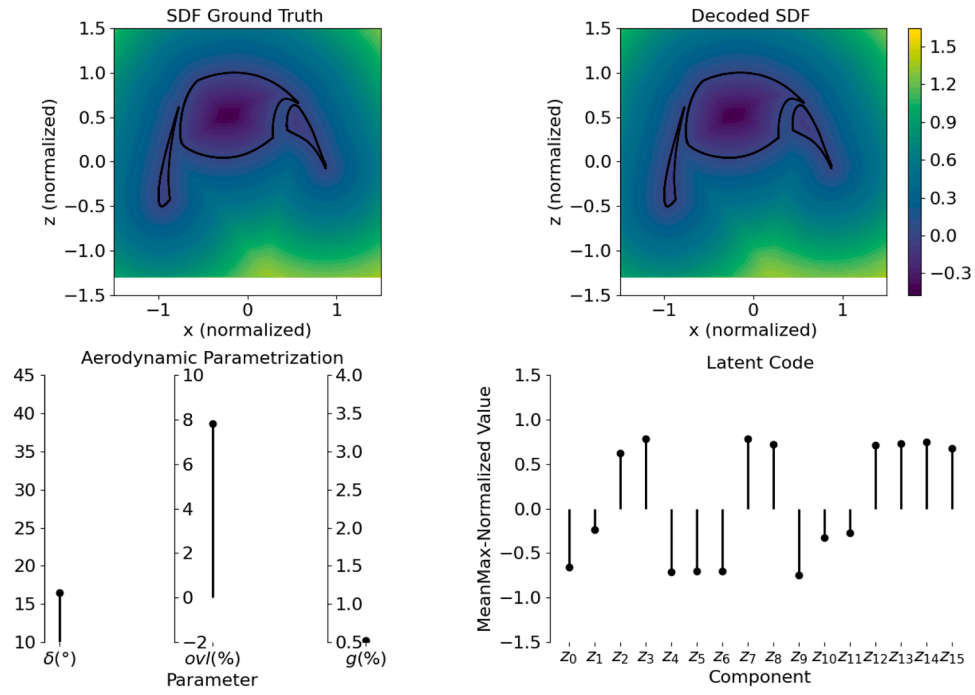


Fig. B.3. Comparison of signed distance function and decoded signed distance function for a test airfoil of the DLR-F15 Multi-Element Airfoils Dataset.

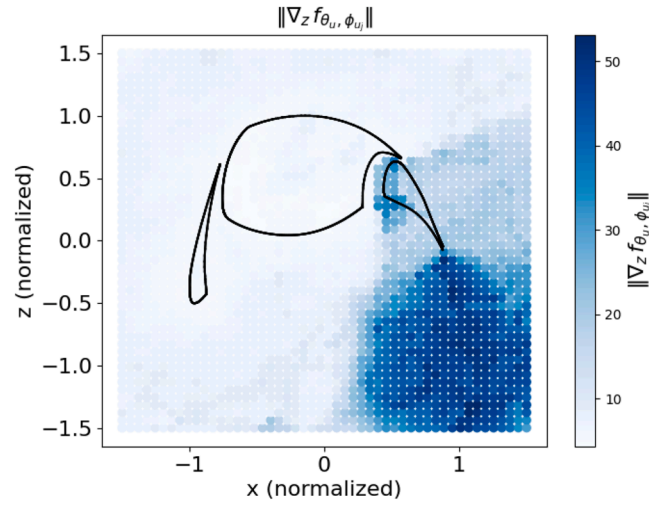


Fig. B.4. Representation of the norm of the gradient of the predicted signed distance function with respect to the code DLR-F15 Multi-Element Airfoils Dataset.

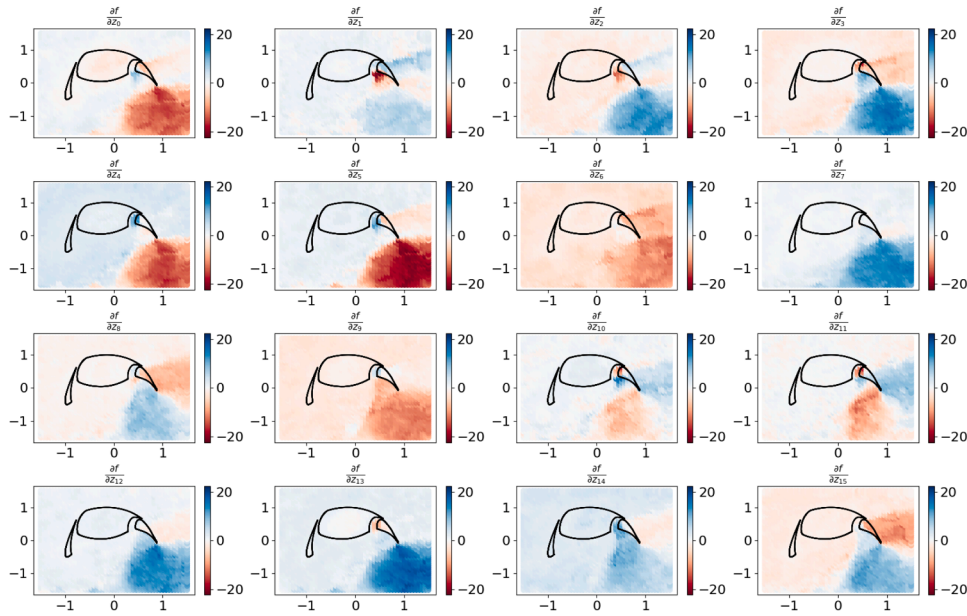


Fig. B.5. Representation of the partial derivatives of the predicted signed distance function with respect the code components DLR-F15 Multi-Element Airfoils Dataset.

Appendix C. Shock location error parameters

Table C.1
Shock location error parameters.

Parameter	Value
γ_1	0.2
γ_2	0.3
$w_{\max}$	0.06
$\Delta c_{p_{\min}}$	0.4
$\delta_{\min}$	0.2
$\tau_{\min}$	3
$n_{\max}$	2
Derivatives	Downwind

References

- [1] Bui-Thanh T, Damodaran M, Willcox K. Proper orthogonal decomposition extensions for parametric applications in compressible aerodynamics. In: 21st AIAA applied aerodynamics conference. 2003, p. 4213.
- [2] Iuliano E, Quagliarella D. Proper orthogonal decomposition, surrogate modelling and evolutionary optimization in aerodynamic design. *Comput Fluids* 2013;84:327–50.
- [3] Franz T, Zimmermann R, Görtz S, Karcher N. Interpolation-based reduced-order modelling for steady transonic flows via manifold learning. *Int J Comput Fluid Dyn* 2014;28. <https://doi.org/10.1080/10618562.2014.918695>
- [4] Fossati M. Evaluation of aerodynamic loads via reduced-order methodology. *AIAA J* 2015;53(8):2389–405.
- [5] Ripepi M, Verveld MJ, Karcher NW, Franz T, Abu-Zurayk M, Görtz S, et al. Reduced-order models for aerodynamic applications, loads and MDO. *CEAS Aeronaut J* 2018;9(1):171–93.
- [6] Forrester A, Keane A. Recent advances in surrogate-based optimization. *Prog Aerosp Sci* 2009;45:50–79. <https://doi.org/10.1016/j.paerosci.2008.11.001>
- [7] Sabater C, Stürmer P, Bekemeyer P. Fast predictions of aircraft aerodynamics using deep-learning techniques. *AIAA J* 2022;60(9):5249–61. <https://doi.org/10.2514/1.61234>
- [8] Guo X, Li W, Iorio F. Convolutional neural networks for steady flow approximation. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. KDD '16; New York, NY, USA: Association for Computing Machinery. ISBN 9781450342322; 2016, p. 481–90. <https://doi.org/10.1145/2939672.2939738>
- [9] Jin X, Cheng P, Chen W-L, Li H. Prediction model of velocity field around circular cylinder over various reynolds numbers by fusion convolutional neural networks based on pressure on the cylinder. *Phys Fluids* 2018;30(4):047105. <https://doi.org/10.1063/1.5024595>
- [10] Bhatnagar S, Afshar Y, Pan S, Duraisamy K, Kaushik S. Prediction of aerodynamic flow fields using convolutional neural networks. *Comput Mech* 2019;64:525–45. <https://doi.org/10.1007/s00466-019-01740-0>
- [11] Thuerey N, Weißenow K, Prantl L, Hu X. Deep learning methods for reynolds-averaged navier-stokes simulations of airfoil flows. *AIAA J* 2020;58(1):25–36. <https://doi.org/10.2514/1.61058291>
- [12] Wu H, Liu X, An W, Chen S, Lyu H. A deep learning approach for efficiently and accurately evaluating the flow field of supercritical airfoils. *Comput Fluids* 2020;198:104393. <https://doi.org/10.1016/j.compfluid.2019.104393>
- [13] Greenman R, Roth K. In: ASME 1998 Computers in engineering conference. American Society of Mechanical Engineering; 1998, https://archive.org/details/NASA_NTRS_Archive_20020060751.
- [14] Wallach R, Mattos B, Girardi R, Curvo M. Aerodynamic coefficient prediction of transport aircraft using neural network. In: 44th AIAA aerospace sciences meeting and exhibit. American Institute of Aeronautics and Astronautics; 2006. <https://doi.org/10.2514/6.2006-658>
- [15] Park JJ, Florence P, Straub J, Newcombe R, Lovegrove S. DeepSDF: learning continuous signed distance functions for shape representation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR). 2019.
- [16] Serrano L, Boudec LL, Koupai AK, Wang TX, Yin Y, Vittaut J-N, et al. Operator learning with neural fields: tackling PDEs on general geometries. In: Thirty-seventh conference on neural information processing systems. 2023, <https://openreview.net/forum?id=4JEq5nhg1>.
- [17] Serrano L, Migus L, Yin Y, Mazari JA, Gallinari P. Infinity: neural field modeling for reynolds-averaged navier-stokes equations. 2023. <https://arxiv.org/abs/2307.13538>
- [18] Catalani G, Agarwal S, Bertrand X, Tost F, Bauerheim M, Morlier J. Neural fields for rapid aircraft aerodynamics simulations. *Sci Rep* 2024;14(1). <https://doi.org/10.1038/s41598-024-76983-w>
- [19] Bronstein MM, Bruna J, LeCun Y, Szlam A, Vandergheynst P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Process Mag* 2017;34(4):18–42. <https://doi.org/10.1109/MSP.2017.2693418>
- [20] Sanchez-Gonzalez A, Godwin J, Pfaff T, Ying R, Leskovec J, Battaglia PW. Learning to simulate complex physics with graph networks. In: International conference on machine learning. 2020.
- [21] Pfaff T, Fortunato M, Sanchez-Gonzalez A, Battaglia PW. Learning mesh-based simulation with graph networks. In: International conference on learning representations. 2021.
- [22] Harsch L, Riedelbauch S. Direct prediction of steady-state flow fields in meshed domain with graph networks. 2021. <https://doi.org/10.48550/ARXIV.2105.02575>
- [23] Hines D, Bekemeyer P. Graph neural networks for the prediction of aircraft surface pressure distributions. *Aerosp Sci Technol* 2023;137:108268. <https://doi.org/10.1016/j.ast.2023.108268>
- [24] Cao Y, Chai M, Li M, Jiang C. Efficient learning of mesh-based physical simulation with BSMS-GNN. 2023. <https://arxiv.org/abs/2210.02573>
- [25] Lino M, Fotiadis S, Bharath AA, Cantwell C. Towards fast simulation of environmental fluid mechanics with multi-scale graph neural networks. 2022. <https://arxiv.org/abs/2205.02637>.

- [26] Li Z, Kovachki N, Azizzadenesheli K, Liu B, Bhattacharya K, Stuart A, et al. Multipole graph neural operator for parametric partial differential equations. In: 34th conference on neural information processing systems (neurIPS 2020). 2020.
- [27] Liu W, Yagoubi M, Schoenauer M. Multi-resolution graph neural networks for PDE approximation. In: Farkaš I, Masulli P, Otte S, Wermter S, editors. Artificial neural networks and machine learning – ICANN 2021. Cham: Springer International Publishing. ISBN 978-3-030-86365-4; 2021, p. 151–63.
- [28] Fortunato M, Pfaff T, Wirnsberger P, Pritzel A, Battaglia P. Multiscale meshgraphnets 2022; <https://doi.org/10.48550/ARXIV.2210.00612>
- [29] Nabian MA, Liu C, Ranade R, Choudhry S. X-meshgraphnet: scalable multi-scale graph neural networks for physics simulation. 2024. <https://arxiv.org/abs/2411.17164>.
- [30] Ashton N, Mockett C, Fuchs M, Fliessbach L, Hetmann H, Knacke T, et al. DrivaerML: high-fidelity computational fluid dynamics dataset for road-car external aerodynamics. 2025. <https://arxiv.org/abs/2408.11969>.
- [31] Mousavi S, Wen S, Lingsch L, Herde M, Raonić B, Mishra S. Rigno: a graph-based framework for robust and accurate operator learning for pdes on arbitrary domains. 2025. <https://arxiv.org/abs/2501.19205>.
- [32] Massegur Sampietro D, Da Ronch A. Augmenting hierarchical graph convolutional networks for aerodynamic prediction of shape variations. In: AIAA Aviation Forum and Ascend 2025. American Institute of Aeronautics and Astronautics; 2025, <https://doi.org/10.2514/6.2025-3230>
- [33] Alkin B, Bleeker M, Kurl R, Kronlachner T, Sonnleitner R, Dorfer M, et al. Ab-upt: scaling neural cfd surrogates for high-fidelity automotive aerodynamics simulations via anchored-branched universal physics transformers. 2025. <https://arxiv.org/abs/2502.09692>.
- [34] Wang S, Zhang X, Tie Y, Xia W, Jiang Y. Aerodynamic performance prediction of 3d aircraft based on point cloud deep learning. J Comput Des Eng 2025;12(5):175–89. <https://doi.org/10.1093/jcde/qwaf051>
- [35] Zhang H, Shen Y, Huang W, Xie Z, Niu Y-B. Deep transfer learning for three-dimensional aerodynamic pressure prediction under data scarcity. Theor Appl Mech Lett 2025;15(2):100571. <https://doi.org/10.1016/j.taml.2025.100571>
- [36] Shen Y, Huang W, Wang Z-g, Xu D-f, Liu C-Y. A deep learning framework for aerodynamic pressure prediction on general three-dimensional configurations. Phys Fluids 2023;35(10). <https://doi.org/10.1063/5.0172437>
- [37] Bekemeyer P, Hariharan N, Wissink AM, Cornelius J. Introduction of applied aerodynamics surrogate modeling benchmark cases. In: AIAA SCITECH 2025 Forum. American Institute of Aeronautics and Astronautics; 2025, <https://doi.org/10.2514/6.2025-0036>
- [38] Peter J, Bennehard Q, Heib S, Hantrais-Gervois J-L, Moens F. Onera's crm wbpn database for machine learning activities, related regression challenge and first results. 2025. <https://arxiv.org/abs/2505.06265>.
- [39] Cloud L. Shift-wing: high-fidelity computational fluid dynamics dataset for transonic aerospace external aerodynamics. 2025. <https://huggingface.co/datasets/luminary-shift/Wing>.
- [40] Bonnet F, Mazari AJ, Cinnella P, Gallinari P. AirfRANS: high fidelity computational fluid dynamics dataset for approximating reynolds-averaged navier-stokes solutions. 2023. <https://arxiv.org/abs/2212.07564>.
- [41] Cornelius J, Peters N, Aagren TS, Nieves Lugo D. Palmo: an overflow machine learning airfoil performance database. In: AIAA SCITECH 2025 Forum. American Institute of Aeronautics and Astronautics; 2025, <https://doi.org/10.2514/6.2025-0038>
- [42] Kanchi RS, Melanson B, Somasekharan N, Pan S, He S. Unifoil: a universal dataset of airfoils in transitional and turbulent regimes for subsonic and transonic flows. 2025. <https://arxiv.org/abs/2505.21124>.
- [43] Zhang C, Hu Z, Shi Y, Xu G. Fast aerodynamic prediction of airfoil with trailing edge flap based on multi-task deep learning. Aerospace 2024;11(5). <https://doi.org/10.3390/aerospace11050377>
- [44] Wild J. Mach and reynolds number dependencies of the stall behavior of high-lift wing-sections. J Aircr 2013;50(4):1202–16. <https://elib.dlr.de/87409/>.
- [45] Zintgraf L, Shiarli K, Kurin V, Hofmann K, Whiteson S. Fast context adaptation via meta-learning. In: Chaudhuri K, Salakhutdinov R, editors. Proceedings of the 36th international conference on machine learning; vol. 97 of *Proceedings of Machine Learning Research*. PMLR; 2019, p. 7693–702. <https://proceedings.mlr.press/v97/zintgraf19a.html>.
- [46] Bekemeyer P, Bertram A, Chaves D AH, Ribeiro MD, Garbo A, Kiener A, et al. Data-driven aerodynamic modeling using the DLR smarty toolbox. In: AIAA Aviation 2022 forum. 2022.
- [47] Wild J. High-Lift Aerodynamics. Taylor & Francis Group; 2022. <https://elib.dlr.de/148685/>.
- [48] Brodersen O, Hepperle M, Ronzheimer A, Rossow C-C, Schöning B. The parametric grid generation system megacads. In: Proc. 5ath international conference on numerical grid generation in computational field simulations, 1.-5.4.1996, MS. NSF Engineering Research Center, Mississippi State University; 1996, p. 353–62. institution = Soni, B.K.; Thompson, J.F.; Häuser, J.; Eiseman, P.R. (Ed.), LIDO-Berichtsjahr = 1996; <https://elib.dlr.de/36487/>.
- [49] Mufti B, Bhaduri A, Ghosh S, Wang L, Mavris DN. Shock wave prediction in transonic flow fields using domain-informed probabilistic deep learning. Physics of Fluids 2024;36(1):016121. <https://doi.org/10.1063/5.0185370>
- [50] Spearman C. The proof and measurement of association between two things. Am J Psychol 1904;15(1):72. <https://doi.org/10.2307/1412159>