Graph Convolutional Multi-Mesh Autoencoder for Steady Transonic Aircraft Aerodynamics

David Massegur‡ and Andrea Da Ronch§

Faculty of Engineering and Physical Sciences

University of Southampton, Southampton, United Kingdom, SO16 7QF

Abstract. Calculating aerodynamic loads around an aircraft using computational fluid dynamics is a user's and computer-intensive task. An attractive alternative is to leverage neural networks bypassing the need of solving the governing fluid equations at all flight conditions of interest. Neural networks have the ability to infer highly nonlinear predictions if a reference dataset is available. This work presents a geometric deep learning based multi-mesh autoencoder framework for steady-state transonic aerodynamics. The framework builds on graph neural networks which are designed for irregular and unstructured spatial discretisations, embedded in a multi-resolution algorithm for dimensionality reduction. The test case is for the NASA Common Research Model wing/body aircraft configuration. Thorough studies are presented discussing the model predictions in terms of vector fields, pressure and shear-stress coefficients, and scalar fields, total force and moment coefficients, for a range of nonlinear conditions involving shock waves and flow separation. We note that the cost of the model prediction is minimal having used an existing database.

Keywords: geometric deep learning, autoencoder, CFD, aerodynamics, graph convolutional network, multi mesh

Nomenclature

AE	=	Autoencoder
CFD	=	Computational Fluid Dynamics
CNN	=	Convolutional Neural Network
COO	=	Coordinate List
CRM	=	Common Research Model
DNN	=	Dense Neural Network
GCN	=	Graph Convolutional Network
GDL	=	Geometric Deep Learning
GNN	=	Graph Neural Network

‡ PhD Student. Corresponding author. E-mail: David.Massegur-Sampietro@southampton.ac.uk

§ Associate Professor, AIAA Senior Member.

MM	=	Multi Mesh
ML	=	Machine Learning
MSE	=	Mean Squared Error
NN	=	Neural Network
POD	=	Proper Orthogonal Decomposition
PReLU	=	Parametric Rectified Linear Unit
ROM	=	Reduced Order Model
α	=	angle of attack
$\boldsymbol{A}$, A_{ij}	=	adjacency matrix
$\hat{\boldsymbol{A}}$, $\hat{A}_{ij}$	=	adjacency matrix with self loops
b	=	constant term
β	=	slope of the PReLU function
c	=	reference chord
C_D	=	drag coefficient
C_f	=	skin friction coefficient
c_l	=	number of output channels in a layer
C_L	=	lift coefficient
C_{M_y}	=	pitching moment coefficient
C_P	=	pressure coefficient
$\boldsymbol{C_\tau}$	=	wall shear-stress coefficients
$\hat{\boldsymbol{D}}$	=	diagonal degree matrix
e	=	edge weight in a graph
$\varepsilon\,[\%]$	=	relative error
$\boldsymbol{\phi}$	=	shape functions
$\boldsymbol{I}$	=	identity matrix
$\mathcal{I}$	=	interpolation matrix
M	=	Mach number
m	=	batch size
n	=	number of grid points in a mesh
$\boldsymbol{p}(\boldsymbol{x})$	=	polynomial basis function
$p(i)$	=	probability function
Re	=	Reynolds number
S	=	reference surface
$\boldsymbol{\mathcal{S}}$	=	surface mesh nodes
$\boldsymbol{s}$	=	network input vector
$\boldsymbol{\tau_w}$	=	wall shear-stress components
$\boldsymbol{\theta}, \boldsymbol{W}$	=	model parameters
$\boldsymbol{w}$	=	weight function
$\boldsymbol{x}$	=	grid-point position in a graph
$\boldsymbol{Y}_i$	=	model prediction on the graph vertices
y	=	generic variable

Subscripts:

r	=	coarse mesh level
e	=	element edges
i	=	target node index
j	=	source node index
l	=	layer
mml	=	multi-mesh level
n	=	fine mesh level
∞	=	freestream condition

1. Introduction

Aerodynamic analyses in many engineering sectors, from aerospace to automotive, remain computationally expensive with high-fidelity computational fluid dynamics (CFD). The need to provide physical insights at very small scales around a complex geometry with limited resources contrasts with the ever pressing project deadlines. This motivates the investigation of reduced order models (ROM) for accurate approximations of the physical system via numerical techniques of reduced computational complexity. The recent breakthrough in machine learning (ML) has prompted the development of new methodologies particularly suited for ROMs. Modern, sophisticated ML algorithms are attractive for approximating highly complex and nonlinear systems from data. For example, Massegur *et al* [1] leveraged ML to predict the formation of ice on a wing section and the resulting aerodynamic degradation across a range of freezing conditions. Furthermore, Massegur *et al* [2] developed a ML based aerodynamic model coupled with a structural model for aeroelastic and flutter search analyses.

Herein, we focus on three-dimensional (3D) geometries and we address the added complexities compared to two-dimensional (2D) cases, which are around ten times more frequent in the literature. Regarding the physics, 3D problems present more abundant nonlinear flow features and interactions [3, 4], which include cross-flow interactions, wing-tip vortices and separation bubbles, among others. ROM approaches developed for 2D problems [2], featuring direct prediction of aerodynamic forces, are questionable to address the additional intricacies that arise in 3D problems. To this goal, we use deep learning neural networks (NN) [5] to model distributed aerodynamic quantities on the aircraft surface.

The primary challenge in reduced-order modelling of 3D aerodynamic fields is the large dimensional space, reflecting the numerical discretisation of the computational domain [6]. Common nonlinear ROM approaches, including Kriging [7] and dense neural networks (DNN) [8, 9] result ill-suited. To address the problem of poor scalability, convolutional neural networks (CNN) are better suited for flow-field analyses because they are designed to extract spatial features from digital images [10–12]. The problem is that CNNs are not directly suitable for unstructured meshes typical of CFD applications because of their limitation to Euclidean domains (cartesian grids). Interpolating data to a regular grid is essential, but it can lead to additional errors and loss of resolution in refined regions. Alternatively, CNN kernels of size 1 can be used as these are applicable to any mesh arrangement, as implemented by Immordino *et al* [13] or Sabater *et al* [14]. The issue with this approach is that the mesh connectivity is unused, resulting in simpler cloud-point modelling. The absence of propagation of information across the mesh limits the capability to capture coherent flow features and reconstruct continuous solutions. To leverage connectivity throughout the mesh, geometric deep learning [15] offers a range of convolutional methods, known as graph neural networks (GNN). GNNs are capable of processing graphs or manifold-unstructured meshes, which makes them a better fit for CFD applications that deal with non-Euclidean data [16]. Examples of GNNs for steady-state aerodynamics are not abundant. Ogoke *et al* [17] addressed aerodynamic solution of a 2D aerofoil rather than a more relevant test case. Baque *et al* [18] projected the 3D geometry to a simpler prismatic graph to contain the computational burden. On the other hand, comprehensive message-passing methods, as adopted by Hines *et all* [19] or Han *et al* [20], which feature dedicated learnable weights for each edge and node of the mesh, can lead to excessive memory requirements.

To maximise ROM computational efficiency when dealing with large spatial domains, dimensionality reduction to compress the domain size is crucial. The classical proper orthogonal decomposition (POD) [21, 22] is only limited to linear projections of the data onto the compressed space, causing general loss of information in highly complex physics. Autoencoders (AE) [23–26], which are the neural-network alternative to the POD, are a better alternative allowing a nonlinear compression and recovery. This work contributes generally to the question of dimensionality reduction of large datasets defined on irregular spatial domains. To this aim, an autoencoder approach embedded in the graph-based convolutional framework is sought. We solved this challenge by devising a multi-resolution scheme, inspired by the common multi-grid methods to solve partial differential equations [27]. While the methodology applies naturally to other fields and disciplines, we demonstrate the applicability on a 3D aerodynamic problem.

We propose a graph-convolutional multi-mesh autoencoder tailored for predicting distributed aerodynamic loads around a full-scale air vehicle. Unique contributions of this work are the applicability to non-Euclidean domains and a unique multi-resolution embedding for dimensionality reduction and modelling efficiency purposes. Furthermore, a building-block implementation, consisting of composition of separate NN units, facilitates evaluation of different model architectures to address the task. We have chosen the wing/body configuration of the NASA common research model (CRM) for demonstration, predicting steady-state transonic aerodynamic loads across a range of Mach numbers and angles of attack.

The paper continues in Section 2 with a description of the problem we solved and the identification of nonlinear features to be retained in the model outputs. The methodology for steady-state problems is explained in Section 3. Then, Section 4 summarises the main results. Finally, conclusions and an outlook on future work are given in Section 5.

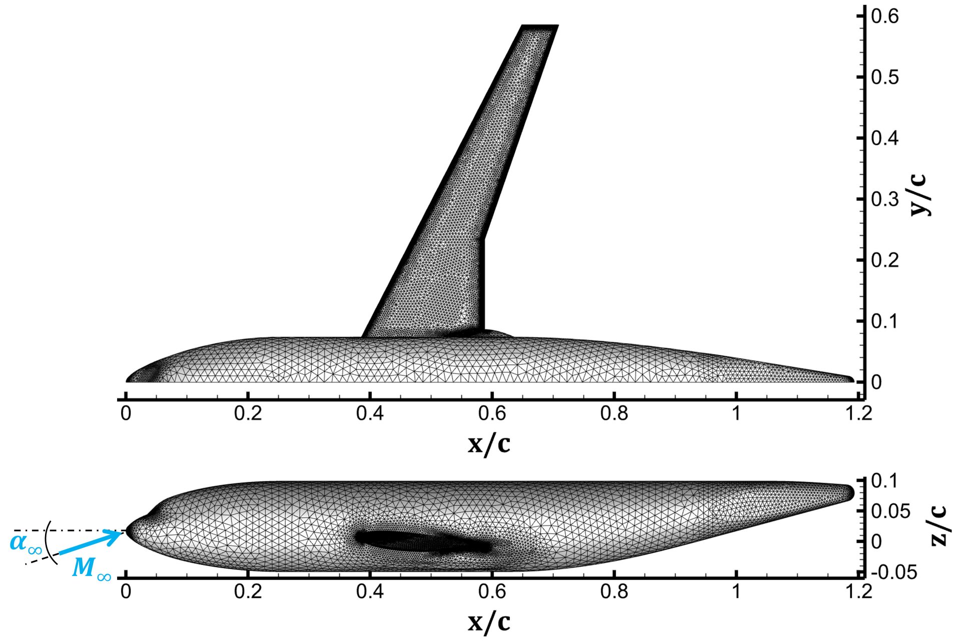


Figure 1: Surface mesh representation of the Common Research Model.

2. NASA Common Research Model Wing/Body Configuration

The test case is for the NASA Common Research Model (CRM) wing/body configuration, which is representative of a transonic transport aircraft designed to fly at a cruise Mach number of 0.8 [28, 29]. The CRM is shown in Figure 1. The reference geometric chord is $c = 0.1412$ m, the surface area is $S = 0.0727$ m^2, and the pitch moment is taken around $x_a/c = 0.5049$.

The CFD dataset was taken from Immordino *et al* [13], where `SU2 7.5` solver was used. The surface mesh shown in Figure 1 features around 78,000 grid points in an unstructured triangular topology with adapted discretisation density around the edges. The volume mesh consists of over 15 million cells, with a prism-layer mesh to promote $y^+ < 1$ near the wall, denoting the non-dimensional height of the first cell normal to the boundary. The y^+ ensures an adequate resolution of the boundary layer. The Spalart-Allmaras turbulence model was chosen. For good convergence, the JST scheme with added dissipation was adopted to discretise the convective term, Green Gauss was chosen to compute the discretised gradients and the biconjugate gradient with ILU preconditioner was applied to solve the linear solver.

In this work, we are interested in the steady-state prediction of the pressure coefficient:

$$C_P = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty U_\infty^2} \tag{1}$$

with p_∞, ρ_∞ and U_∞ the freestream pressure, density and velocity, respectively. The

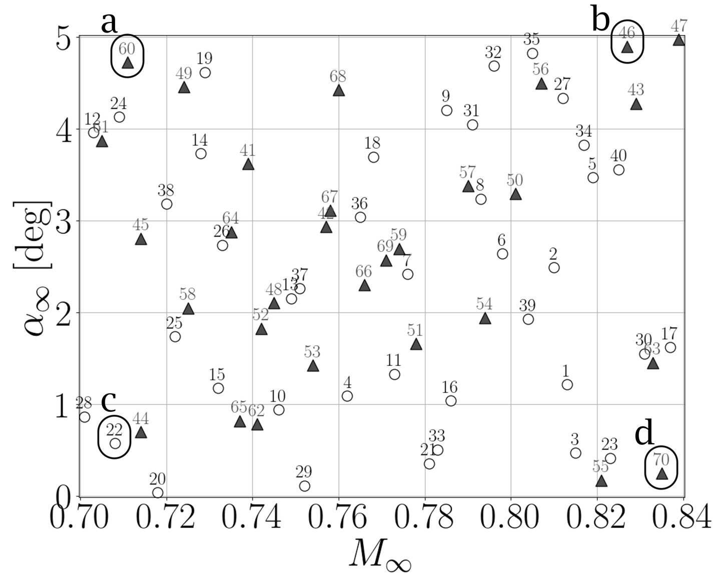


Figure 2: Dataset samples across the operating envelope of the CRM model. Circles correspond to training samples and triangles, validation cases. Lettered labels indicate selected samples to showcase the different physics across the envelope, Figures 4 and 5.

shear-stress coefficient components:

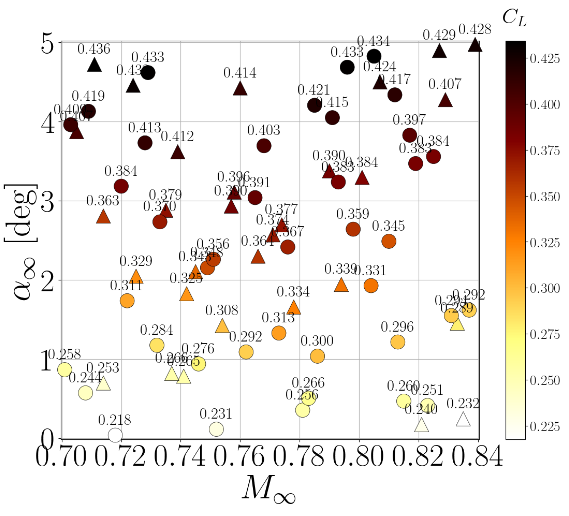
$$\boldsymbol{C_\tau} = \frac{\boldsymbol{\tau_w}}{\frac{1}{2}\,\rho_\infty\,U_\infty^2} \tag{2}$$

with $\boldsymbol{\tau_w} = [\tau_x, \tau_y, \tau_z]$, on the surface mesh of the CFD model across an envelope of Mach numbers M_∞ and angles of attack α_∞.

2.1. Reference Dataset

We used the available database of aerodynamic solutions to generate and validate our steady-state predictive model. The database contains 70 CFD calculations in the range of $M_\infty \in [0.70, 0.84]$ and $\alpha_\infty \in [0.0, 5.0]$ deg, at Reynolds number $Re = 5 \cdot 10^6$ and freestream temperature $T_\infty = 311$ K. The range of freestream conditions was chosen to have diverse nonlinear flow phenomena to be predicted by our model. The sampled conditions are shown in Figure 2, of which 40 were used for training (represented with circles) and the remaining 30 for validation (triangles). Latin Hypercube technique was used to define this limited number of samples for the preliminary envelope scan. The resulting sampling distribution did not include points on the boundaries and occasionally left large gaps between samples. Therefore, these experiments are useful to assess the performance of our framework in under-sampled spaces.

First, we delve into the variation of lift and drag coefficients, C_L and C_D, respectively, with the freestream conditions shown in Figure 3. These were obtained by integrating the pressure coefficient C_P and shear-stress $\boldsymbol{C_\tau}$ fields of the reference CFD solutions. Inspecting the colormaps, the lift coefficient correlates predominantly with


(a) Lift coefficient.

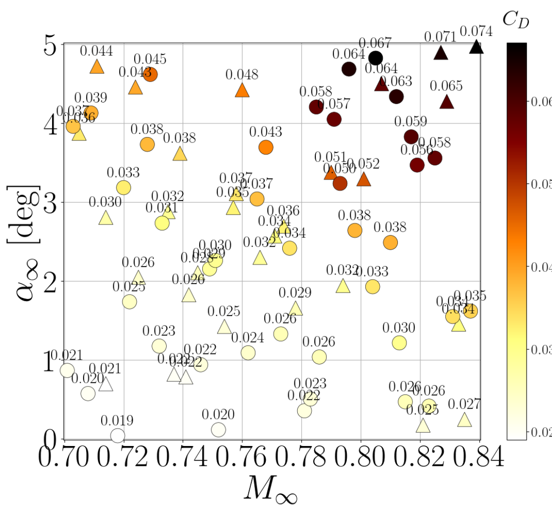
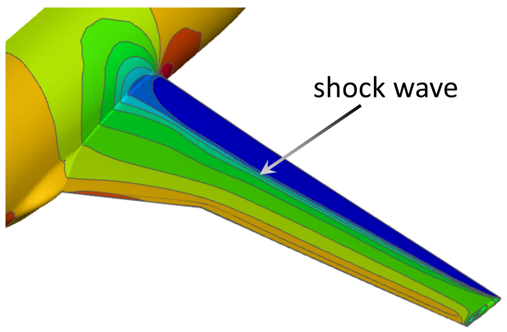

(b) Drag coefficient.

Figure 3: Integrated aerodynamic force coefficients of the NASA CRM wing/body configuration for the reference dataset, classified by training (circles) and validation (triangles) samples.

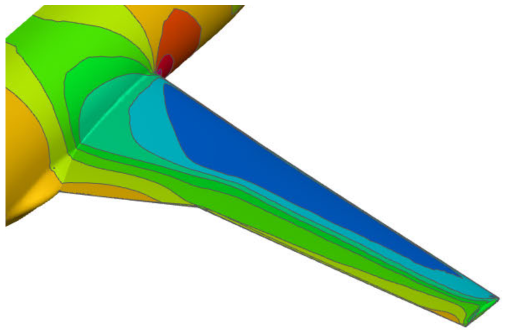
the angle of attack. The variation of the drag coefficient indicates an equal correlation to α_∞ and M_∞. Both the angle of attack and the shock-induced boundary-layer separation contribute to the drag increase.

Then, we extracted few sample points from the database for further demonstration of the complexity of our problem. Figure 4 shows the pressure coefficient for selected conditions, and Figure 5 is for the skin friction coefficient, defined as the shear-stress norm $C_f = ||\boldsymbol{C_\tau}||_2$. These Figures are arranged into four Panels, placed to reflect the location of the samples labelled in Figure 2, with low α_∞ on the bottom Panels and high M_∞ on the right Panels. The pressure distribution, Figure 4, significantly differs depending on the freestream conditions. Inspecting the lower and the upper left Panels, the location of the shock wave moves towards the leading edge and the shock intensity increases with angle of attack at lower Mach numbers. This transition appears to be nonlinear for α_∞ higher than 3 deg. With increasing Mach number, the pressure distribution becomes gradually smoother and the shock wave becomes stronger. As a result, at the highest M_∞, right Panels, the peak C_P values are lower. On the contrary, at these M_∞ conditions, the location of the shock wave remains similar with increasing angle of attack, bottom to top of the right Panels.

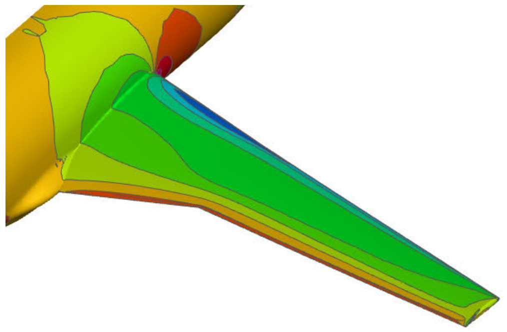
Similarly, the skin friction coefficient distributions, in Figure 5, indicate the boundary-layer separation regions (darker blue) induced by the shock wave. At low Mach number, left Panels, the separation line moves towards the leading edge with increasing angle of attack. Furthermore, in Panel (a) a separation bubble is visible. On the contrary, above $M_\infty = 0.8$, the location of the separation line remains similar across



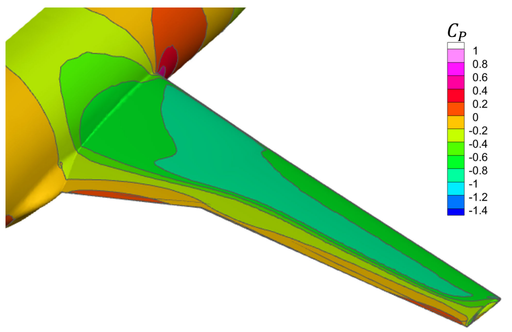
(a) $M_\infty = 0.71$, $\alpha_\infty = 4.72$ deg.



(b) $M_\infty = 0.83$, $\alpha_\infty = 4.90$ deg.

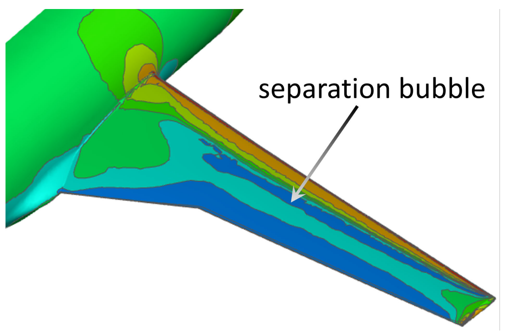


(c) $M_\infty = 0.71$, $\alpha_\infty = 0.57$ deg.

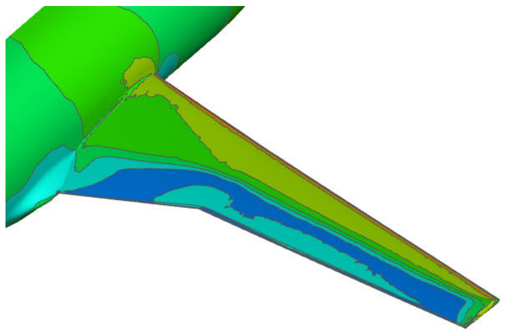


(d) $M_\infty = 0.83$, $\alpha_\infty = 0.25$ deg.

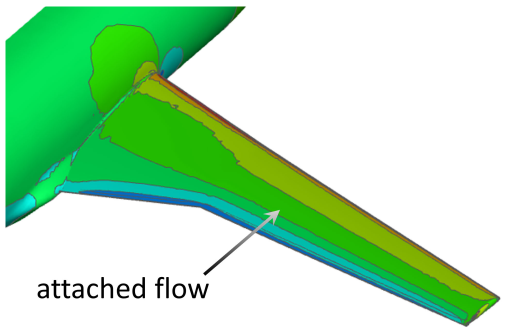
Figure 4: Pressure coefficient distribution, C_P, at the 4 sample points of Figure 2.



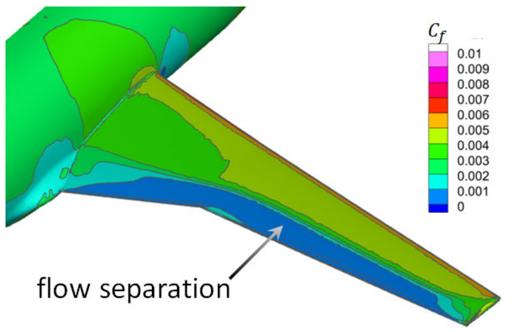
(a) $M_\infty = 0.71$, $\alpha_\infty = 4.72$ deg.



(b) $M_\infty = 0.83$, $\alpha_\infty = 4.90$ deg.



(c) $M_\infty = 0.71$, $\alpha_\infty = 0.57$ deg.



(d) $M_\infty = 0.83$, $\alpha_\infty = 0.25$ deg.

Figure 5: Skin friction coefficient distribution, C_f, at the 4 sample points of Figure 2.

the α_∞ range, as seen in the right Panels.

This brief overview sets the background problem for our ROM. The diverse nonlinear flow characteristics cannot be captured by simple direct modelling of the scalar loads, such as C_L and C_D. On the other hand, the task of predicting distributed quantities at each condition, such as C_P, is more challenging than scalar targets but is more useful from a design standpoint. This motivates our choice to devise a GDL based framework for prediction of the surface aerodynamic fields across the operating envelope.

3. Methodology

In a steady-state formulation, the aerodynamic response is considered dependent on the input conditions only, and any time dependence is neglected. A neural-network function f_{NN} is sought which maps specific user-defined inputs $\boldsymbol{s}$ to desired target fields $\boldsymbol{Y}_i$ on the surface mesh $\mathcal{S}$:

$$\boldsymbol{Y}_i = f_{\text{NN}}\left(\boldsymbol{s}, \boldsymbol{x}_i\right) \qquad \forall\, i \in \mathcal{S} \tag{3}$$

with i denoting a node in $\mathcal{S}$. The grid point coordinates $\boldsymbol{x}_i$ are also embedded. We devise an architecture for f_{NN} by leveraging GDL and dimensionality reduction for unstructurally meshed manifolds. From GDL, we resort to graph neural networks (GNN), which involve the convolution operation on graphs [16, 30].

3.1. Graph Representation

In a GNN approach, the surface mesh is represented as a graph where the vertices (or nodes) contain the position coordinates $\boldsymbol{x}_i$ and variable fields $\boldsymbol{y}_i$ (features). The graph edges connecting the grid points are determined by the mesh connectivity. Figure 6 illustrates a representation of a graph where a target node i is connected to $j \in \mathcal{S}$ surrounding grid points. Features y_i and weights e_{ij} are assigned, respectively, to each node and edge. The edges defining the graph connectivity and chosen weights are arranged to form the adjacency matrix [16]:

$$\boldsymbol{A} = e_{ij} \quad \text{for} \quad i, j = 1, ..., n \tag{4}$$

This is an $n \times n$ matrix containing the edge weights, and n is the total number of grid points in $\mathcal{S}$. The subscript ij denotes the j-th source node connected to a given node i. We assign the weights as the inverse of the distance between the two nodes forming the edge:

$$e_{ij} = \frac{1}{||\boldsymbol{x}_i - \boldsymbol{x}_j||_2} \tag{5}$$

We then normalise the edge weights to be $\in (0, 1]$, where the upper end is inclusive because self loops, i.e. $e_{ii} = 1$, are inserted by adding the identity matrix to the adjacency matrix: $\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$. In addition, we choose the edge weights to be non-directional, i.e. $e_{ij} = e_{ji}$, which results in a symmetric adjacency matrix, $\hat{\boldsymbol{A}} = \hat{\boldsymbol{A}}^T$.

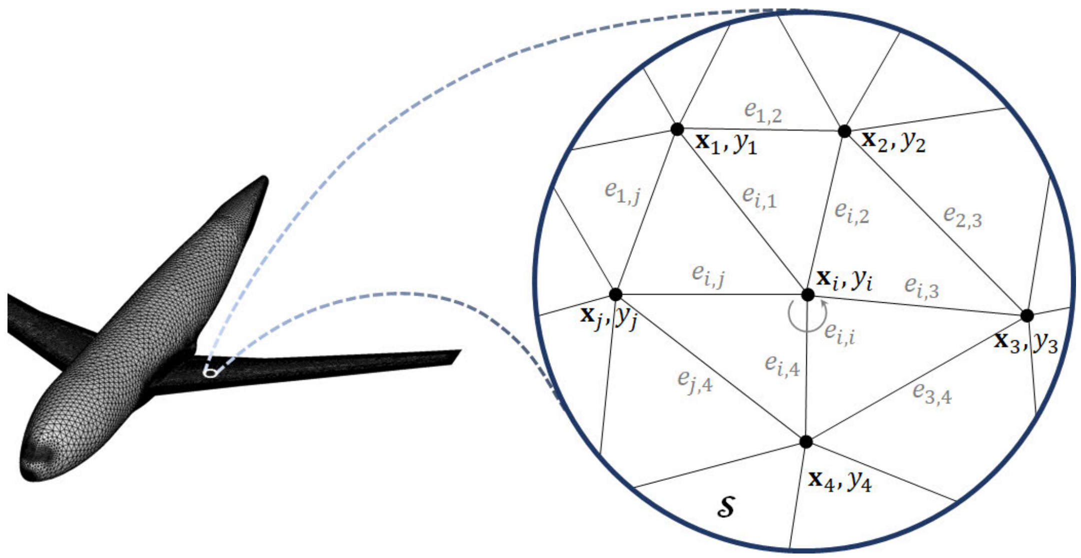


Figure 6: CRM mesh represented as a graph with node features and edge weights.

Note that $\hat{\boldsymbol{A}}$ is largely sparse as each row contains only a few non-zero elements. Consequently, it is more memory-efficient to arrange the adjacency matrix in coordinate list (COO) format. This format consists of two vectors: the edge-index and the edge-weight vectors. The edge-index vector contains the pair of node indices $[i, j]$ for each edge, of size $n_e \times 2$ and n_e the number of edges. The edge-weight vector contains the assigned weight edges e_{ij} Eq. (5), with size $n_e \times 1$. In the CRM test case, the surface mesh consists of $n = 78,829$ points and $n_e = 472,404$ edges.

3.2. Graph Convolutional Network

From the family of GNN architectures, we leverage the *graph convolutional network* (GCN) by Kipf et al. [31]. The GCN operator at a given target node is defined as:

$$g(\boldsymbol{y}) = \boldsymbol{\theta}^T \, \hat{\boldsymbol{D}}^{-\frac{1}{2}} \hat{\boldsymbol{A}} \, \hat{\boldsymbol{D}}^{-\frac{1}{2}} \boldsymbol{y} + b \tag{6}$$

with $\boldsymbol{\theta}$ a layer-specific trainable weight vector, b a constant term and $\boldsymbol{y}$ the node-based input vector at each node of the mesh $\mathcal{S}$. $\hat{\boldsymbol{D}} = \mathrm{diag}\big(\sum_{j \neq i} e_{ij} + 1\big) \; \forall \, i$ contains the sum of the edge weights connected to each node i, known as the diagonal degree matrix.

At each layer l, the GCN operation, Eq. (6), is executed on the output from the previous layer $\boldsymbol{y}_{l-1}$, followed by a nonlinear activation function h:

$$\boldsymbol{y}_l = h\big(g(\boldsymbol{y}_{l-1})\big) \tag{7}$$

We adopted for h the parametric rectified linear unit (PReLU) [32]:

$$h(y) = \begin{cases} y & \text{if} \quad y \geq 0 \\ \beta y & \text{if} \quad y < 0 \end{cases} \tag{8}$$

with β another learnable parameter.

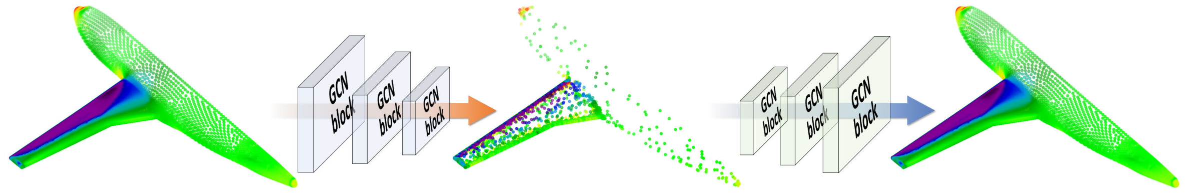


Figure 7: Autoencoder concept, involving the embedding of GCN blocks in the reduction (encoder) and reconstruction (decoder) of the field data.

Note how the GCN operator Eq. (6) takes the convolutional analogy of the CNN for Euclidean domains [33] but with a single-parameter filter swept across each row of the adjacency matrix. In fact, if we set set $e_{ij}=0$ for $i\neq j$, the standard CNN layer with kernel size 1 is obtained, as Immordino *et al* [13] opted.

3.3. GCN-based Autoencoder

From Eq. (6) we realise that successive GCN executions are required to exert influence between far-away grid points. For instance, with k_l GCN layers in succession, only k_l neighbourhoods around target node i are influenced. Consequently, the propagation of information across the mesh is slow. This leads to two issues. The first is that a deep network with an excessive number of layers would be necessary to propagate the information in refined regions. The second issue relates to the memory size of the network becoming computationally unmanageable in large spatial domains and high number of features. To alleviate these issues, as introduced in Section 1, we adopted an autoencoder approach for the compression of the spatial domain.

The autoencoder involves the projection of the states from the original domain onto a compressed space, operation known as encoder. Then, these latent states are recovered back onto the original domain in an inverse operation, known as decoder. Embedding NNs in this process makes the autoencoders more attractive than POD as nonlinear projections are possible [23]. Figure 7 lays out the autoencoder process embedded with GCNs.

3.4. Multi-Mesh Scheme

Autoencoders for discretised domains entail a multi-resolution scheme, which involves gradual coarsening operations and a subsequent refining of the grid. Reduction techniques in cartesian arrangements are trivial, including, for example, pooling operations [11]. In contrast, coarsening of unstructured meshes is a more difficult task. One of the primary challenges is that the adjacency matrix needs to be regenerated at every coarsening step. A second challenge relates to reliable transfer of information between the various grid resolutions.

To solve these challenges, we present a novel hierarchical multi-mesh (MM) scheme for the autoencoder. In the encoder process, the mesh is coarsened between blocks of

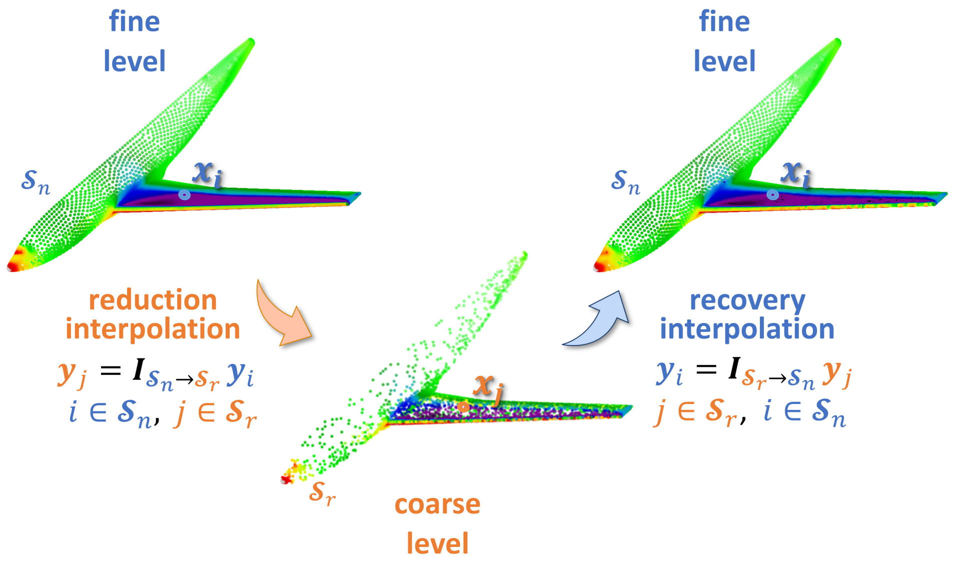


Figure 8: The 2-level multi-mesh scheme for the CRM test case, showcasing the resulting pressure reconstruction from one mesh coarsening-refining cycle.

GCN layers, intended for extraction and compression of crucial flow-field features. The latent states on the coarsened mesh are decoded in a recovery operation interleaved with additional GCN blocks, to reconstruct the solution onto the original domain. Figure 8 illustrates the coarsening and recovery operations of the proposed 2-level multi-mesh cycle for the CRM model. This method is reminiscent of the common multi-grid V-cycle algorithms to solve partial differential equations [27, 34]. Operating on a MM cycle is advantageous for: 1) reducing the computing memory size, given the compressed spatial domain; 2) extracting features of different spatial scales by means of the different mesh resolutions; and 3) enabling direct information exchange among distant nodes, avoiding the need for a deep network to spread influence across the grid, which results in a significantly smaller model.

The coarsening operation in the encoder involves the removal of grid points from the original mesh. The strategy taken to coarsen the mesh is crucial to prevent loss of essential information. For instance, a uniform random selection should preserve the original mesh topology, in terms of relative cell sizes, on the reduced mesh. However, there is risk of insufficient resolution left in regions where the initial node density was already low. On the other hand, excessive removal of nodes on originally refined regions could lead to inappropriate reconstructions where the solution is likely to present larger gradients.

For adequate representation of the distinct spatial regions at the coarse level, a balanced node selection is essential. We achieved this by selecting the nodes according to a probability function based on the corresponding face area:

$$p(i) = 1 + \frac{1 - e^{-2\frac{i}{n}}}{1 - e^{-2}}\,(p_1 - p_n) + p_1 \quad \text{for} \quad i = 1...n \quad \text{with} \quad a_{i-1} < a_i \tag{9}$$

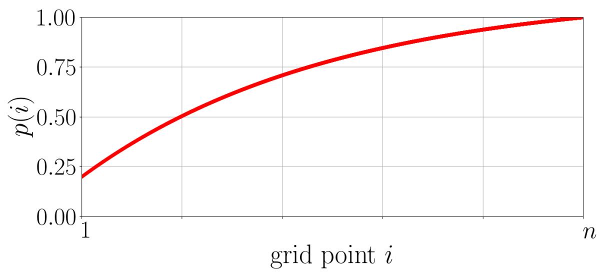


Figure 9: Probability density function to select the grid points in the mesh coarsening operation. Grid points ordered by face area in ascending order.

Table 1: Surface mesh sizes (number of nodes) of the two multi-mesh levels for the CRM case.

(level 0) Original mesh size $\mathcal{S}_\mathrm{n}$	**(level 1) Coarsened mesh size $\mathcal{S}_\mathrm{r}$**	**Compression ratio $\frac{\mathcal{S}_\mathrm{n}}{\mathcal{S}_\mathrm{r}}$**
78,829	5,000	15.8

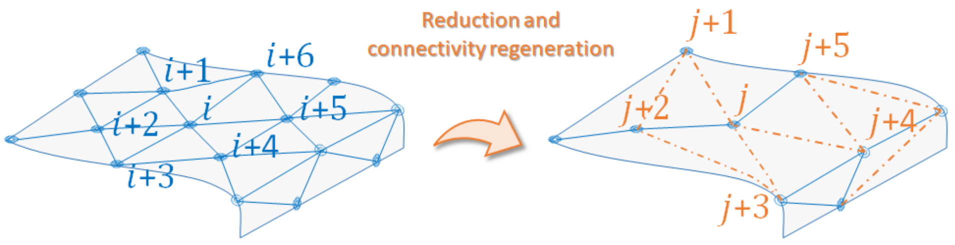


Figure 10: Graph coarsening procedure and connectivity regeneration as part of the dimensionality reduction algorithm.

with i the index of the grid points sorted by their face area a_i in ascending order, and n the total node count. We chose $p_1 = 0.2$ and $p_n = 1$ for the smallest and largest elements, respectively. The resulting distribution is demonstrated in Figure 9. Probability of being selected is higher in nodes with larger face areas, likely in unrefined regions, as opposed to nodes found in dense discretisations. The coarsened mesh resulting from this selection probability is illustrated at the bottom of Figure 8. The mesh size (node count) in the various multi-resolution levels is reported in Table 1 for the CRM test case. Note how a compression ratio of 16 was adopted.

Upon selection of the grid points to be kept in the coarse mesh, the graph connectivity was regenerated by reconnecting remaining nodes that shared connections with discarded ones. Figure 10 demonstrates the process of restoring graph connectivity following a coarsening operation, with new edges shown in orange.

3.5. Weighted Moving Least Squares for Grid Interpolation

As shown in Figure 8, information must be transferred across multiple grids. In the reduction step of the encoder, the original field data must be interpolated onto the compressed mesh. In the decoder, the recovery of the latent states onto the fine grid entails an inverse interpolation. These are critical operations in the MM cycle. The interpolation consists of a functional $\boldsymbol{\mathcal{I}}_{\mathcal{S}_\mathrm{n}\to\mathcal{S}_\mathrm{r}} : \mathbb{R}^{n_n} \to \mathbb{R}^{n_r}$ to map a spatial field $\boldsymbol{y}_i$ from a source grid $\mathcal{S}_\mathrm{n}$, which contains n_n nodes, to a target grid $\mathcal{S}_\mathrm{r}$, with n_r nodes, where both grids discretise the same spatial domain:

$$\boldsymbol{y}_j = \boldsymbol{\mathcal{I}}_{\mathcal{S}_\mathrm{n}\to\mathcal{S}_\mathrm{r}}\, \boldsymbol{y}_i \qquad j \in \mathcal{S}_\mathrm{r}, \forall\, i \in \mathcal{S}_\mathrm{n} \tag{10}$$

with $\boldsymbol{\mathcal{I}}_{\mathcal{S}_\mathrm{n}\to\mathcal{S}_\mathrm{r}}$ the interpolation matrix from the source to the target mesh, and $\boldsymbol{y}_j$ the interpolated field data.

The following properties are desirable for an adequate interpolation [35]: 1) interpolated values at the source nodes should match the original data; 2) integrated resultants should be conserved; and 3) interpolated fields should be continuous. Consequently, directly recasting the data across coincident points and nearest-neighbour interpolation, as Han *et al* [20] proposed, result inappropriate because conservation and continuity properties are not satisfied.

There is multitude of multi-grid algorithms, often devised to accelerating the solution of finite-volume discretisation of partial differential equations, as proposed for example by Smith *et al* [34]. However, we chose a different approach which is particularly suited for fluid-structure interaction problems, where satisfying the above properties is crucial for adequate transfer of loads and deflections across models. In particular, we implemented a weighted moving least squares (WMLS) scheme [35, 36]. The idea is to generate a shape function $u(\boldsymbol{x})$ to approximate the input data y_i at source nodes $i \in \mathcal{S}_\mathrm{n}$ with coordinates $\boldsymbol{x}_i$ by least-square-error minimisation:

$$\min_{\boldsymbol{a}} \; \mathcal{L} = \sum_{i \in \mathcal{S}_\mathrm{n}} \left(u(\boldsymbol{x}_i) - y_i\right)^2 \, \boldsymbol{w}(||\boldsymbol{x} - \boldsymbol{x}_i||) \tag{11}$$

where the weight $\boldsymbol{w}(||\boldsymbol{x} - \boldsymbol{x}_i||)$ is a function of the distance between the source and the target points. We specify $u(\boldsymbol{x})$ as a polynomial combination:

$$u(\boldsymbol{x}) = \boldsymbol{p}^T(\boldsymbol{x})\, \boldsymbol{a} \tag{12}$$

with $\boldsymbol{p}(\boldsymbol{x}) = [1, x, y, z, x^2, y^2, z^2, xy,\, xz, yz]^T$ the second-order polynomial basis function, and $\boldsymbol{a}$ the vector of respective coefficients. The analytical solution of the least square minimisation can be shown to yield the resulting approximation at every node $j \in \mathcal{S}_\mathrm{r}$ of the target grid:

$$u(\boldsymbol{x}_j) = \boldsymbol{\phi}(\boldsymbol{x}_j)\, \boldsymbol{y}_i \qquad j \in \mathcal{S}_\mathrm{r}, \forall\, i \in \mathcal{S}_\mathrm{n} \tag{13}$$

with the shape functions defined as:

$$\boldsymbol{\phi}(\boldsymbol{x}_j) = \boldsymbol{p}^T(\boldsymbol{x}_j) \left(\boldsymbol{P}^T \boldsymbol{W} \boldsymbol{P}\right)^{-1} \boldsymbol{P}^T \boldsymbol{W} \tag{14}$$

and

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{p}^T(\boldsymbol{x}_1) \\ \boldsymbol{p}^T(\boldsymbol{x}_2) \\ \vdots \\ \boldsymbol{p}^T(\boldsymbol{x}_{n_n}) \end{bmatrix} \tag{15}$$

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{w}(||\boldsymbol{x}_j - \boldsymbol{x}_1||) & 0 & \dots & 0 \\ 0 & \boldsymbol{w}(||\boldsymbol{x}_j - \boldsymbol{x}_2||) & \dots & 0 \\ \vdots & & & \\ 0 & 0 & \dots & \boldsymbol{w}(||\boldsymbol{x}_j - \boldsymbol{x}_{n_n}||) \end{bmatrix} \tag{16}$$

Therefore, the interpolation matrix $\boldsymbol{\mathcal{I}}_{\mathcal{S}_\mathrm{n} \to \mathcal{S}_\mathrm{r}}$ is:

$$\boldsymbol{\mathcal{I}}_{\mathcal{S}_\mathrm{n} \to \mathcal{S}_\mathrm{r}} = \begin{bmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1) \\ \boldsymbol{\phi}(\boldsymbol{x}_2) \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_{n_r}) \end{bmatrix} \tag{17}$$

The least squares approximation, Equation (14), must be computed for each target grid point, resulting in computationally intractable matrix operations when dealing with large meshes. To reduce the computing burden, we adopt a moving interpolation consisting of limiting each target node to be influenced only by the k_n closest source points:

$$\boldsymbol{w}(||\boldsymbol{x}_j - \boldsymbol{x}_i||) = \begin{cases} 1 & \text{for } \; ||\boldsymbol{x}_j - \boldsymbol{x}_i|| < ||\boldsymbol{x}_j - \boldsymbol{x}_{i+1}|| \; \text{ with } \; i = 1...k_n \text{ and } i \in \mathcal{S}_\mathrm{n} \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

In this work, we found $k_n = 10$ a good compromise between interpolation accuracy and computational efficiency. Note that $\boldsymbol{\mathcal{I}}_{\mathcal{S}_\mathrm{n} \to \mathcal{S}_\mathrm{r}}$ is a largely sparse and non-square matrix of size $n_r \times n_n$, each row containing just k_n non-zero values. This matrix is not invertible and two different interpolation matrices must be generated for the encoder and decoder operations, $\boldsymbol{\mathcal{I}}_{\mathcal{S}_\mathrm{n} \to \mathcal{S}_\mathrm{r}}$ and $\boldsymbol{\mathcal{I}}_{\mathcal{S}_\mathrm{r} \to \mathcal{S}_\mathrm{n}}$, respectively. Figure 8 illustrates the dual interpolation process. We observe how the resulting pressure reconstruction (right) after execution of the MM cycle matches well the original field (left).

3.6. Steady-state Prediction Framework

We now complete the construction of the predictive model architecture, here referred to as steady-state GCN-MM-AE. For the CRM use case, the target fields are the pressure coefficient and the shear stress components, $\boldsymbol{Y} = [C_P, C_{\tau_x}, C_{\tau_y}, C_{\tau_z}]$, across the input envelope of Mach numbers and angles of attack, $\boldsymbol{s} = [M_\infty, \alpha_\infty]$. The architecture of the final steady-state GCN-MM-AE model is shown in Figure 11. The scalar inputs $\boldsymbol{s}$ are casted to each node of the graph, concatenated to the grid-point coordinates $\boldsymbol{x}_i \; \forall \, i \in \mathcal{S}_\mathrm{n}$. The input vectors are processed by the encoder, involving the coarsening step of the MM

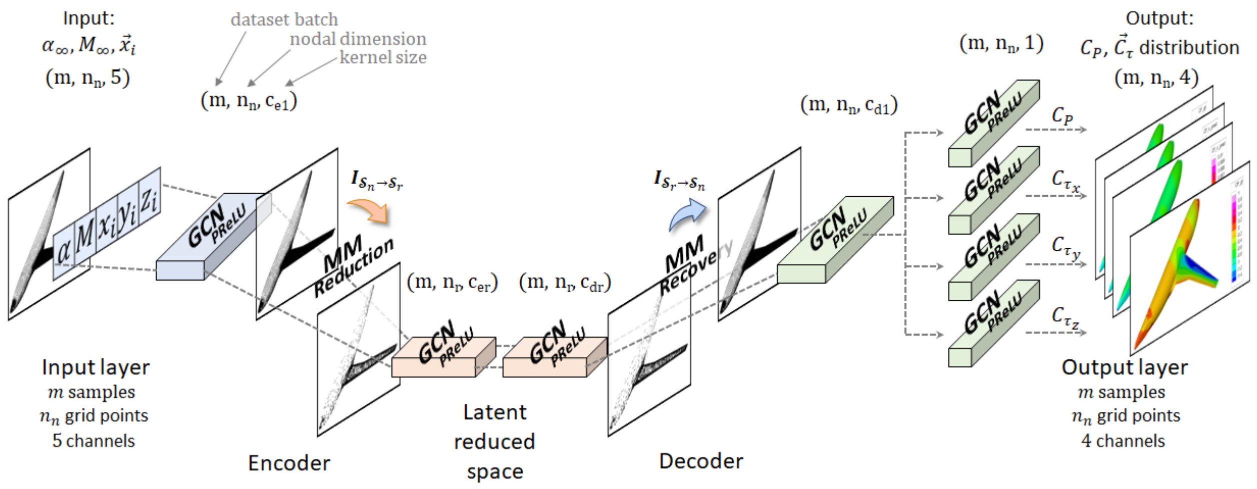


Figure 11: Steady-state GCN-MM-AE model architecture for aerodynamic predictions of the CRM test case.

cycle and two GCN blocks. Subsequently, the decoder comprises the recovery operation of the MM embedded in two additional GCN blocks. The network ramifies at the end into separate blocks for each field quantity to predict. This architecture defines f_{NN} in Eq. (3).

To the best of our knowledge, this framework is novel on several fronts: 1) GDL based autoencoder framework for spatial predictions on large and unstructured discretisations, applied to an aerospace problem; 2) multi-resolution scheme embedded in the nonlinear autoencoder for dimensionality reduction of unstructured manifolds, aimed at capturing different spatial scales, promoting influence across the grid and maximising ROM computational efficiency; and 3) building-block functionality to address distinct tasks within the same framework: multi-resolution reconstructions, steady-state predictions and extension to dynamic simulations. This framework was implemented using `PyTorch 1.11`, an optimised deep-learning library in `Python`, and `PyTorch Geometric`, an open-source graph neural-network package built upon `PyTorch`.

4. Results

This Section is organised in two parts. The first part focuses on the prediction of scalar quantities - in our case, the integrated force and moment coefficients. The second part is related to the model prediction of the distributed fields - the pressure and the shear-stress coefficient distributions. The Appendix contains more background information, including the steady-state GCN-MM-AE architecture from Figure 11 and the model optimisation procedure. Comparison between our WMLS scheme and the multi-grid method by Smith *et al* [34] for the two-way interpolation of the MM cycle was also investigated. To complete the framework set-up, sensitivity assessments to key hyperparameters, such as the training set size, the model weights and the MM cycle set-up, are also reported. Worth noting that at each sample point the model prediction outputs C_P and $\boldsymbol{C_\tau}$ distributions, from which the resulting force and moment coefficients were obtained by integration.

4.1. Integrated Loads

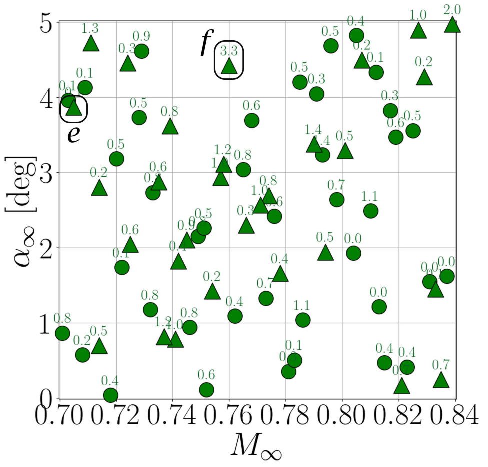
Figure 12 presents the analysis for the integrated lift coefficient C_L, drag coefficient C_D and pitching moment coefficient C_{M_y}. The left Panels show the prediction error on the dataset samples, with training samples in circles and validation in triangles. The error is defined as:

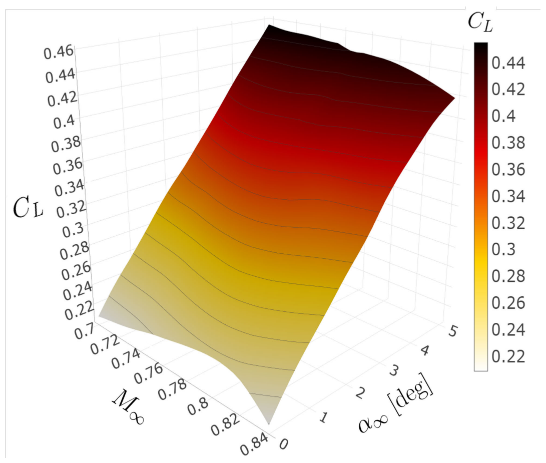
$$\varepsilon_{C_{\boldsymbol{y}}}\,[\%] = \frac{|C_{\boldsymbol{y}\mathrm{ROM}} - C_{\boldsymbol{y}\mathrm{CFD}}|}{|C_{\boldsymbol{y}\mathrm{CFD}}|} \cdot 100 \tag{19}$$

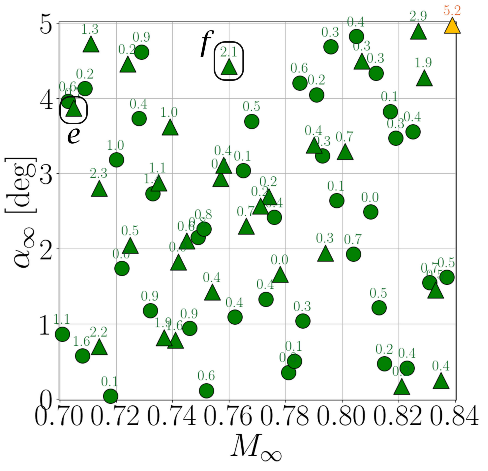
with $\boldsymbol{y} = [L, D, M_y]$. The errors are classified by a traffic-light color scheme: green corresponds to prediction errors below 4%, amber between 4% and 10%, and red above 10%. For convenience, the percent error is reported above each sample point. Best and worst predictions are highlighted as A and B, respectively. The prediction error is generally small throughout for the C_L predictions, where the error is below 2.2%. Additionally, the worst prediction (B) was found at $M_\infty = 0.76$, $\alpha_\infty = 4.42$ deg, with pitching moment error of 24.3%. This point stands out for not including training samples within a wide surrounding. The model found more difficult learning this region of the envelope. An adaptive sampling method to include training samples in under-sampled regions would be convenient for improved model accuracy. However, this is beyond the scope of this work. We also found the C_{M_y} prediction accuracy degrades slightly towards high angles of attack, where nonlinear aerodynamic response occurs. Nevertheless, the accuracy of the model is overall high.

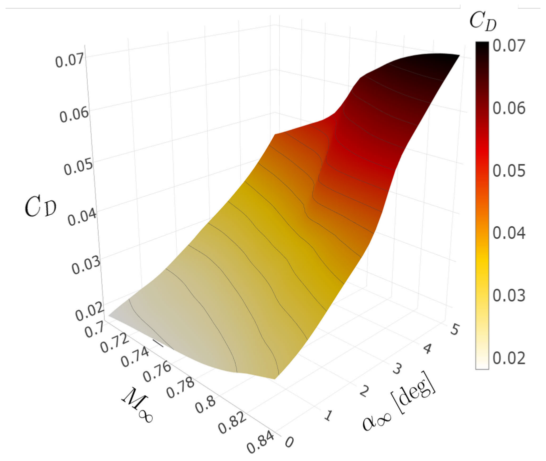
Table 2 provides a statistical summary of the prediction errors. The average error, standard deviation and the worst prediction for each aerodynamic coefficient across the complete dataset and the validation dataset are reported. The statistics are similar between datasets. This suggests that there is no over-fitting and the model performs well to new conditions. In addition, the low standard deviations indicate good model precision. The average errors were also found low, with C_L the best predicted quantity. The errors for the C_D and C_{M_y} tend to be skewed by the reference values being an order of magnitude smaller than the lift values. Last, the worst statistics are for the C_{M_y}, consequence of the larger errors at high angles of attack. Small errors on the shock-wave location around the reference axis can contribute to a magnified error too.

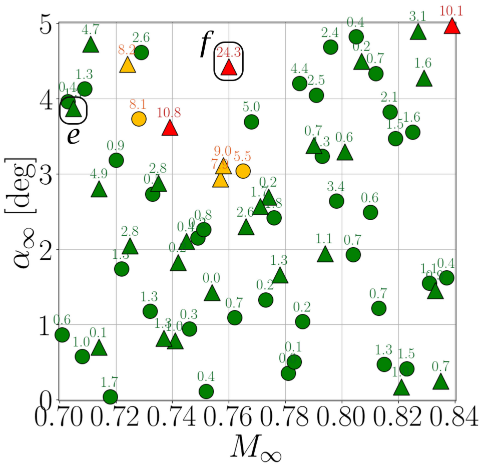
The adoption of the traffic-light system for the error plots in Figure 12 is convenient to judge the adequacy of the model for design purposes. The low error obtained for the lift is essential because this is regarded the most important design parameter. In contrast, larger errors for the drag and pitching moments are acceptable due to the smaller magnitudes. In general, errors lower than 4-5% (green) are within typical simulation tolerance and, therefore, acceptable. For errors between 5 and 10-15%


(a) C_L percent error, ε_{C_L}, whole dataset.


(b) C_L, full envelope.


(c) C_D percent error, ε_{C_D}, whole dataset.


(d) C_D, full envelope.


(e) C_{M_y} percent error, $\varepsilon_{C_{M_y}}$, whole dataset.

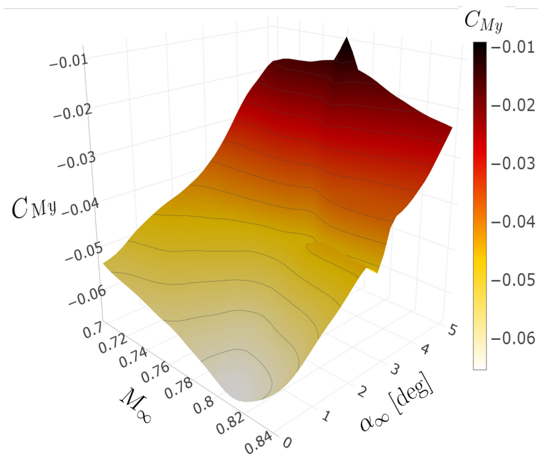

(f) C_{M_y}, full envelope.

Figure 12: Aerodynamic coefficient predictions with the steady-state GCN-MM-AE model for the CRM test case. Error plots on the left are classified by training (circles) and validation (triangles) samples; percent error below 4% in green, amber between 4% and 10%, and red larger than 10%.

Table 2: Statistical summary of the prediction error for the CRM test case and various datasets.

	ε_{C_L} [%]			ε_{C_D} [%]			$\varepsilon_{C_{M_y}}$ [%]		
	mean	std dev	worst	mean	std dev	worst	mean	std dev	worst
Full dataset	0.6	0.5	3.3	0.7	0.8	5.2	2.4	3.6	24.3
Validation set	0.8	0.7	3.3	1.0	1.1	5.2	3.5	5.0	24.3

(amber), engineering judgement should take consideration of the discrepancies. Larger errors (red) could cause wrong aerodynamic design directions. Action should be taken to improve the model, by iteratively including new training experiments and regenerating the model until acceptable error is achieved. In practice, however, the error tolerance is determined by application. For example, multiphysics simulations (e.g. aeroelastic analyses) require stricter modelling tolerances from each separate model than single physics simulations (e.g. common aerodynamic responses). Nevertheless, there is no risk of unpredicted structural failures as safety factors are enforced to be over 200%.

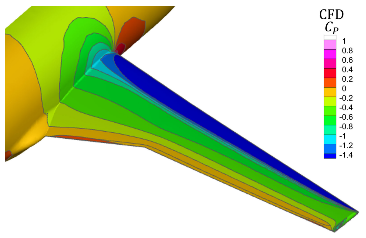
4.2. Predicted Distributions

Figures 13 and 14 analyse the predicted pressure coefficient field for the two labelled conditions in Figure 12. The contour plots illustrate the reference C_P solution from CFD (left), the prediction by our model (middle) and the error (right). Panel (d) compares the C_P distributions at the cross sections specified in the right contour. We observe that in sample e the prediction is in good agreement over the whole surface, Figure 13. For sample f, the model is overall correct except for a small discrepancy on the shock-wave location, predicted slightly further downstream from 30% of the span, Figure 14.

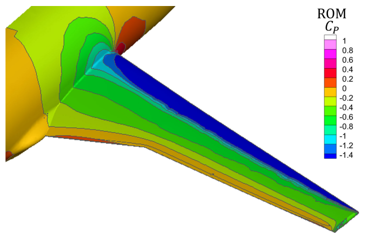
Figures 15 and 16 provide a similar comparison for the skin friction coefficient C_f. The shear-stress field is also found correct in the first condition. In the second case, we observe that the boundary-layer separation is slightly delayed. This is in line with the predicted location of the shock wave shown in the previous Figure. Remarkably, this result indicates that our proposed framework appears to understand the relationship between the physical quantities. This demonstrates the reason for developing a single model for multiple target fields.

4.3. Full Envelope Prediction

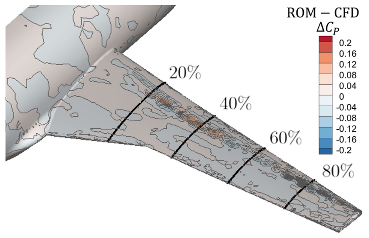
The implemented ROM is convenient to efficiently interrogate the complete operating envelope. The right Panels in Figure 12 present the resulting aerodynamic maps. The 3D contour plots were built by integrating the surface field predictions of 30 × 30 samples uniformly distributed. The C_L correlates with α_∞ as expected, with the nonlinear C_L slope at high angles of attack also well captured by the model. In addition, a shallow valley along the diagonal is visible. This seems to be caused by the shock



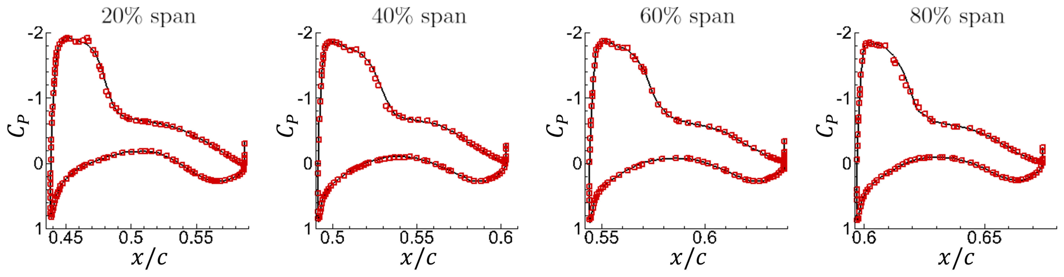
(a) C_P from CFD reference.



(b) Predicted C_P.

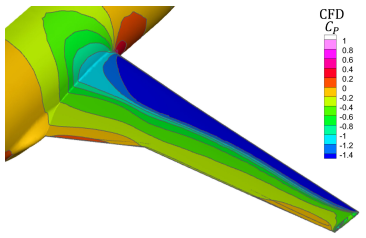


(c) C_P error (difference).

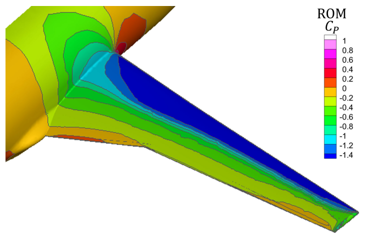


(d) C_P distribution at different cross sections for the ROM prediction (red squares) and the CFD reference (black curve).

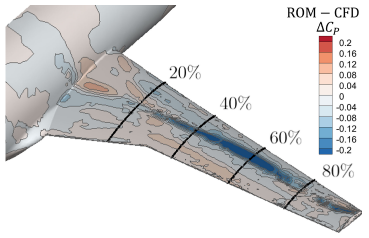
Figure 13: Pressure coefficient C_P comparison of our steady-state GCN-MM-AE model against CFD reference at $M_\infty = 0.70$ and $\alpha_\infty = 3.87$ deg (sample e in Figure 12); prediction error $\varepsilon_{C_L} = 0.1\%$, $\varepsilon_{C_D} = 0.7\%$ and $\varepsilon_{C_{M_y}} = 1.3\%$.



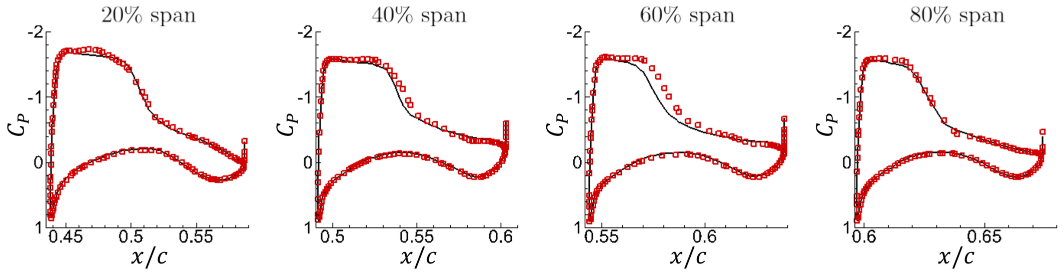
(a) C_P from CFD reference.



(b) Predicted C_P.

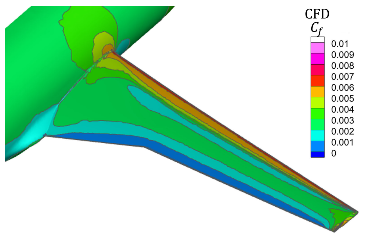


(c) C_P error (difference).

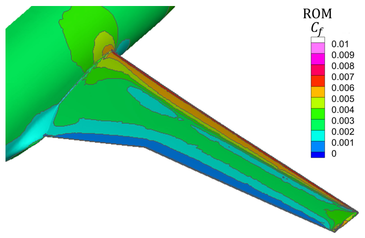


(d) C_P distribution at different cross sections for the ROM prediction (red squares) and the CFD reference (black curve).

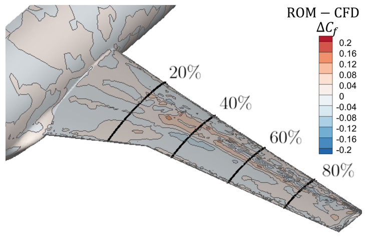
Figure 14: Pressure coefficient C_P comparison of our steady-state GCN-MM-AE model against CFD reference at $M_\infty = 0.76$ and $\alpha_\infty = 4.42$ deg (sample f in Figure 12); prediction error $\varepsilon_{C_L} = 3.3\%$, $\varepsilon_{C_D} = 2.1\%$ and $\varepsilon_{C_{M_y}} = 24.3\%$.



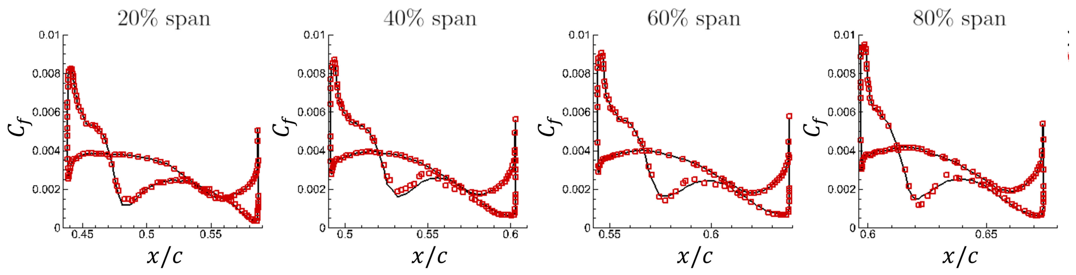
(a) C_f from CFD reference.



(b) Predicted C_f.

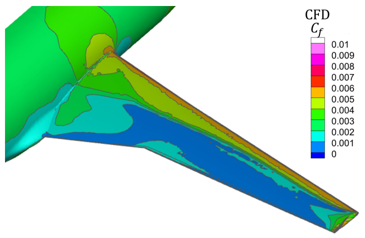


(c) C_f error (difference).

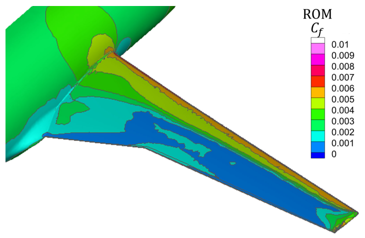


(d) C_f distribution at different cross sections for the ROM prediction (red squares) against CFD reference (black curve).

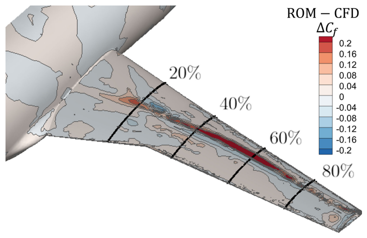
Figure 15: Skin friction C_f comparison of our steady-state GCN-MM-AE model and CFD reference at $M_\infty = 0.70$ and $\alpha_\infty = 3.87$ deg (sample e); prediction error $\varepsilon_{C_L} = 0.1\%$, $\varepsilon_{C_D} = 0.7\%$ and $\varepsilon_{C_{M_y}} = 1.3\%$.



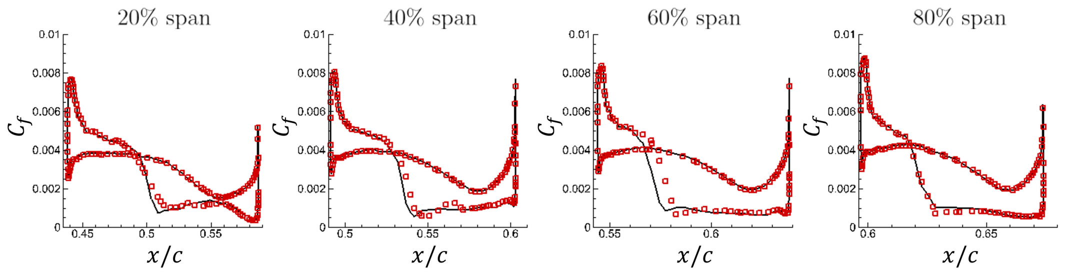
(a) C_f from CFD reference.



(b) Predicted C_f.



(c) C_f error (difference).



(d) C_f distribution at different cross sections for the ROM prediction (red squares) and the CFD reference (black curve).

Figure 16: Skin friction C_f comparison of our steady-state GCN-MM-AE model against CFD reference at $M_\infty = 0.76$ and $\alpha_\infty = 4.42$ deg (sample f); prediction error $\varepsilon_{C_L} = 3.3\%$, $\varepsilon_{C_D} = 2.1\%$ and $\varepsilon_{C_{M_y}} = 24.3\%$.

Table 3: Computing costs for steady-state modelling of the CRM test case.

Computing Task	Cost	
1 steady-state CFD run:	$\sim$400	CPU-h
Training dataset of 40 CFD run:	$\sim$16,000	CPU-h
ROM training execution:	$\sim$2.5	GPU-h
1 steady-state ROM prediction:	$\sim$0.00014	GPU-h
Full-envelope of 900 ROM prediction:	$\sim$0.1	GPU-h

wave intensifying while the peak pressure gradually decreases. The isocontours on the C_D envelope reveal a nonlinear behaviour along the α_∞ axis, which resembles the expected quadratic dependence, especially at low M_∞. The drag increase at higher Mach numbers is related to the boundary-layer separation induced by the shock wave. Last, the C_{M_y} envelope reveals highly nonlinear phenomena. The pitching moment decreases (in magnitude) with α_∞ caused by the shock wave intensifying and moving upstream. By contrast, C_{M_y} is largest at low α_∞ and $M_\infty \sim 0.81$ consequence of the downstream location of the shock wave. The small spike observed at high α_∞ is likely consequence of the lack of sampling in that region.

4.4. Note on Computing Costs

A summary of the computing costs involved in the deployment of the framework and the significant saving compared to high-fidelity simulations is reported in Table 3. The steady-state CFD conditions by Immordino *et al* [13] were solved with a 120-core HPC, totalling up to 16,000 CPU-h for the 40 training CFD samples. The ROM was generated with a 6GB GPU and the training process required around 2.5 GPU-h. New aerodynamic predictions are completed in less than a second, rather than almost 3 hours in CFD, i.e. a speed-up of well over 99.9%. As a result, the 30×30 samples to construct the 3D envelopes in the right Panels of Figure 12 were completed within minutes, whereas it would be impractical using only CFD. Furthermore, to address a typical aerodynamic characterisation campaign [37], comprising 10 points along the angle-of-attack axis and a Mach-number resolution of 0.02, the overall computing gain, including the generation of the training dataset in CFD, would still be up to 50%.

5. Conclusions

The flow analysis around a three-dimensional aircraft remains an expensive task despite access to larger and more performing computing services than ever before. This limitation takes on an even higher criticality when the designer is tasked with delivering the performance of the system across a range of relevant flow conditions. In engineering, the loads experienced by a reentry vehicle passing through the Earth's atmosphere, the stability and control characteristics of a transport aircraft across the flight envelope or the aerodynamic map of a racing car are examples of common tasks. To overcome the computational burden associated with running a multitude of computational fluid dynamics analyses, whose number is at the designer's discretion to obtain data in time for pressing deadlines, the use of reduced order models is a viable alternative. However, these models still present a number of challenging decisions. The most critical decision one has to make is the choice of the mathematical structure for the reduced-order model.

We developed a geometric deep-learning autoencoder framework to achieve a cost-effective predictive model of output quantities of interest defined on a large spatial domain with an unstructured, irregular discretisation. The framework dealt with over 300 thousand outputs and about 80 thousand grid points distributed on a three-dimensional discretisation of a wing/body aircraft configuration. We faced specific challenges that required the development of a novel approach, which can be taken to other disciplines with immediate applicability. The first challenge is represented by the large set of points used for the spatial discretisation. We created a dedicated multi-resolution autoencoder for extracting multi-scale features from the data field, transferring influence across the domain and for memory efficiency. The second challenge is related to the unstructured and irregular point discretisation. We made use of graph convolutional networks that enable the convolutional operation on irregular domains using the mesh connectivity, very attractive to emulate high-fidelity computational engineering analyses. The resulting predictive framework offers a novel approach when data are defined on large three-dimensional unstructured manifolds, which is the case for any realistic problem. Nonetheless, the framework keeps the ability to use the simpler cases of structured domains when available.

It is worth noting that the steady-state framework builds on one single model that outputs multiple vector fields distributed on an unstructured domain. For our application to aircraft aerodynamic loads, the predicted pressure and shear-stress coefficient distributions were integrated in space to calculate the total force and moment coefficients. This operation not only mimics the way integrated loads are obtained in a computational fluid dynamics solver, but there are advantages too. First, the generation, training and validation of one single model is noticeably cheaper and easier than doing it for two separate and distinct models. Then, it avoids having two independent models that may be best fit for their specific outputs but do not recover the actual relationship among the distinct outputs, i.e. integration in space for our aerodynamic problem. Finally, a physically sound distribution of flow quantities obtained from one single model leads to a sound interpretation of aerodynamic loads.

The NASA Common Research Model wing/body aircraft configuration was used for demonstration. We used an existing database of 70 pre-computed cases to generate and validate the predictive model. In the reference study that provided us the database, sample points were placed across the flight envelope using a Latin Hypercube method, which is not receptive of any feature learned during the design space exploration. The model predictions achieved a good match to reference data across the flight envelope. It is expected that further improvements in model predictions are obtained using an adaptive design of experiments where sample points are placed at strategic locations of the design space. Once the model is generated, load predictions across the whole flight envelope of angle of attack and Mach number is possible within minutes. A thorough study demonstrated that the model captured the various nonlinear effects throughout the envelope, including variations of shock wave strength and position with the angle of attack and Mach number, and the appearance of shock-induced boundary-layer separation at certain flow conditions.

Today, there is an abundance of data from calculations and measurements. Our choice of using an existing database reflects this situation. Cost-wise, model predictions are obtained at a minimal cost, 0.1%, compared to running the computational fluid dynamics solver. In design applications, the issue of database generation is still actual. To minimise the preliminary data requirements, a combined data-driven with physics-knowledge implementation could be thought by embedding physics-informed loss terms during training. However, application of physics terms on the surface of a complex geometry is not trivial and proper consideration is sought as the fluid dynamics equations are formulated for the fluid volume. We believe the generalisation to variable shapes is attractive to further expand the applicability of our model. Since our model is designed to include the coordinates of the mesh nodes as inputs, the current implementation may be adapted to wing deflections for static, three-dimensional aeroelastic analysis and for aerodynamic shape design and optimisation.

Funding

This research received no external funding.

References

[1] Massegur D, Clifford D, Da Ronch A, Lombardi R and Panzeri M 2023 Low-Dimensional Models for Aerofoil Icing Predictions *Aerospace Journal* **10** 444 ISSN 2226-4310 URL `https://www.mdpi.com/2226-4310/10/5/444`

[2] Massegur D, Immordino G, Da Ronch A, Righi M, Düzel S, Anderegg D and Soukhmane I 2022 ROM-Based Uncertainties Quantification of Flutter Speed Prediction of the BSCW Wing *AIAA SCITECH 2022 Forum* (Reston, Virginia: American Institute of Aeronautics and Astronautics) ISBN 978-1-62410-631-6 URL `https://arc.aiaa.org/doi/10.2514/6.2022-0179`

[3] Anderson J 2016 *Fundamentals of Aerodynamics* (McGraw-Hill Education) ISBN 9781259129919 URL `https://books.google.co.uk/books?id=D1ZojgEACAAJ`

[4] Pope S B 2000 *Turbulent Flows* (Cambridge University Press)

[5] Goodfellow I, Bengio Y and Courville A 2016 *Deep Learning* (MIT Press) `http://www.deeplearningbook.org`

[6] Brunton S L and Kutz J N 2019 *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control* 1st ed (USA: Cambridge University Press) ISBN 1108422098

[7] Glaz B, Liu L and Friedmann P P 2010 Reduced-order nonlinear unsteady aerodynamic modeling using a surrogate-based recurrence framework *AIAA Journal* **48** 2418–2429 ISSN 00011452

[8] Li J, Du X and Martins J R 2022 Machine learning in aerodynamic shape optimization *Progress in Aerospace Sciences* **134** 100849 ISSN 03760421

[9] Wang Q, Cesnik C E and Fidkowski K 2020 Multivariate Recurrent Neural Network Models for Scalar and Distribution Predictions in Unsteady Aerodynamics *AIAA Scitech 2020 Forum* (Reston, Virginia: American Institute of Aeronautics and Astronautics) ISBN 978-1-62410-595-1 URL `https://arc.aiaa.org/doi/10.2514/6.2020-1533`

[10] Sureshbabu S, Tejero F, Sanchez-Moreno F, MacManus D G and Sheaf C 2023 Deep-Learning Methods for Non-Linear Transonic Flow-Field Prediction *AIAA AVIATION 2023 Forum* (Reston, Virginia: American Institute of Aeronautics and Astronautics) ISBN 978-1-62410-704-7 URL `https://arc.aiaa.org/doi/10.2514/6.2023-3719`

[11] Morimoto M, Fukami K, Zhang K, Nair A G and Fukagata K 2021 Convolutional neural networks for fluid flow analysis: toward effective metamodeling and low dimensionalization *Theoretical and Computational Fluid Dynamics* **35** 633–658 ISSN 14322250

[12] Fukami K, Fukagata K and Taira K 2020 Assessment of supervised machine learning methods for fluid flows *Theoretical and Computational Fluid Dynamics* **34** 497–519 ISSN 14322250

[13] Immordino G, Da Ronch A and Righi M 2023 Deep–learning framework for aircraft aerodynamics prediction *AIAA AVIATION 2023 Forum* (American Institute of Aeronautics and Astronautics Inc, AIAA) (*Preprint* `https://arc.aiaa.org/doi/pdf/10.2514/6.2023-3846`) URL `https://arc.aiaa.org/doi/abs/10.2514/6.2023-3846`

[14] Sabater C, Stürmer P and Bekemeyer P 2022 Fast Predictions of Aircraft Aerodynamics Using Deep-Learning Techniques *AIAA Journal* **60** 5249–5261 ISSN 1533385X

[15] Bronstein M M, Bruna J, Cohen T and Veličković P 2021 Geometric Deep Learning Grids, Groups, Graphs, Geodesics, and Gauges Tech. rep. URL `https://arxiv.org/abs/2104.13478`

[16] Bronstein M M, Bruna J, Lecun Y, Szlam A and Vandergheynst P 2017 Geometric Deep Learning: Going beyond Euclidean data *IEEE Signal Processing Magazine* **34** 18–42 ISSN 10535888

[17] Ogoke F, Meidani K, Hashemi A and Farimani A B 2021 Graph convolutional networks applied to unstructured flow field data *Machine Learning: Science and Technology* **2** ISSN 26322153

[18] Baqué P, Remelli E, Fleuret F and Fua P 2018 Geodesic Convolutional Shape Optimization URL `http://arxiv.org/abs/1802.04016`

[19] Hines D and Bekemeyer P 2023 Graph neural networks for the prediction of aircraft surface pressure distributions *Aerospace Science and Technology* **137** ISSN 12709638

[20] Han X, Gao H, Pfaff T, Wang J X and Liu L P 2022 Predicting Physics in Mesh-reduced Space with Temporal Attention URL `http://arxiv.org/abs/2201.09113`

[21] Park K H, Jun S O, Baek S M, Cho M H, Yee K J and Lee D H 2013 Reduced-order model with an artificial neural network for aerostructural design optimization *Journal of Aircraft* **50** 1106–1116 ISSN 15333868

[22] Ribau M, Gonçalves N D, Ferrás L L and Afonso A M 2021 Flow structures identification through proper orthogonal decomposition: The flow around two distinct cylinders *Fluids* **6** ISSN 23115521

[23] Brunton S L, Noack B R and Koumoutsakos P 2020 Machine Learning for Fluid Mechanics *Annual Review of Fluid Mechanics* **52** 477–508 ISSN 00664189

[24] Saetta E, Tognaccini R and Iaccarino G 2023 Abbottae: An autoencoder for airfoil aerodynamics *AIAA AVIATION 2023 Forum* (American Institute of Aeronautics and Astronautics Inc, AIAA) (*Preprint* `https://arc.aiaa.org/doi/pdf/10.2514/6.2023-4364`) URL `https://arc.aiaa.org/doi/abs/10.2514/6.2023-4364`

[25] Liu Y, Ponce C, Brunton S L and Kutz J N 2023 Multiresolution convolutional autoencoders *Journal of Computational Physics* **474** ISSN 10902716

[26] Hinton G E and Salakhutdinov R R 2006 Reducing the dimensionality of data with neural networks *Science* **313** 504–507 URL `http://www.ncbi.nlm.nih.gov/sites/entrez?db=pubmed&uid=16873662&cmd=showdetailview&indexed=google`

[27] McCormick S F 1987 *Multigrid Methods* (Society for Industrial and Applied Mathematics) (*Preprint* `https://epubs.siam.org/doi/pdf/10.1137/1.9781611971057`) URL `https://epubs.siam.org/doi/abs/10.1137/1.9781611971057`

[28] Taylor N J, Gammon M and Vassberg J C 2016 The NASA Common Research Model: A Geometry-Handling Perspective *46th AIAA Fluid Dynamics Conference*

[29] Rivers M 2019 NASA Common Research Model: A History and Future Plans *AIAA Scitech 2019 Forum*

[30] Zhou J, Cui G, Zhang Z, Yang C, Liu Z, Wang L, Li C and Sun M 2018 Graph Neural Networks: A Review of Methods and Applications URL `https://arxiv.org/abs/1812.08434`

[31] Kipf T N and Welling M 2017 Semi-supervised classification with graph convolutional networks *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings* 1–14

[32] He K, Zhang X, Ren S and Sun J 2015 Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification URL `http://arxiv.org/abs/1502.01852`

[33] LeCun Y, Bottou L, Bengio Y and Haffner P 1998 Gradient-Based Learning Applied to Document Recognition *Proceedings of the IEEE* vol 86 pp 2278–2324 URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.7665`

[34] Smith W A 1990 Multigrid Solution of Transonic Flow on Unstructured Grids *Recent Advances and Applications in Computational Fluid Dynamics* ed Baysal O (Hanover: Proceedings of the ASME Winter Annual Meeting)

[35] Quaranta G, Masarati P and Mantegazza P 2005 A Conservative Mesh-Free Approach For Fluid-Structure Interface Problems *Int. Conf. on Computational Methods for Coupled Problems in Science and Engineering, Coupled Problems 2005 - ECCOMAS* ed CIMNE (Barcelona) pp 1–22

[36] Joldes G R, Chowdhury H A, Wittek A, Doyle B and Miller K 2015 Modified moving least squares with polynomial bases for scattered data approximation *Applied Mathematics and Computation* **266** 893–902 ISSN 0096-3003 URL `https://www.sciencedirect.com/science/article/pii/S0096300315007924`

[37] Rivers M B and Balakrishna S 2014 NASA Common Research Model Test Envelope Extension with Active Sting Damping at NTF *32nd AIAA Applied Aerodynamics Conference* (American Institute of Aeronautics and Astronautics Inc, AIAA) (*Preprint* `https://arc.aiaa.org/doi/pdf/10.2514/6.2014-3135`) URL `https://arc.aiaa.org/doi/abs/10.2514/6.2014-3135`

[38] Kingma D P and Ba J L 2015 Adam: A method for stochastic optimization *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* 1–15

[39] Bickel P and Doksum K 2015 *Mathematical Statistics: Basic Ideas and Selected Topics* 2nd ed (*Chapman & Hall/CRC Texts in Statistical Science* vol I) (CRC Press) ISBN 9781498723817 URL `https://books.google.co.uk/books?id=y5i9BwAAQBAJ`

Appendix

Appendix .1. Model Architecture & Optimisation Procedure

Table 1 reports the model architecture developed for the steady-state aerodynamic simulations. A diagram of the model architecture is reported in Figure 11 of the main article. Details of the GCN and MM layers in the encoder and the decoder blocks are provided. The PReLU activation function to introduce nonlinearities was added to each

GCN layer, except at the model output. The dimension of the tensor for each layer is indicated, arranged in $m \times n_{\text{mml}} \times c_l$, with m the batch size, n_{mml} the grid size from the corresponding MM level and c_l the number of layer output channels. The kernel size (number of learnable parameters) in each layer is also indicated.

Table 1: GCN-MM-AE model architecture for steady-state aerodynamic prediction of the CRM case. Refer to diagram in Figure 11 of the main article.

Layer	**Dimension**	**Operation**	**Kernel**
Input: $[M_\infty, \alpha_\infty]$, $\boldsymbol{x}_i$, $\hat{\boldsymbol{A}}_0$	$m \times 78829 \times 5$		
Encoder:			
GCN Enc0.1	$m \times 78829 \times 72$	Eq. (6)	$5 \times 72 + 72$
PReLU Enc0.1	$m \times 78829 \times 72$	Eq. (8)	72
GCN Enc0.2	$m \times 78829 \times 144$	Eq. (6)	$72 \times 144 + 144$
PReLU Enc0.2	$m \times 78829 \times 144$	Eq. (8)	144
MM $\hat{\boldsymbol{A}}_0 \to \hat{\boldsymbol{A}}_1$	$m \times 5000 \times 144$	Eq. (10)	
GCN Enc1.1	$m \times 5000 \times 144$	Eq. (6)	$144 \times 144 + 144$
PReLU Enc1.1	$m \times 5000 \times 144$	Eq. (8)	144
GCN Enc1.2	$m \times 5000 \times 288$	Eq. (6)	$144 \times 288 + 288$
PReLU Enc1.2	$m \times 5000 \times 288$	Eq. (8)	288
Decoder:			
GCN Dec1.1	$m \times 5000 \times 144$	Eq. (6)	$288 \times 144 + 144$
PReLU Dec1.1	$m \times 5000 \times 144$	Eq. (8)	144
GCN Dec1.2	$m \times 5000 \times 144$	Eq. (6)	$144 \times 144 + 144$
PReLU Dec1.2	$m \times 5000 \times 144$	Eq. (8)	144
MM $\hat{\boldsymbol{A}}_1 \to \hat{\boldsymbol{A}}_0$	$m \times 78829 \times 144$	Eq. (10)	
GCN Dec0.1	$m \times 78829 \times 72$	Eq. (6)	$144 \times 72 + 72$
PReLU Dec0.1	$m \times 78829 \times 72$	Eq. (8)	72
GCN Dec0.2	$m \times 78829 \times 72$	Eq. (6)	$72 \times 72 + 72$
PReLU Dec0.2	$m \times 78829 \times 72$	Eq. (6)	72
Repeat: $\mathbf{y}_i \to [\mathbf{y}_i, \mathbf{y}_i, \mathbf{y}_i, \mathbf{y}_i]$	$m \times 78829 \times 72 \times 4$		
GCN Dec0.3	$m \times 78829 \times 72 \times 4$	Eq. (6)	$72 \times 72 \times 4 + 72 \times 4$
PReLU Dec0.3	$m \times 78829 \times 72 \times 4$	Eq. (6)	72×4
GCN Dec0.4	$m \times 78829 \times 1 \times 4$	Eq. (6)	$72 \times 1 \times 4 + 1 \times 4$
Output: $[C_{P_i}, C_{\tau x i}, C_{\tau y_i}, C_{\tau z i}]$	$m \times 78829 \times 4$	Eq. (3)	

To complete the description of the model generation, Table 2 provides details of the optimisation strategy adopted to generate the ROM. In brief, we chose the Adam gradient-based algorithm [38] as optimiser and the mean squared error (MSE) as loss function [39]. The input data was standardised and normalised using the the field mean and standard deviation, respectively, which is a common practice for a more successful training outcome. The minimisation history of the mean squared error is shown in

Table 2: Training strategy for the generation of the steady-state GCN-MM-AE model.

Parameter	**Value**
Trainable parameters	$174,460$
Dataset samples	70
Training set (58%)	40
Batch size	1
Training epochs	800
Loss function	MSE
Optimiser	Adam
Start learning rate	0.0009
Learning rate decay	0.333 / 300 epochs
GPU machine	`NVIDIA GeForce RTX` 2060
Training time	2.5 h

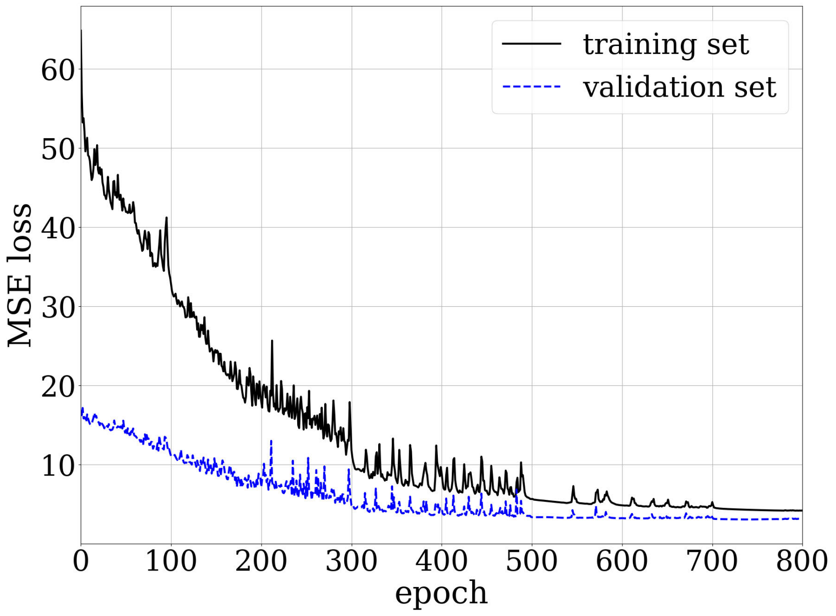


Figure 1: Mean squared error (loss function) history during model weights optimisation for the training and validation sets.

Figure 1 for both the training and validations sets. The final MSE values between sets are similar, which indicates that the model is not over-fitting.

Appendix .2. Multi-Mesh Interpolation Comparison

Various interpolation methods can be thought for the coarsening and recovery steps of Multi-Mesh scheme. To demonstrate the adequacy of our proposed WMLS method, the multi-grid scheme by Smith *et al* [34], adapted to our surface-mesh task, was considered

Table 3: Prediction error for each MM interpolation method.

	ε_{C_L} [%]			ε_{C_D} [%]			$\varepsilon_{C_{M_y}}$ [%]		
MM cycle	mean	std dev	worst	mean	std dev	worst	mean	std dev	worst
WMLS	0.8	0.7	3.3	1.0	1.1	5.2	3.5	5.0	24.3
MG *Smith*	0.8	0.5	2.2	1.2	1.1	4.3	4.6	8.4	48.9

as an alternative option. In brief, Smith *et al* adopted a volume-averaged solution of the fine cells embedded to each coarse cell, for the reduction step; and a direct re-casting of the solution on the fine grid from each associated coarse cell, for the recovery operation. Table 3 reports the summary of the prediction error obtained with each interpolation method. Both options were found similar with regards to the prediction of the C_L and C_D. However, our WMLS approach provided marginally improved C_{M_y} results.

As a result of the interpolation, differences on the solution fields can be expected. Therefore, we analysed the solutions predicted by each MM method. Figure 2 illustrates a comparison on the predicted C_P (upper panels) and the error against the CFD reference (lower panels) on the worst validation case (sample f in Figure 12 of the main text). Our WMLS is in the middle panels while Smith's alternative is in the right panels. We observe a noisier output solution with Smith's method, therefore, less physically representative. This issue could be solved with additional GCN blocks after the recovery step, but at the expense of a larger computational burden. By contrast, smoother solutions are obtained with our WMLS scheme. Nonetheless, the error is visibly larger with Smith's method in this particular sample. These results demonstrate our proposed WMLS scheme for the MM cycle is an appropriate choice for the transonic aerodynamics task.

Appendix .3. Sensitivity to Model Architecture Hyperparameters

The influence from the various key hyperparameters of the proposed model architecture is analysed here. In particular, we are interested in assessing the sensitivity of the model performance to the training set size, the number of model weights or the multi-mesh cycle reduction.

Sensitivity to MM compression rate: Table 4 reports the prediction error summary for various mesh compression rates adopted in the MM cycle. Different models were generated for mesh coarsenings to 10,000, 5,000 and 2,500 grid points, respectively. A compression ratio larger than 30 was not pursued. The final mesh coarsening choice adopted in the Results section is highlighted in bold. We observe that the prediction error statistics are similar among the various MM compressions. Despite the prediction error is expected to worsen with coarser meshes, the results suggest that the various GCN blocks are able to compensate for this. The model performance remains acceptable even on significantly large mesh reductions.

Table 4: Prediction error comparison for various MM compression ratios on the CRM validation dataset.

Coarse mesh size	ε_{C_L} [%]			ε_{C_D} [%]			$\varepsilon_{C_{M_y}}$ [%]		
(Compress. ratio)	mean	std dev	worst	mean	std dev	worst	mean	std dev	worst
10,000 (7.9)	0.5	0.4	1.7	0.8	0.9	3.3	4.7	6.2	30.7
5,000 (15.8)	0.8	0.7	3.3	1.0	1.1	5.2	3.5	5.0	24.3
2,500 (31.6)	0.7	0.5	2.0	0.6	0.6	2.6	3.4	4.2	19.1

Table 5: Prediction error comparison for different MM cycle levels on the CRM validation dataset.

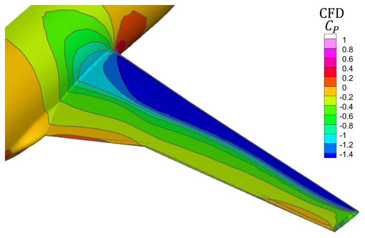
	ε_{C_L} [%]			ε_{C_D} [%]			$\varepsilon_{C_{M_y}}$ [%]		
MM cycle	mean	std dev	worst	mean	std dev	worst	mean	std dev	worst
2 levels	0.8	0.7	3.3	1.0	1.1	5.2	3.5	5.0	24.3
3 levels	0.8	0.7	2.6	1.1	0.8	3.4	3.9	4.3	17.1
4 levels	0.8	0.9	4.1	1.3	1.8	9.1	3.7	7.0	38.5

Table 6: Prediction error comparison for different training dataset sizes on the CRM validation dataset.

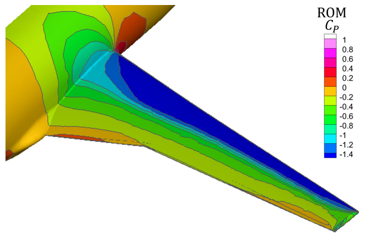
	ε_{C_L} [%]			ε_{C_D} [%]			$\varepsilon_{C_{M_y}}$ [%]		
Training set	mean	std dev	worst	mean	std dev	worst	mean	std dev	worst
40 samples	0.8	0.7	3.3	1.0	1.1	5.2	3.5	5.0	24.3
20 samples	1.3	1.2	5.6	2.0	2.5	13.9	5.8	7.3	41.6

Table 7: Prediction error comparison for different model sizes on the CRM validation dataset.

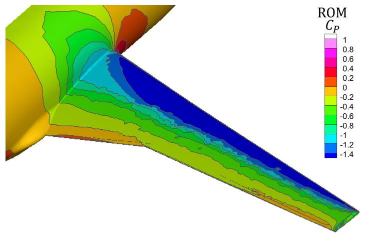
	ε_{C_L} [%]			ε_{C_D} [%]			$\varepsilon_{C_{M_y}}$ [%]		
Model weights	mean	std dev	worst	mean	std dev	worst	mean	std dev	worst
174,460	0.8	0.7	3.3	1.0	1.1	5.2	3.5	5.0	24.3
98,770	0.6	0.5	2.1	0.5	0.3	1.6	4.7	5.6	26.9
44,464	1.0	1.0	5.0	1.0	0.9	5.0	5.6	6.3	34.1



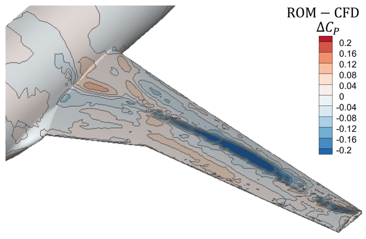
(a) C_P CFD reference.



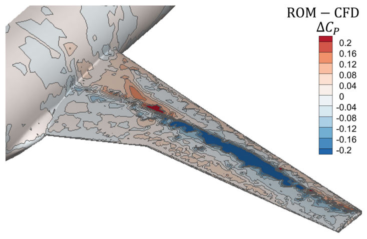
(b) Predicted C_P, MM WMLS.



(c) Predicted C_P, MM *Smith*.



(e) C_P error, MM WMLS.



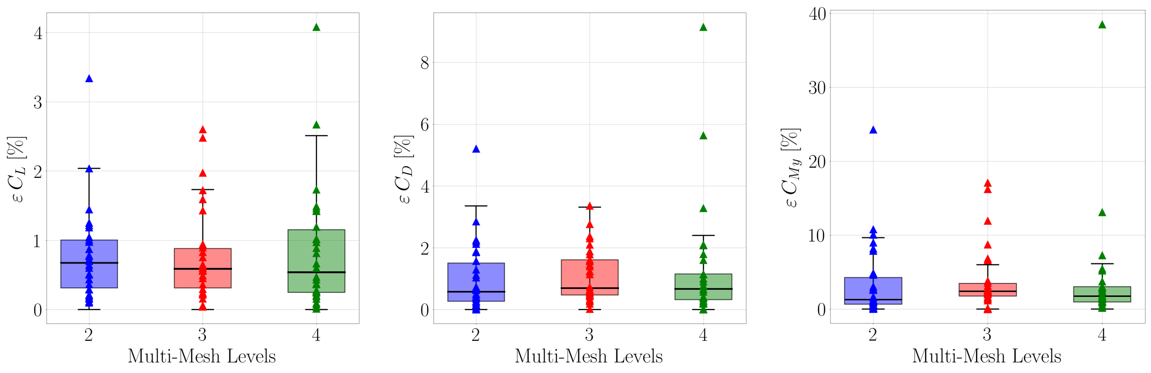
(f) C_P error, MM *Smith*.

Figure 2: C_P comparison between the two interpolation methods in the MM cycle against CFD reference at sample f in Figure 12.

Sensitivity to MM cycle levels: The sensitivity to the number of MM reduction layers is assessed in Table 5. Results are reported for MM cycles of 2 (final choice highlighted in bold), 3 and 4 levels. The 3-level MM cycle was constructed by first reducing to 20,000 nodes and subsequently reducing to 5,000, assigning the same node count as for the 2-level case. The 4-level scheme was achieved by reducing to 40,000, 20,000 and 5,000. GCN blocks were embedded between mesh levels, for a total of 174,460, 352,156 and 436,252 model weights, respectively. The statistics are similar among the various MM cycles. By contrast, the computational cost involved in the 3-level and the 4-level schemes was found, respectively, 32% and 81% larger compared to the 2-level implementation. This motivated our choice of the 2-level reduction as a more efficient implementation in terms of model memory requirements.

A quantification of the prediction confidence for each multi-mesh cycle is illustrated in Figure 3. Inter-quartile range plots for the three different MM levels and the various performance metrics, C_L error, C_D error and C_{M_y} error. The uncertainty intervals are based on predictions of the validation dataset. The error for each validation sample is also illustrated in triangles. We observe that the 3-level option was found with the marginally lowest variability. Nevertheless, the uncertainty is similar for the various MM levels, which provides confidence on the final choice of the 2-level MM cycle.

Sensitivity to training dataset size: We now analyse how the prediction performance is affected with a smaller training dataset. In particular, the number of training samples was halved, i.e. using 20 samples to generate the model as opposed to the original 40. Minimising the amount of data requirements is interesting to reduce the computiional



(a) C_L error. (b) C_D error. (c) C_{M_y} error.

Figure 3: Box-and-whisker plots for the prediction error among the various MM levels. The error of each validation sample is also shown in triangles.

cost involved with running the CFD simulations. The prediction error summary for the various load coefficients is reported in Table 6, with the original dataset highlighted in bold. A degradation of the performance is observed with the reduced training set. However, the error values remain still reasonably good considering the small number of preliminary samples to generate the model.

Sensitivity to model size: The performance of the model for smaller number of weights is assessed here. Evaluating different model sizes is useful to identify potential over-fitting issues and reduce GPU memory requirements. In this study, the model size is reduced by halving the number of weights in two successive steps, for a total of 98,770 and 44,464 parameters, respectively. The results are reported in Table 7. A gradual degradation of the performance is observed with reducing the size of the model. Nevertheless, the degradation is fairly low. We conclude that there is no data over-fitting with the largest model. And the smallest model still showcases good capacity to learn the physics of the system adequately.