Aerospace Science and Technology 160 (2025) 110037

Contents lists available at ScienceDirect

Aerospace Science and Technology

journal homepage: www.elsevier.com/locate/aescte

Augmenting mesh-based data-driven models with physics gradients

David Massegur *, Andrea Da Ronch

Faculty of Engineering and Physical Sciences, University of Southampton, Southampton, SO16 7QF, United Kingdom

ARTICLE INFO

Communicated by Mehdi Ghoreyshi

Keywords:
Geometric deep learning
Physics informed
Gradient guided
Computational fluid dynamics
Transonic aerodynamics
Graph convolutional network
Multi mesh
Autoencoder
Hybrid approach
Common research model

ABSTRACT

Deep learning technologies are increasingly used in various applications, with significant potential in aerospace for reduced-order modelling due to their ability to handle nonlinear systems. The effectiveness of data-driven methods relies on the adequacy and volume of training data, which poses a challenge in a design environment. To address this, physics-informed machine learning, which integrates physics knowledge into data-driven frameworks, has emerged as a promising solution. Directly applying physics terms to aircraft surfaces is complex, so this study utilizes solution gradients to effectively capture flow features. We introduce a hybrid framework that combines geometric deep learning with gradient terms, building on a previous data-driven approach for aerodynamic modelling on large-scale, three-dimensional unstructured grids. We evaluated various hybrid schemes to enhance prediction accuracy. Two gradient-enhanced approaches were found to outperform the purely data-driven model: the first integrates output differentiation into the training loss, achieving the highest accuracy at an increased training cost; the second employs a masking technique to weight regions with large gradients, providing a reasonable accuracy improvement at a lower training cost. This study focuses on predicting distributed aerodynamic loads around the NASA Common Research Model wing/body configuration under various transonic flight conditions. Our findings show that incorporating gradient information into deep learning models significantly improves the accuracy of the predictions and can compensate for a smaller dataset without compromising accuracy. Furthermore, the approaches proposed herein are directly applicable to any problem with discretised spatial domain.

1. Introduction

The use of three-dimensional, unstructured grids for aerodynamic analysis of full-scale aircraft, is an established process in industry. Grids often consist of tens of million of grid points. In the case of aircraft aerodynamics across the flight envelope, for example, high-fidelity computational fluid dynamics (CFD) are typically solved around these grids on high-performance computing facilities [1,31]. Each flight condition is important for design and sizing of various aircraft components, and this is regulated by airworthiness authorities [8]. Aircraft spend most of the flight time in the transonic regime where, for example, the target cruise drag is set during the design process [17]. Running aerodynamic analyses at hundreds of flight points to establish performance characteristics is far from routine due to the increased computing costs. The work presented herein is motivated by the desire to have a solution to this long-standing problem, which is common in many other engineering fields, from the automotive industry and racing cars to the wind energy.

Reduced order modelling (ROM) and model order reduction are techniques for reducing the complexity of a full-order, high-fidelity model providing a trade-off between cost/complexity and prediction accuracy. The main features of a ROM [6,24] are: a) reduce the size and complexity of the computational model; b) retain dynamic nonlinearities; and c) recover the full-order dynamics of the system. There are two general classes of ROMs: intrusive approaches that manipulate the governing equations; and non-intrusive approaches that only require available data. The work here presented exploits the latter category, making progress in the deployment of deep learning algorithms.

Common nonlinear ROM approaches, including Kriging [10], are designed for the prediction of (scalar) aerodynamic forces. By contrast, modelling of 3D aerodynamic fields is more interesting for design optimisation purposes. This motivates the adoption of deep learning. A fundamental problem of deep learning algorithms for the generation of an aerodynamic ROM is that common neural network (NN) architectures are inadequate when confronted with the large-scale, unstructured grids of full-scale simulations [12]. Fully-connected NN [37,36]

* Corresponding author.
E-mail address: David.Massegur-Sampietro@southampton.ac.uk (D. Massegur).

https://doi.org/10.1016/j.ast.2025.110037
Received 12 November 2024; Received in revised form 5 February 2025; Accepted 6 February 2025

Available online 10 February 2025

are known for poor scalability, and convolutional NN [7,28] are only applicable to Euclidean (regular) domains. Geometric deep learning (GDL) is an umbrella of NN approaches designed for non-Euclidean domains [4]. GDL methods leverage the mesh connectivity to execute message-passing operations across the grid [5]. To address the issue of scalability, a dimensionality reduction technique to compress the large spatial domain without general loss of information is convenient [21,20]. Neural-network dimensionality reduction approaches, known as autoencoders [13], compared to the classical Proper Orthogonal Decomposition (POD) method, provide the added benefit of non-linear domain compressions and subsequent recovery [23].

This work builds on previous studies [25,26] that introduced a graph-convolutional multi-mesh autoencoder framework, referred to as GCN-MM-AE, demonstrated on a transonic aircraft. Both steady-state and unsteady conditions have been considered. Despite a good agreement with reference data, results highlighted a persistent problem with data-driven models: the inability to perform well in new experiments away from the training samples, a property known as generalisation [29,38]. To capture the physics correctly, data-driven NNs remain strongly reliant on sufficient training data distributed across the entire design space. However, the available number of training samples, in the context of costly engineering analyses, such as CFD for aerodynamics, is strongly limited by computing power. A new paradigm is sought to improve the inference performance of data-driven models when the data available is limited.

In contrast to purely data-driven approaches, physics-informed machine learning (Phi-ML) [41,43] integrates governing equations directly into the model architecture. Phi-ML algorithms utilize the residuals of partial differential equations as loss functions for model parameter optimization. Despite their potential, physics-informed NNs (PINNs) face significant challenges in fluid dynamic applications [33,2], being primarily limited to two-dimensional laminar flows [1] and exhibiting poor scalability by adopting fully-connected NNs, which restricts their applicability from practical aerodynamic problems. Works adopting Convolutional NNs with fully embedding of the physics equations exist but these required the non-trivial transformation of the irregular domain into a cartesian grid, limiting the complexity of the domain shape [9,14].

Furthermore, in the case of fluid dynamics, the governing equations apply within the fluid volume. By contrast, to maximise computational efficiency, ROM predictions are generally tailored to address surface predictions, where the physics equations are not applicable. For cases with large spatial domains (e.g. aircraft design), complex physics equations (turbulent Navier-Stokes equations [31]) and the high costs associated with training data from computational fluid dynamics (CFD) simulations, a hybrid approach combining data-driven and physics-informed methods is a potential alternative. Embedding partial physics knowledge into data-driven algorithms can enhance the performance of predictive frameworks [42,44], particularly in under-sampled regions of the design space. For example, multi-scale methods to enhance the model prediction accuracy of distributed fields and scalar loads by constraining the dedicated networks to shared latent features were proposed by [45,30]. However, these works dealt with 2D configurations and there is no certainty of a direct correlation between the predicted aerodynamic fields and the predicted forces.

The challenge we address in this work is the prediction of the surface flow solution, including the pressure and skin friction coefficient, around a three-dimensional aircraft configuration at any flight point within a pre-defined flight envelope. Our conjecture to improve the predictive capability of the model is to leverage on the surface gradients of the solution field, $\nabla(\cdot) = \left[\frac{\partial(\cdot)}{\partial x}, \frac{\partial(\cdot)}{\partial y}, \frac{\partial(\cdot)}{\partial z}\right]$. The gradient terms are a convenient choice because they are:

- Computationally cost-effective, in the sense that differentiation of distributed quantities is easier than solving partial differential equations.
- Useful to identify flow physics characterised by a rapid variation of the flow quantities, such as shock waves, boundary-layer separation and flow acceleration at the leading edge.

The aim of this work is to explore the feasibility of an hybrid predictive framework whereby a purely data-driven approach is enhanced with gradients of the surface field. Starting from the GCN-MM-AE model, we answer the following research question: what is the best route to embed gradients information into the model architecture? Various approaches can be considered to embed gradient terms into the model architecture. For example, a mixed-gradient-error-based loss function was incorporated in [39] to enhance a CNN-based model for predicting the Mach field around 2D nacelles. In that study, Sobel filters were applied to flow-field data to emphasize edges, a technique common in image processing. However, while Sobel filters are effective for highlighting large gradients in structured domains, they are unsuitable for irregular grids, necessitating alternative methods to computing gradients. Consequently, we resort to spatial differentiation of the solution fields. In the following sections, we propose five distinct approaches to introduce gradients into the model, enriching the predictive model and enhancing the understanding of the underlying physics particularly in complex and under-sampled regions. The performance assessment is carried out on the NASA Common Research Model (CRM) [25,15] at transonic conditions. It is worth noting that, despite the test case addressing aerodynamic predictions, no domain-specific knowledge is used to derive the gradients. As a result, the methodology presented herein is adequate for any other field (in engineering and beyond) involving mesh-based simulations.

The paper continues in Section 2 with a description of the methodology and the different routes to embed gradient terms. A description of the test case providing a justification for the proposed hybrid approach is given in Section 3. Section 4 contains an analysis of the results for the various schemes. Conclusions are given in Section 5.

2. Methodology

2.1. Graph-convolutional multi-mesh autoencoder overview

First, we provide a brief description of the methodology for the entirely data-driven architecture, which serves as reference for the hybrid approach implementation. A neural-network based architecture is sought that generates a map between the vector of input conditions $\boldsymbol{s}$ and the fields to predict $\boldsymbol{Y}_i$ on the nodes i of a surface mesh $\mathcal{S}_\mathrm{n}$:

$$\boldsymbol{Y}_i = f_\mathrm{NN}\left(\boldsymbol{s}, \boldsymbol{x}_i, \boldsymbol{\Theta}\right) \qquad \forall\, i \in \mathcal{S}_\mathrm{n} \tag{1}$$

with the $[x, y, z]$ coordinates $\boldsymbol{x}_i \in \mathbb{R}^{n_n \times 3}$, where n_n the number of mesh nodes, being also included as inputs to embed the spatial mapping. The optimal model parameters $\boldsymbol{\Theta}$ are sought that minimise the mean squared error (MSE) [3] between the predicted and the reference solutions fields:

$$\min_{\boldsymbol{\Theta}} \; \mathcal{L}_0 = ||\boldsymbol{Y}_\mathrm{NN} - \boldsymbol{Y}_\mathrm{CFD}||_2^2 \tag{2}$$

To leverage geometric deep learning [4] on unstructured CFD meshes, the surface mesh is represented as a graph. Target fields $\boldsymbol{Y}_i$ and position coordinates $\boldsymbol{x}_i$ are assigned on the graph nodes; and user-defined weights e_{ij}, on the edges of connected nodes. Fig. 1 illustrates a graph representation from the triangular mesh of the CRM model, including node-based features, coordinates and edge weights.

From the family of GDL methods, we adopted the GCN operator [19]:

$$g(\boldsymbol{y}) = h\left(\boldsymbol{\theta}^T \hat{\boldsymbol{D}}^{-\frac{1}{2}} \hat{\boldsymbol{A}} \hat{\boldsymbol{D}}^{-\frac{1}{2}} \boldsymbol{y} + b\right) \tag{3}$$

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 $\mathcal{S}$. $\hat{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$, with $\boldsymbol{A} = e_{ij}$, is the adjacency matrix and $\hat{\boldsymbol{D}} = \mathrm{diag}\left(\sum_{j \neq i} e_{ij} + 1\right)$, the diagonal degree matrix, i.e. the sum of the edge weights connected to target node

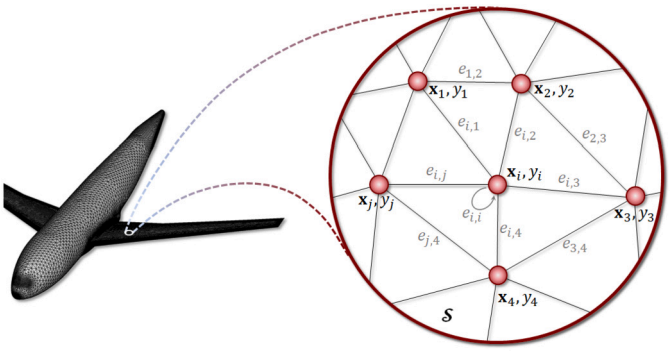


Fig. 1. Mesh represented as a graph with node features and edge weights.

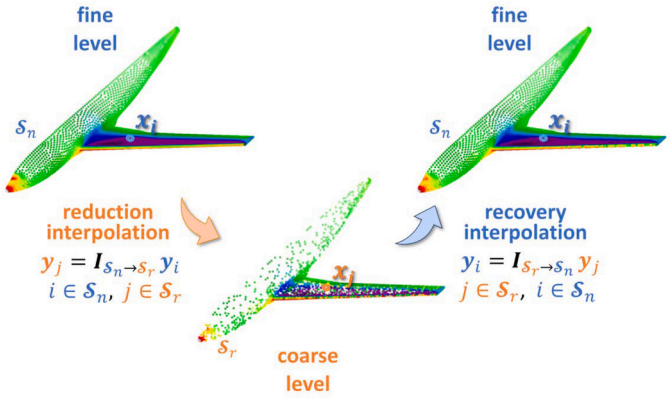


Fig. 2. Multi-mesh cycle, demonstrating interpolation between mesh levels of different resolution.

i. It is convenient to arrange the connectivity matrix in sparse form $\hat{\boldsymbol{A}} \in \mathbb{R}^{n_e \times 3}$ containing, for each edge, the indices of the connected node pairs i, j and the respective weight value e_{ij}. The PReLU function [11] was chosen as nonlinear activation: $h(x) = \{x \text{ for } x >= 0,\ \beta x \text{ otherwise}\}$, with β a learnable parameter.

To address large spatial domains, an autoencoder approach for dimensionality was adopted [13]. To this aim, the GCN layers were embedded in a multi-mesh (MM) scheme, resembling the multi-grid method to solve partial differential equations [27]. The MM cycle coarsens the mesh while extracting crucial features and subsequently refines the latent states back onto the original mesh, as illustrated in Fig. 2. To transfer the information between mesh levels, a weighted moving least squares interpolation was implemented [32,16]. Refer to Refs. [25,26] for complete description of this developed data-driven framework formulation.

The reference data-driven GCN-MM-AE architecture for steady aerodynamic predictions is illustrated in Fig. 3. In the encoder, the input vector is fed through a GCN block, followed by the coarsening interpolation of the MM cycle and another GCN block. It follows the decoder, consisting of a GCN block, then the refining step of the MM and a GCN block acting on the refined mesh. Last, the network ramifies into separate blocks for each field quantity to predict.

The methodology just described corresponds to the regular data-driven (DD) approach, to which the gradient-guided (GG) contribution is embedded.

2.2. Gradient-guided schemes

Five different hybrid gradient-guided data-driven approaches were investigated, to compare and identify the best suited hybrid scheme for aerodynamic field predictions.

2.2.1. Gradients as target predictions

The gradients of the target fields are also included as predicted quantities alongside the respective fields. Therefore, the NN-based model f_{NN} predicts as outputs both the target fields and the gradients:

$$[\boldsymbol{Y}_i, \nabla \boldsymbol{Y}_i]_{\text{NN}} = f_{\text{NN}_1}\left(\boldsymbol{s}, \boldsymbol{x}_i, \boldsymbol{\Theta}\right) \qquad \forall i \in \mathcal{S}_\text{n} \tag{4}$$

The model parameters $\boldsymbol{\Theta}$ are consequently optimised with minimisation of the loss against the CFD solution and gradient data:

$$\min_{\boldsymbol{\Theta}} \mathcal{L}_1 = ||[\boldsymbol{Y}, \nabla \boldsymbol{Y}]_{\text{NN}} - [\boldsymbol{Y}, \nabla \boldsymbol{Y}]_{\text{CFD}}||_2^2 \tag{5}$$

This first approach is illustrated in Fig. 4 Panel (a). The implementation remains identical to the data-driven framework in Fig. 3 but with additional output quantities. The idea here is leveraging more information in terms of outputs to assist the model towards a more physical solution.

2.2.2. Model output gradient loss

With the output quantities $\boldsymbol{Y}_i$ from the NN model, the respective surface gradients are subsequently computed by spatial differentiation [22]:

$$\boldsymbol{Y}_{i\ \text{NN}} = f_{\text{NN}_2}\left(\boldsymbol{s}, \boldsymbol{x}_i, \boldsymbol{\Theta}\right) \xrightarrow{\nabla_{x,y,z}} \nabla \boldsymbol{Y}_{i\ \text{NN}} \qquad \forall i \in \mathcal{S}_\text{n} \tag{6}$$

The error between these computed gradients from the model predictions and the ground-truth gradients from CFD is incorporated as a second loss term for the model optimisation. As a result, the model parameters are optimised with this composed loss function:

$$\min_{\boldsymbol{\Theta}} \mathcal{L}_2 = ||\boldsymbol{Y}_{\text{NN}} - \boldsymbol{Y}_{\text{CFD}}||_2^2 + \lambda_2 ||\nabla \boldsymbol{Y}_{\text{NN}} - \nabla \boldsymbol{Y}_{\text{CFD}}||_2^2 \tag{7}$$

where λ_2 is a lagrange multiplier to calibrate the contribution between the DD loss, related to the target fields, and the gradient loss following the differentiation operation. Note that the spatial differentiation implementation must be compatible with the back-propagation execution [35] from the gradient loss for the model parameter optimisation. The complete gradient-loss framework, now featuring separate DD and GG embeddings, is illustrated in Fig. 4 Panel (b). The complete process is only required during model training. To execute new predictions, the gradient-guided block is not required but just the usual data-driven block is called.

2.2.3. Gradients as model latent states

The NN-based model is imposed to compute the gradients as latent states at the end of the decoder, before the architecture bifurcates for each target quantity. As a result, the predicted fields are constrained by the solution gradients being predicted at an interim stage:

$$\begin{aligned} \nabla \boldsymbol{Y}_{i\ \text{LS}} &= f_{\text{NN}_{3,1}}\left(\boldsymbol{s}, \boldsymbol{x}_i, \boldsymbol{\Theta}_{\text{AE}}\right) \ \rightarrow\ \boldsymbol{Y}_{i\ \text{NN}} \\ &= f_{\text{NN}_{3,2}}\left(\boldsymbol{s}, \boldsymbol{x}_i, \nabla \boldsymbol{Y}_{i\ \text{LS}}, \boldsymbol{\Theta}_{\text{branch}}\right) \quad \forall i \in \mathcal{S}_\text{n} \end{aligned} \tag{8}$$

The autoencoder parameters, i.e. before the model bifurcates to each quantity in Fig. 3, are optimised with the composed loss function, using a lagrange multiplier for calibration between the two terms. By contrast, the parameters in the final branches are trained with the single data-driven loss relevant to the target fields:

$$\begin{cases} \min_{\boldsymbol{\Theta}_{\text{AE}}} \mathcal{L}_{3,1} = ||\boldsymbol{Y}_{\text{NN}} - \boldsymbol{Y}_{\text{CFD}}||_2^2 + \lambda_3 ||\boldsymbol{Y}_{\text{LS}} - \nabla \boldsymbol{Y}_{\text{CFD}}||_2^2 \\ \min_{\boldsymbol{\Theta}_{\text{branch}}} \mathcal{L}_{3,2} = ||\boldsymbol{Y}_{\text{NN}} - \boldsymbol{Y}_{\text{CFD}}||_2^2 \end{cases} \tag{9}$$

This gradient-guided latent-state implementation is shown in Fig. 4 Panel (c). The idea is to embed the gradients into the latent states of the model to obtain a more physical inference system.

2.2.4. Gradient weighted model output loss

Reducing the prediction error where the solution gradients are larger is key to capturing the flow features. To achieve this, the idea is to penalise the loss further in the regions of the domain with larger variation

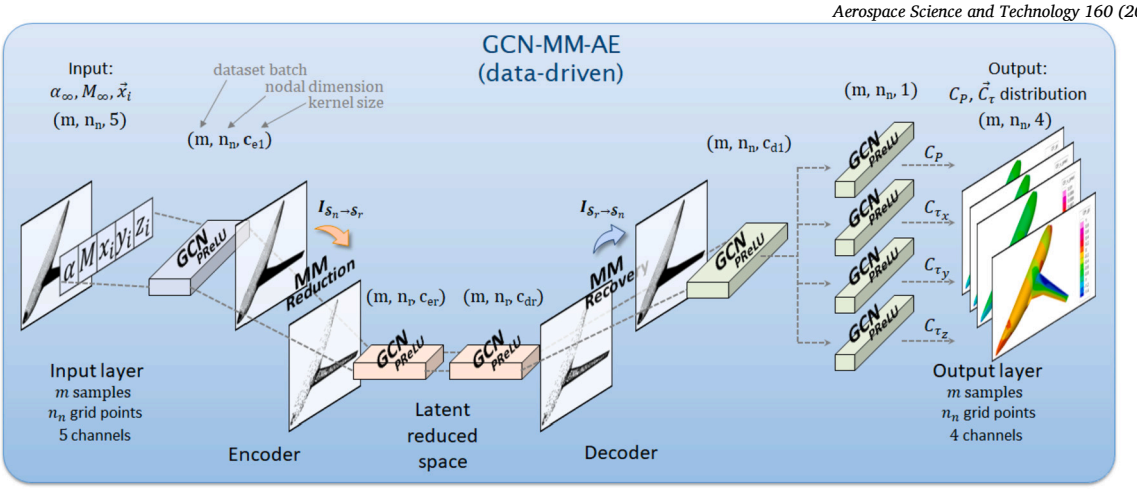


Fig. 3. Schematic of the data-driven steady-state GCN-MM-AE model architecture, introduced in previous work [25] used as baseline configuration for the gradient-guided schemes.

of the solution. We use the reference gradients to weight the target loss during the optimisation of the model. This scheme is illustrated in Fig. 5 Panel (a), where the Hadamard product is used to scale the loss by the gradient value. The model loss is therefore penalised (magnified) where the solution presents larger variation. Note that we chose as weights the derivative in the streamwise direction $\frac{\partial(\cdot)}{\partial x}$ only, as this is the most interesting of the three components from a physical standpoint. Alternatively, weighting for the remaining two derivatives is also possible through a linear combination. This GG loss is added to the original DD loss via a lagrange multiplier:

$$\min_{\boldsymbol{\Theta}} \mathcal{L}_4 = ||\boldsymbol{Y}_{\mathrm{NN}} - \boldsymbol{Y}_{\mathrm{CFD}}||_2^2 + \lambda_4 \left|\left| \frac{\partial(\boldsymbol{Y}_{\mathrm{CFD}})}{\partial x} \odot \left(\boldsymbol{Y}_{\mathrm{NN}} - \boldsymbol{Y}_{\mathrm{CFD}}\right) \right|\right|_2^2 \tag{10}$$

2.2.5. Gradient masked model output loss

To emphasise the regions with larger solution variation, a mask filter is applied to select only the grid points that exceed a fixed derivative threshold or excluded otherwise. The masked loss is added to the main loss again with a lagrange multiplier for calibration, useful to further penalise the mesh nodes with derivative values above a chosen limit:

$$\min_{\boldsymbol{\Theta}} \mathcal{L}_5 = ||\boldsymbol{Y}_{\mathrm{NN}} - \boldsymbol{Y}_{\mathrm{CFD}}||_2^2 + \lambda_5 ||\boldsymbol{M} \odot \left(\boldsymbol{Y}_{\mathrm{NN}} - \boldsymbol{Y}_{\mathrm{CFD}}\right)||_2^2 \tag{11}$$

where the mask matrix is defined as:

$$\boldsymbol{M} = \begin{cases} 1 & \text{if } |\nabla \boldsymbol{Y}_{\mathrm{CFD}}| \geq \delta_5 \\ 0 & \text{else} \end{cases} \tag{12}$$

with δ_5 a fixed threshold, which is manually calibrated to encapsulate the regions with large solution gradients. The gradient-guided masking process is shown in Fig. 5 Panel (b).

2.3. Additional methodology remarks

The proposed hybrid schemes differ by concept and complexity. This motivates the aim to leverage the various embeddings and assess their ability to assist the model towards understanding the physics better. Note also that these schemes are not exclusive but they are actually combinable together. As a result, and to the best of our knowledge, our work is novel on these fronts:

1. Development of a cost-effective hybrid data-driven and physics-guided approach to improve the machine-learning modelling performance of aerodynamics systems.
2. Embedding of the proposed gradient-guided approach into a geometric deep learning based framework for large and unstructured domains.
3. Investigation of distinct schemes to embed physics knowledge, leveraging the solution gradients, into the geometric deep learning based framework for direct prediction of the aerodynamic solution fields on the aircraft surfaces.

The various neural-network frameworks were implemented using `PyTorch 1.13`,[1] an optimised deep-learning library in `Python`, and `PyTorch Geometric`,[2] a user-friendly graph neural-network library built upon `PyTorch`. Regarding the solution gradient computations, the `MeshGradientPy`[3] package was adopted because of being embeddable with `PyTorch`'s back-propagation scheme. This package's gradient implementation is based on the formulation from [22]. The widely used `PyVista`[4] package was preferred for post-processing of the gradients.

3. NASA common research model

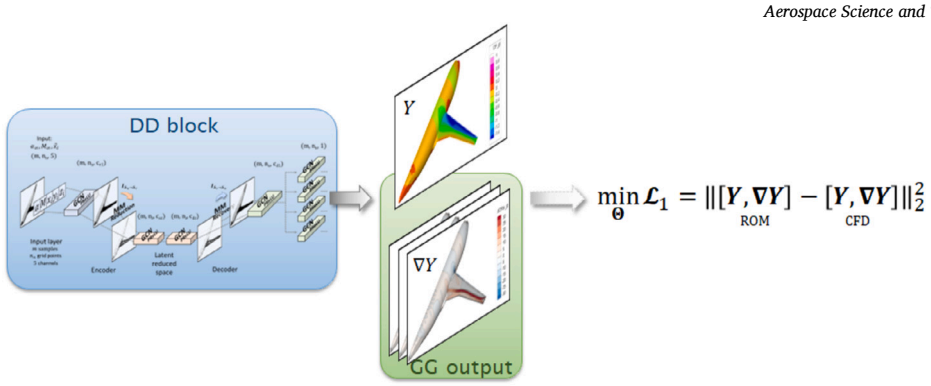
We adopted a relevant problem in the aerodynamics field as test case. Nevertheless, the advantage of our proposed gradient-based approaches is that they are directly applicable to any other discipline, independently of the underlying physics, wherein the spatial domain is discretised as a mesh.

3.1. Reference CFD dataset

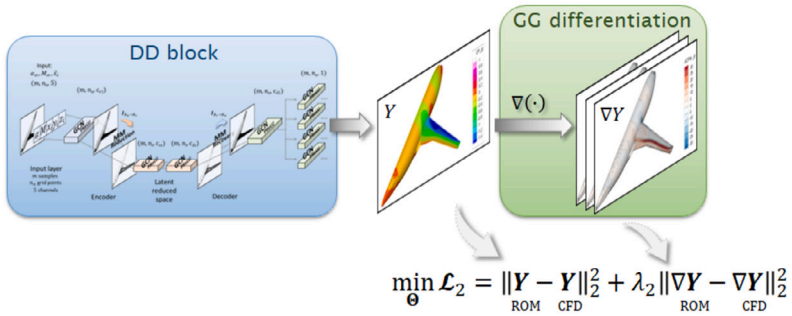
To demonstrate our models we used the NASA Common Research Model (CRM) [40,34], which consists of the wing/body model illustrated in Fig. 6 Panel (a). Reference geometric chord is $c = 0.1412$ m, surface area $S = 0.0727$ m^2 and the origin for moment calculations $x_{ea}/c = 0.5049$. We address steady-state prediction of the pressure and wall shear stress coefficients on the nodes of the surface mesh, $\boldsymbol{Y}_i = [C_P, C_{\tau_x}, C_{\tau_y}, C_{\tau_z}]_i \in \mathbb{R}^{n_n \times 4}$, given the mesh node coordinates $\boldsymbol{x}_i$ and the envelope of Mach numbers $M_\infty \in [0.70, 0.84]$ and angles of attack $\alpha_\infty \in [0.0, 5.0]$, at constant Reynolds number $Re = 5 \cdot 10^6$ and freestream temperature $T_\infty = 311$ K. We used 70 steady-state RANS CFD solutions from a companion paper [25] to sample the envelope, shown in Fig. 6 Panel (b), of which 40 were used to train the models (circles with black labels) and 30 left for final testing (triangles with red labels).

The CFD model involved 50 million cells, of which 74k on the walls, shown in Fig. 6 Panel (a), one-equation Spalart-Allmaras RANS turbulence model and the `SU2` solver was used. Full details on the generation

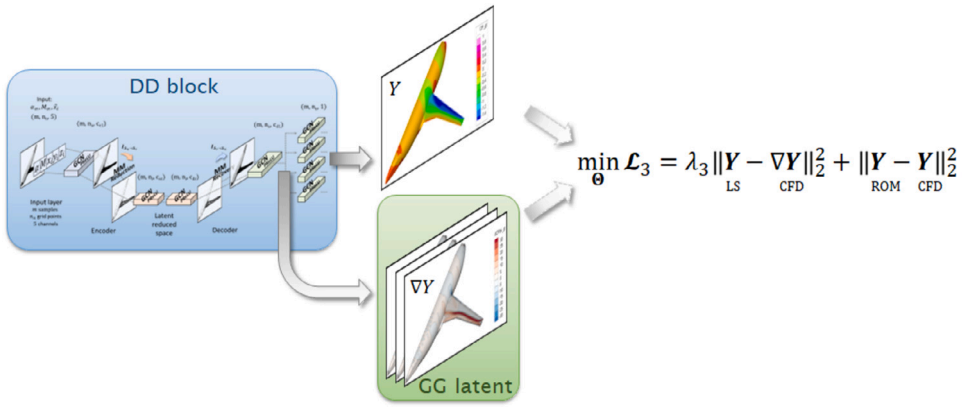
[1] https://pytorch.org/.
[2] https://pytorch-geometric.readthedocs.io/en/latest/.
[3] https://github.com/DonsetPG/MeshGradientPy.
[4] https://docs.pyvista.org/.



(a) Approach 1: gradients predicted by the data-driven model, together with the target quantities.



(b) Approach 2: gradients computed via differentiation of the model predictions and added to the optimisation loss.



(c) Approach 3: solution gradients imposed as latent states at the autoencoder output before the model bifurcation to each quantity field.

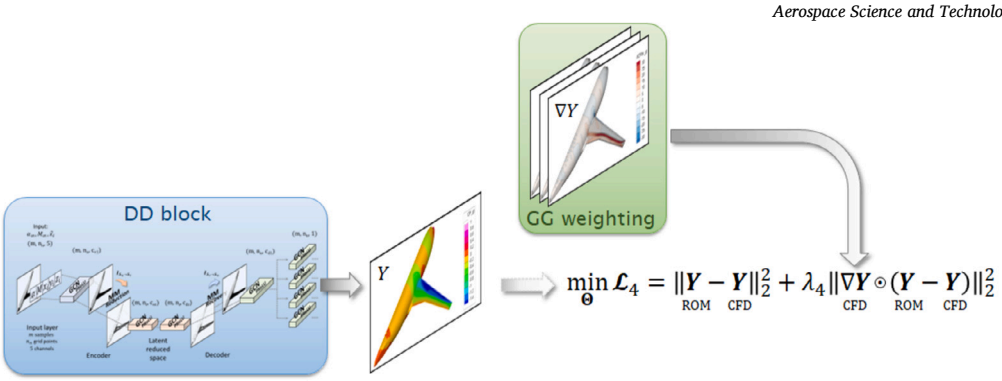
Fig. 4. Schematics of the various gradient-guided approaches. The blue block is for the GCN-MM-AE, corresponding to the data-driven model in Fig. 3; the green block is for the gradient-guided embedding. (For interpretation of the colours in the figure(s), the reader is referred to the web version of this article.)

of the CFD simulations are in Ref. [15]. Fig. 6 Panel (b) illustrates the variation of the lift coefficient C_L with the sampled freestream conditions, obtained by integration of the pressure and shear stresses results from the reference CFD solutions. The lift coefficient correlates nonlinearly with the angle of attack α_∞ and there is also a dependency across the Mach range M_∞.

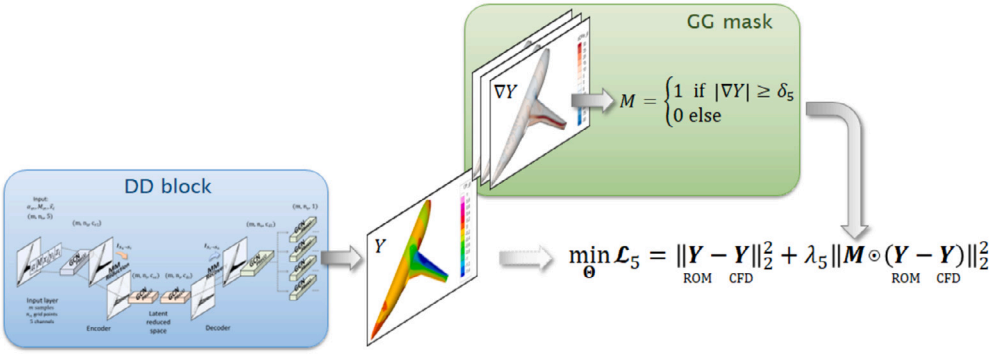
Table 1 clarifies the input and output variables involved in the CRM test case, including the dimensions of the tensors. The inputs also involve the coordinates of the mesh $\boldsymbol{x}$, aimed at inferring the spatial mapping, and the connectivity $\hat{\boldsymbol{A}}$, required to embed the information across from neighbouring nodes in the GCN layers. The physics embedding terms, consisting of the solution gradients, are embedded at different stages of the training framework depending on the specified approach, as described in Section 2.2. Despite this test case deals with the envelope of Mach number and angle of attack conditions, the framework is suitable to other types of input conditions (side slip or Reynolds number) and include design parameters in design optimisation tasks.

Table 1
Relation of inputs, outputs and physics embeddings involved in the CRM framework. The predicted quantities involves the pressure coefficient $\boldsymbol{Y} = [C_P, C_{\tau_x}, C_{\tau_y}, C_{\tau_z}]$, global inputs are the $\boldsymbol{s} = [\alpha_\infty, M_\infty]$ and $\boldsymbol{x} = [x, y, z]$ the mesh coordinates.

Inputs	Outputs	Physics Terms
$[\boldsymbol{s}, \boldsymbol{x}] \in \mathbb{R}^{n_n \times (2+3)}$, $\hat{\boldsymbol{A}} \in \mathbb{R}^{n_e \times 3}$	$\boldsymbol{Y} \in \mathbb{R}^{n_n \times 4}$	$\nabla\boldsymbol{Y} \in \mathbb{R}^{n_n \times 4 \times 3}$

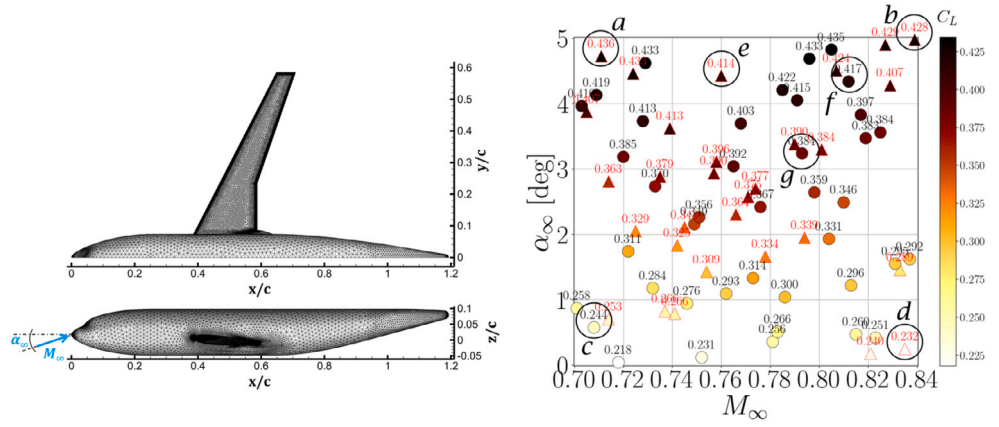


(a) Approach 4: reference gradients used as weights to magnify the model outputs loss.



(b) Approach 5: masking of the model outputs loss by a gradient threshold.

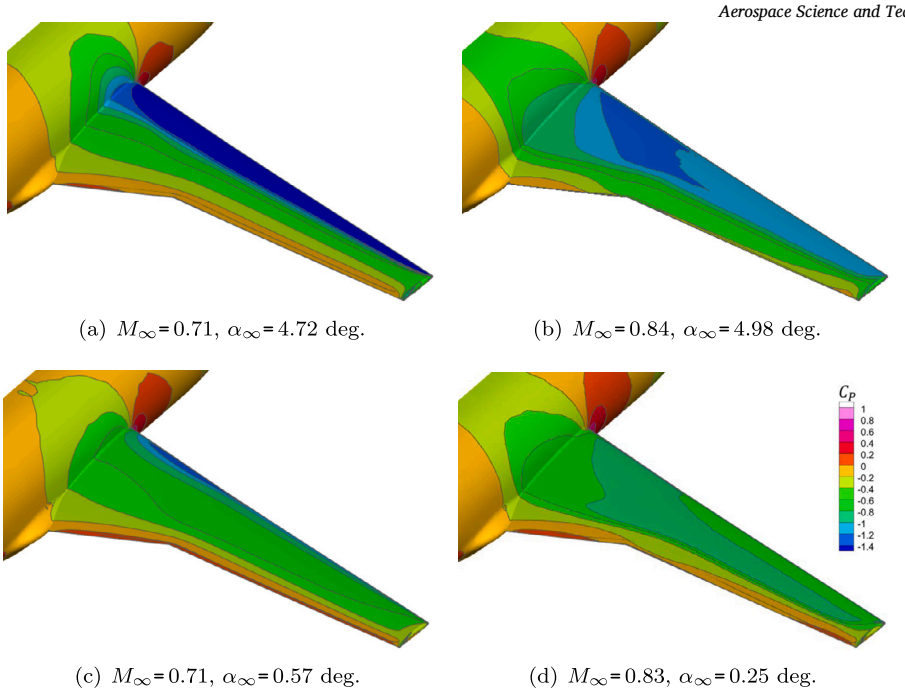
Fig. 5. Gradient-guided approaches continued from Fig. 4. Operator ⊙ denotes Hadamard (element-wise) product.



(a) Surface mesh representation of the CRM model.

(b) Lift coefficient C_L results from CFD of the envelope sampling. Values are also labelled for clarity. Circles with black labels are the samples chosen to train our models, and triangles with red labels are only used for testing.

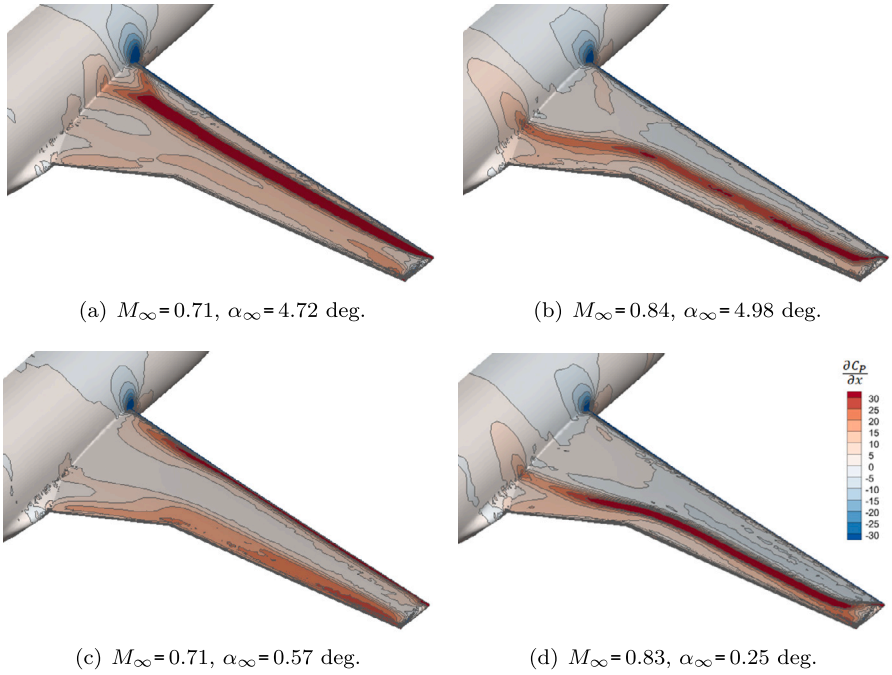
Fig. 6. Reference dataset of the Common Research Model test case.



(a) $M_\infty = 0.71$, $\alpha_\infty = 4.72$ deg. (b) $M_\infty = 0.84$, $\alpha_\infty = 4.98$ deg.

(c) $M_\infty = 0.71$, $\alpha_\infty = 0.57$ deg. (d) $M_\infty = 0.83$, $\alpha_\infty = 0.25$ deg.

Fig. 7. Pressure coefficient C_P distribution for the CFD samples labelled with the same letters in Fig. 6 Panel (b).



(a) $M_\infty = 0.71$, $\alpha_\infty = 4.72$ deg. (b) $M_\infty = 0.84$, $\alpha_\infty = 4.98$ deg.

(c) $M_\infty = 0.71$, $\alpha_\infty = 0.57$ deg. (d) $M_\infty = 0.83$, $\alpha_\infty = 0.25$ deg.

Fig. 8. Streamwise derivative of the pressure coefficient $\frac{\partial C_P}{\partial x}$ distribution for the same selected CFD cases.

3.2. Background physics

From the reference CFD database, we analysed the results at the extremes of the envelope, labelled with letters *a* to *d* in Fig. 6 Panel (b). Labelled samples *e* to *g* will be used for analysis of results in Section 4. Fig. 7 shows the pressure coefficient C_P for the selected samples, arranged by low α_∞ on the bottom Panels and high M_∞ on the right Panels. Significant variation of the pressure distribution with the freestream conditions occur. At lower Mach numbers, left Panels, we observe a transition of the shock wave towards the leading edge with increasing the angle of attack, as well as an intensity increase. By contrast, increasing the Mach number, the peak pressure distribution flattens and the shock wave becomes stronger. Furthermore, the shock-wave location remains stable independently of the angle of attack, right panels.

In this work, we also leverage the gradients of the distributed aerodynamic quantities. Fig. 8 illustrates the derivative in the streamwise direction $\frac{\partial C_P}{\partial x}$. The streamwise derivative is typically the dominant component to capture the most crucial flow features, e.g. flow acceleration along the leading edge of the wing or the shock waves on the upper and lower surfaces. Derivatives along the crosswise direction $\frac{\partial C_P}{\partial y}$, typically useful to capture tip vortices, or the vertical direction $\frac{\partial C_P}{\partial z}$, are also reported in Appendix A. Furthermore, the gradients for the wall shear stress components $\left[\nabla C_{\tau_x}, \nabla C_{\tau_y}, \nabla C_{\tau_z}\right]$, are also computed but not re-

ported here for brevity. We observe how the $\frac{\partial C_P}{\partial x}$ clearly identifies the location of the shock wave on the wing on the various samples. In lower M_∞ (left panels), a significant variation in shock location and intensity is visible. By contrast, at the higher Mach numbers (right panels), the shock wave remains stable, as expected.

Fig. 7 showcases how the range of flight conditions chosen (transonic regime) is particularly challenging because small variations of operating conditions cause large variations in the flow field [25]. The distinct physics phenomena observed in this preliminary analysis justifies the implementation of a method based on geometric deep learning for the prediction of aerodynamic fields on the surface of the aircraft. Furthermore, the behaviour of the solution gradients motivates investigation of hybrid approaches which leverage these terms to enrich the reduced-order modelling inference.

4. Results

To address the comparison of the various frameworks, we laid out different analyses based on three scenarios prescribed via variation of the dataset and model sizes:

1. **Large model and original dataset**: to start with, a similar model architecture from the steady-state framework developed in Refs. [25,26] is adopted, featuring the same kernel size prescription. The dataset includes 40 CFD solutions for training, with the remaining 30 as test set. This scenario considers the situation where enough resources are available, both for a more in-depth preliminary scan of the design space and for computationally sizeable ROM memory.
2. **Large model and reduced dataset**: the model size is preserved but the dataset is reduced to 20 training samples, resulting in 50 for testing. This is the scenario where the amount of data is limited.
3. **Small model and original dataset**: based on the same multi-mesh architecture, the number of kernels (channels) in each layer is halved. The original dataset split of 40 training and 30 test samples is preserved. This is the scenario where the computing memory capacity is limited.

Each of these scenarios are frequent in aerospace related tasks. Hence the interest in demonstrating the adequacy of the various approaches at each of these distinctly possible events. To this aim, we assessed the performance of the proposed frameworks for each scenario by analysing the predicted distributed fields as well as the integrated scalar quantities across the sampled envelope. Note that a fourth scenario involving a small model and a reduced dataset, i.e. combining scenarios 2 and 3, is also interesting but was not considered here for brevity.

The definitive steady-state GCN-MM-AE model implementation, Fig. 3, adopted a mesh-resolution compression ratio of 16 (from 78k to 5k nodes). The first-order gradient-based algorithm Adam [18] and the mean squared error (MSE) loss function [3] were adopted for the optimisation of the model parameters. The inputs were standardised with the mean fields and normalised with the standard deviation to ensure a more efficient training process. Full details of the steady-state GCN-MM-AE model architecture and hyper-parameters are reported in Appendix B. The developed GCN-MM-AE framework from that work is used here as the purely data-driven baseline for comparison with the various hybrid approaches proposed in this study. Furthermore, Appendix C reports the choice of hyper-parameter values relevant to the proposed gradient-based approaches.

The various frameworks are designed to predict the pressure C_P and the three shear-stress $[C_{\tau_x}, C_{\tau_y}, C_{\tau_z}]$ fields. No bespoke ROMs were dedicated to directly obtain the resulting forces and moments. Load resultants are instead obtained by surface integration of the distributed quantities.

4.1. Scenario 1: large model size and original dataset

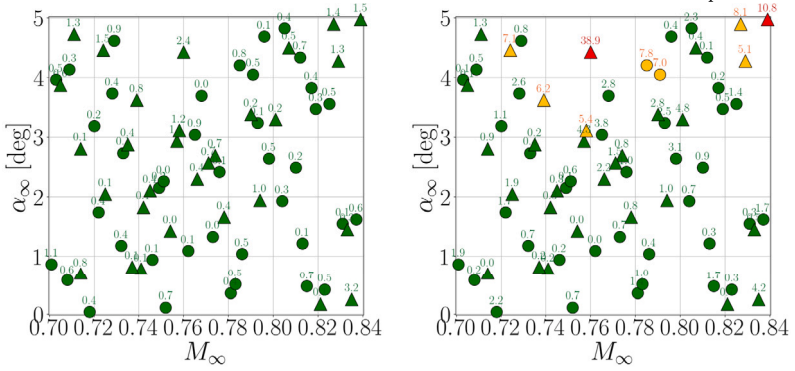
This analysis is for the large model size (reported in Table B.7 of the Appendix) trained with the original training dataset of 40 samples out of the 70. For brevity, we report results on lift C_L and pitching moment coefficients C_{M_y}, as these are the most interesting resultants in aircraft design, with drag typically being an order of magnitude lower. Fig. 9 reports the relative error in percentage for the C_L (left) and the C_{M_y} (right) obtained with the original data-driven model architecture across the entire dataset. Reported errors are classified by training (circles) and test (triangles), to assess model generalisation to unseen conditions. The traffic-light colour scheme, convenient for visual judgement, displays errors below 5% in green, below 10% in amber and above 10% in red. We observe how lift (left panel) is very well predicted across the entire dataset, with all errors consistently below 5%.

The errors on the pitching moment (right panel) were found, by contrast, considerably larger. Slight discrepancy of the shock wave can cause significant variations of the pitching moment. Not surprisingly, the error is larger towards higher angles of attack, where the response becomes nonlinear. In particular, we focus on the sample at $M_\infty = 0.76$ and $\alpha_\infty = 4.42$ deg, sample e in Fig. 6(b). This test sample proved to be the most complicated to predict due to the lack of training samples in the vicinity to adequately learn that region of the envelope. These prediction results represent the benchmark to assess the performance of the proposed gradient-guided schemes. Figs. 10 to 14 illustrate the same analyses for the five distinct GG frameworks. The traffic-light classification results convenient to identify the best suited embeddings for enhanced modelling accuracy. The error on the lift coefficient, left panels, remained low among all the models. Observing the right panels, we identified three out of five GG models that outperformed the pitching moment coefficient predictions: adding the gradient of the model outputs to the loss function, Fig. 11; using the gradients for loss weighting, Fig. 13; and the masking, Fig. 14. By contrast, adding the gradients as target predictions, Fig. 10, or imposing them as latent states, Fig. 12, were not found as interesting.

To complete the error analysis, Table 2 reports a statistical summary of the lift, drag and pitching moment coefficient errors on the test set, represented by triangles in Figs. 9 to 14, to verify model performance in flow conditions not seen during model training. For each approach, the average error, the standard deviation and the worst sample are reported. Furthermore, for the various GG models, the variation with respect to the baseline data-driven implementation is also reported for each metric. We observe how, despite there are large variations on the C_L and C_D errors compared to the baseline values, the average errors are actually significantly low. The average error was found instead an order of magnitude larger for the C_{M_y} and, therefore, more significant. To remark that the pitching moment is strongly influenced by the location of the shock wave, a flow feature that the gradient embedding should crucially capture. Three of the proposed GG frameworks were found to improve the average error on the pitching moment by at least 16%. These are for the gradient loss, the weighting and the masking approaches. By contrast, the gradient targets and the latent state approaches were not found to improve the baseline data-driven accuracies. In the case of the target approach (GG model 1), the worse performance could be attributed to the fact that the same model size was adopted to predict a larger number of target variables. Improvements might be possible with increasing the size of the model layers, impacting on the computing memory. We conclude that embedding physics terms as direct variables to predict, either in interim layers or as outputs, does not seem adequate.

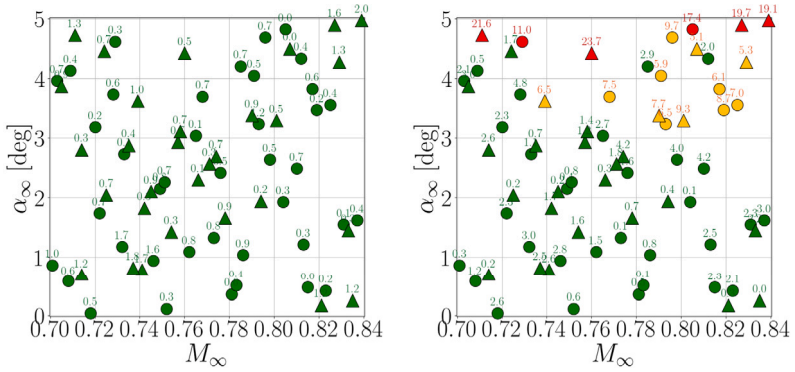
4.2. Scenario 2: large model size and reduced dataset

New ROMs were generated with only 20 dataset samples as opposed to 40 but keeping the same model size. The idea is to demonstrate the performance of the GG frameworks in tasks with limited data availability. From the three best candidates identified in the previous section, the



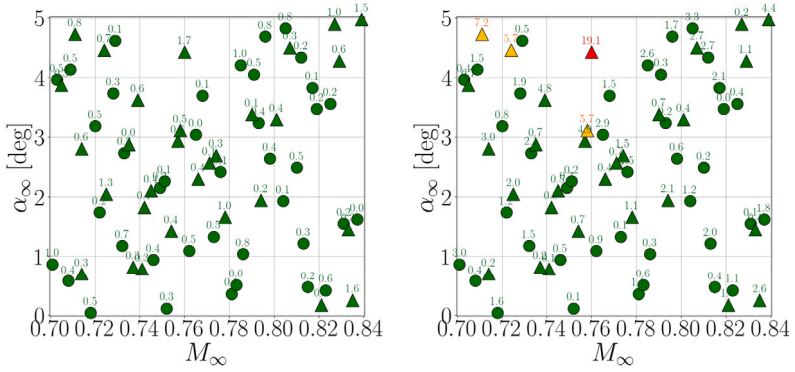
(a) Lift coefficient prediction error ε_{C_L} in %. (b) Pitching moment coefficient prediction error $\varepsilon_{C_{M_y}}$ in %.

Fig. 9. Baseline: data-driven only model (Fig. 3), scenario 1: large model size and original training samples. Prediction error of aerodynamic forces on the entire dataset. Classified by training (circles) and test (triangles) samples. Green: error below 5%, amber: between 5% and 10%, red: above 10%.



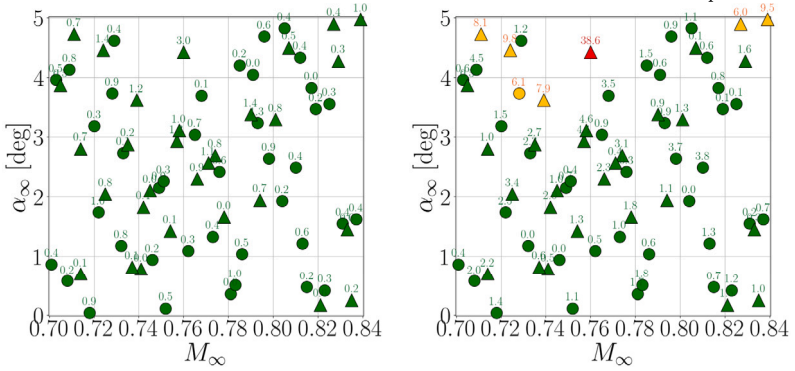
(a) Lift coefficient prediction error ε_{C_L} in %. (b) Pitching moment coefficient prediction error $\varepsilon_{C_{M_y}}$ in %.

Fig. 10. GG model 1: gradients predicted as model outputs (Fig. 4(a)), scenario 1: large model size and original training samples. Prediction error of aerodynamic forces on the entire dataset.



(a) Lift coefficient prediction error ε_{C_L} in %. (b) Pitching moment coefficient prediction error $\varepsilon_{C_{M_y}}$ in %.

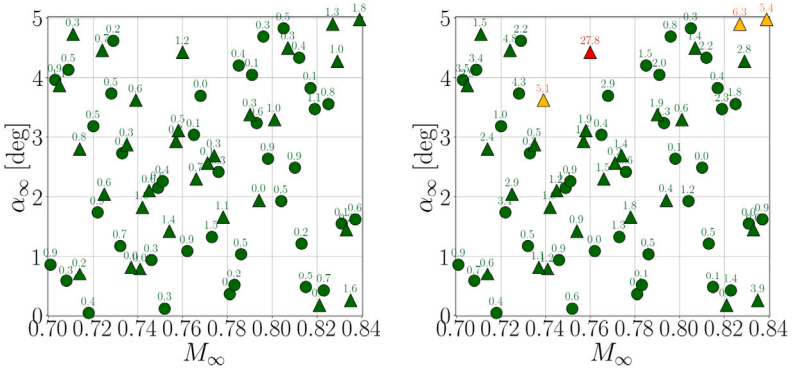
Fig. 11. GG model 2: gradient loss from model outputs (Fig. 4(b)), scenario 1: large model size and original training samples. Prediction error of aerodynamic forces on the entire dataset.



(a) Lift coefficient prediction error ε_{C_L} in %.

(b) Pitching moment coefficient prediction error $\varepsilon_{C_{M_y}}$ in %.

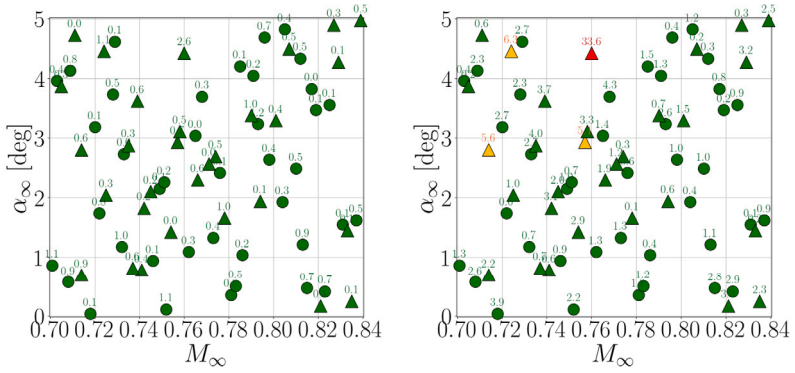
Fig. 12. GG model 3: gradients predicted as latent states (Fig. 4(c)), scenario 1: large model size and original training samples. Prediction error of aerodynamic forces on the entire dataset.



(a) Lift coefficient prediction error ε_{C_L} in %.

(b) Pitching moment coefficient prediction error $\varepsilon_{C_{M_y}}$ in %.

Fig. 13. GG model 4: loss weighted by streamwise derivative (Fig. 5(a)), scenario 1: large model size and original training samples. Prediction error of aerodynamic forces on the entire dataset.



(a) Lift coefficient prediction error ε_{C_L} in %.

(b) Pitching moment coefficient prediction error $\varepsilon_{C_{M_y}}$ in %.

Fig. 14. GG model 5: loss masked by a gradient threshold (Fig. 5(b)), scenario 1: large model size and original training samples. Prediction error of aerodynamic forces on the entire dataset.

Table 2
Prediction error statistics on the test samples of the CRM test case; comparison for the various modelling approaches based on the large model and original training dataset (scenario 1).

	ε_{C_L} [%]			ε_{C_D} [%]			$\varepsilon_{C_{M_y}}$ [%]		
	mean	std dev	worst	mean	std dev	worst	mean	std dev	worst
DD model (baseline)	0.8	0.7	3.2	0.8	0.8	3.6	3.7	7.1	38.9
GG model 1 (grad. targets)	0.8	0.5	2.0	1.0	0.9	3.5	4.9	6.7	23.7
Δ to baseline [%]	7.5	-28.8	-38.7	26.5	10.1	-3.1	32.8	-5.1	-39.1
GG model 2 (grad. loss)	0.6	0.4	1.7	0.9	1.1	5.2	2.5	3.6	19.1
Δ to baseline [%]	-23.3	-38.8	-47.6	23.0	33.7	45.2	-31.3	-48.9	-51.0
GG model 3 (grad. latent state)	0.7	0.6	3.0	1.1	1.0	3.9	4.0	7.0	38.6
Δ to baseline [%]	-9.4	-17.3	-6.2	41.5	21.9	8.5	7.2	-1.7	-0.9
GG model 4 (grad. weighting)	0.7	0.5	1.8	1.0	1.2	5.8	2.9	4.9	27.8
Δ to baseline [%]	-12.4	-33.2	-42.9	27.9	48.8	63.7	-21.7	-31.2	-28.6
GG model 5 (grad. masking)	0.5	0.5	2.6	0.9	0.8	3.6	3.1	5.9	33.6
Δ to baseline [%]	-36.5	-31.8	-20.0	21.5	3.2	0.3	-15.7	-16.9	-13.8

Table 3
Prediction error statistics on the test samples of the CRM test case; comparison for selected modelling approaches based on the large model and reduced training dataset (scenario 2).

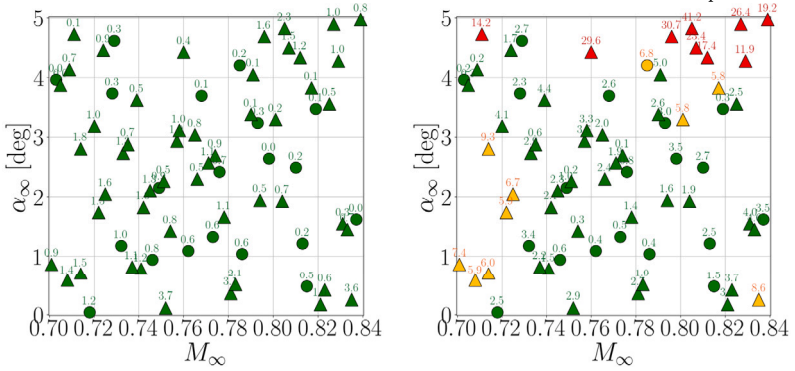
	ε_{C_L} [%]			ε_{C_D} [%]			$\varepsilon_{C_{M_y}}$ [%]		
	mean	std dev	worst	mean	std dev	worst	mean	std dev	worst
DD model (baseline)	1.1	0.8	3.7	1.5	1.5	8.2	7.0	8.9	41.2
GG model 2 (grad. loss)	0.6	0.4	2.2	1.6	1.8	10.3	3.8	5.8	33.7
Δ to baseline [%]	-46.5	-44.5	-39.2	12.5	22.1	26.2	-44.9	-35.4	-18.2
GG model 4 (grad. weighting)	1.1	1.0	4.2	1.9	2.3	12.9	6.2	9.2	37.8
Δ to baseline [%]	4.0	26.8	13.6	31.8	49.4	58.2	-11.5	2.5	-8.4
GG model 5 (grad. masking)	1.3	1.6	7.0	1.7	2.5	11.3	4.5	4.6	25.8
Δ to baseline [%]	23.4	99.1	90.6	15.1	62.9	38.9	-35.4	-48.8	-37.5

Table 4
Prediction error statistics on the test samples of the CRM test case; comparison for selected modelling approaches based on the reduced model and original training dataset (scenario 3).

	ε_{C_L} [%]			ε_{C_D} [%]			$\varepsilon_{C_{M_y}}$ [%]		
	mean	std dev	worst	mean	std dev	worst	mean	std dev	worst
DD model (baseline)	1.0	1.1	5.3	1.1	1.3	6.7	4.8	7.1	33.8
GG model 2 (grad. loss)	0.8	0.5	1.8	1.2	1.1	4.9	3.4	5.8	32.4
Δ to baseline [%]	-18.9	-49.5	-66.3	4.7	-16.4	-26.7	-28.9	-19.0	-4.3
GG model 4 (grad. weighting)	1.2	0.8	3.6	1.3	1.2	5.7	5.2	6.8	26.4
Δ to baseline [%]	20.4	-22.8	-32.4	16.6	-7.5	-15.8	10.1	-4.0	-22.1
GG model 5 (grad. masking)	0.9	1.1	4.9	1.5	1.6	6.7	4.2	6.5	31.9
Δ to baseline [%]	-8.1	3.6	-7.8	31.5	19.1	-0.9	-11.1	-9.1	-5.8

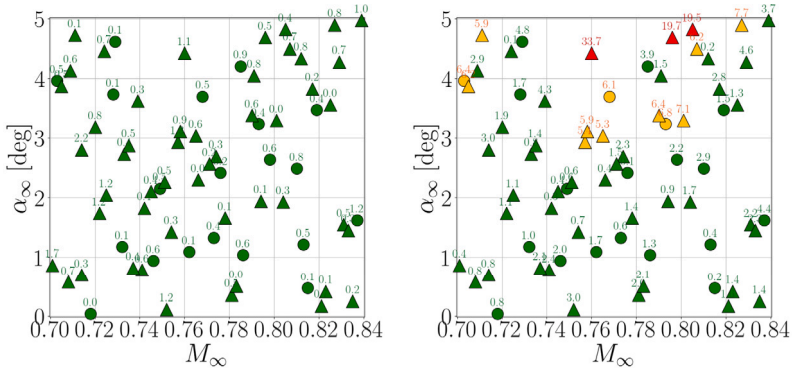
output differentiation (GG method 2) and the masking (GG method 5) approaches were found the better candidates here. To keep brevity, from now on we present results for these two schemes only, and compared their contribution against the baseline data-driven model. Figs. 15 to 17 illustrate the prediction error on the C_L (left panels) and the C_{M_y} (right panels) for the baseline DD model, the output gradient and the masking approaches, respectively; adopting the same traffic-light criteria as before. Once again, the lift is adequately predicted by all the models. Regarding the pitching moment, with less training data, we found significant degradation of the prediction accuracy, Fig. 15 against Fig. 9. This outcome is expected given that the model was trained with half the data size. However, significant improvements are observed with both GG schemes, Figs. 16 and 17, consistent with the observations from the analysis of the first scenario. We also observe how the worst predictions are for the test samples, i.e. most red and amber samples are triangles, not included in the training dataset. This evidences that the design space is not sufficiently sampled. In this scenario, our gradient-guided implementations manage to improve the confidence by the model. By leveraging the gradients better, the resulting ROM outperforms the baseline implementation in regions not covered by training samples, successfully achieving our objective.

The statistical summary of the prediction error across the test set, now comprising 50 samples instead of 30, is reported in Table 3 for the two selected GG approaches compared to the DD baseline results. The results for the weighted method are also included, justifying the choice of the masking over this scheme. We found the gradient loss model only to improve the C_L, while the C_D was found worse for all selected GG approaches. However, again, the magnitude of the errors are low across the envelope (on the range of $1-2\%$) and, consequently, small variations cause large percentages. Therefore, this degradation is less critical. Furthermore, the C_D performance is an order of magnitude lower than the C_L, contributing to larger percent errors. Focusing now on the C_{M_y},



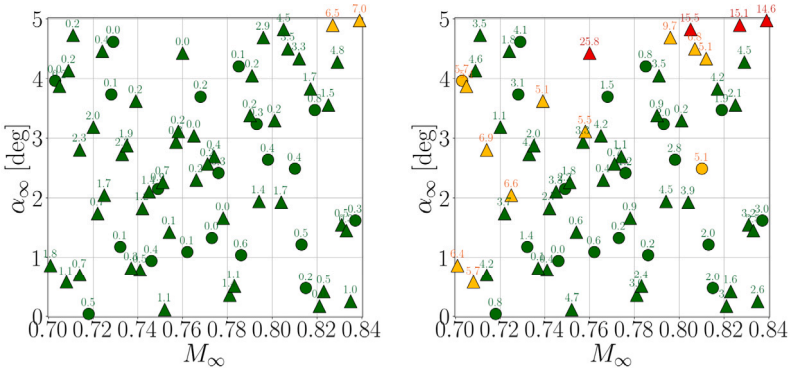
(a) Lift coefficient prediction error ε_{C_L} in %. (b) Pitching moment coefficient prediction error $\varepsilon_{C_{M_y}}$ in %.

Fig. 15. Baseline: data-driven only model (Fig. 3), scenario 2: large model size and reduced training samples. Prediction error of aerodynamic forces on the entire dataset. Classified by training (circles) and test (triangles) samples. Green: error below 5%, amber: between 5% and 10%, red: above 10%.



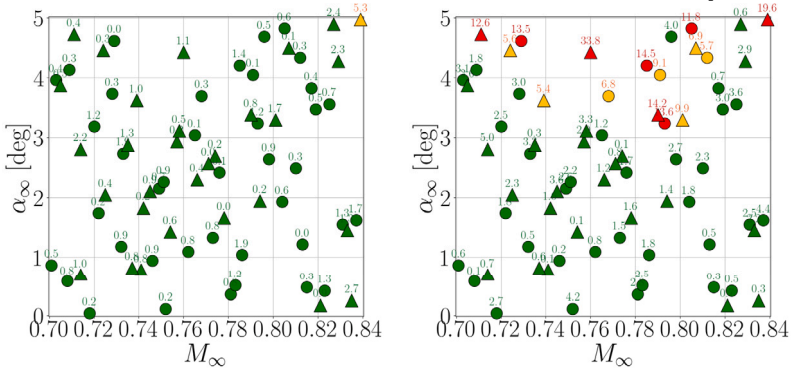
(a) Lift coefficient prediction error ε_{C_L} in %. (b) Pitching moment coefficient prediction error $\varepsilon_{C_{M_y}}$ in %.

Fig. 16. GG model 2: gradient loss from model outputs (Fig. 4(b)), scenario 2: large model size and reduced training samples. Prediction error of aerodynamic forces on the entire dataset.



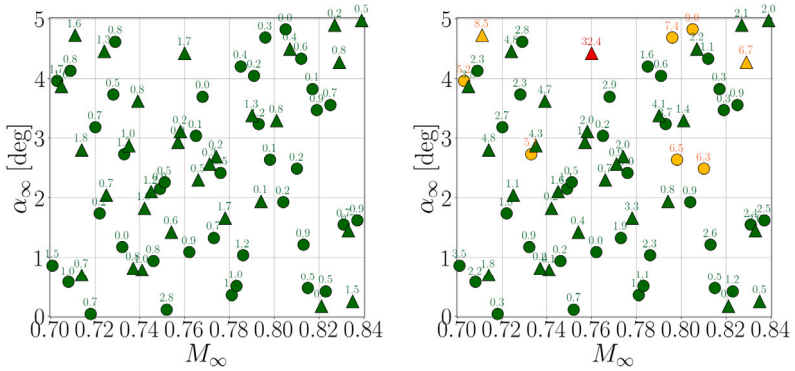
(a) Lift coefficient prediction error ε_{C_L} in %. (b) Pitching moment coefficient prediction error $\varepsilon_{C_{M_y}}$ in %.

Fig. 17. GG model 5: loss masked by a gradient threshold (Fig. 5(b)), scenario 2: large model size and reduced training samples. Prediction error of aerodynamic forces on the entire dataset.



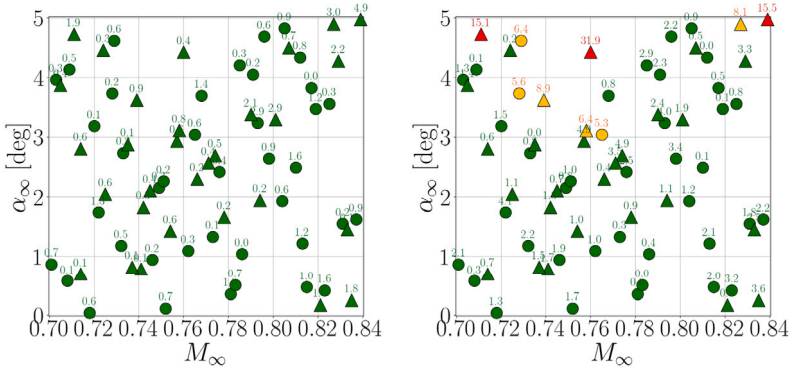
(a) Lift coefficient prediction error ε_{C_L} in %. (b) Pitching moment coefficient prediction error $\varepsilon_{C_{M_y}}$ in %.

Fig. 18. Baseline: data-driven only model (Fig. 3), scenario 3: limited model size and original training samples. Prediction error of aerodynamic forces on the entire dataset. Classified by training (circles) and test (triangles) samples. Green: error below 5%, amber: between 5% and 10%, red: above 10%.



(a) Lift coefficient prediction error ε_{C_L} in %. (b) Pitching moment coefficient prediction error $\varepsilon_{C_{M_y}}$ in %.

Fig. 19. GG model 2: gradient loss from model outputs (Fig. 4(b)), scenario 3: limited model size and original training samples. Prediction error of aerodynamic forces on the entire dataset.



(a) Lift coefficient prediction error ε_{C_L} in %. (b) Pitching moment coefficient prediction error $\varepsilon_{C_{M_y}}$ in %.

Fig. 20. GG model 5: loss masked by a gradient threshold (Fig. 5(b)), scenario 3: limited model size and original training samples. Prediction error of aerodynamic forces on the entire dataset.

the best two GG approaches resulted in significant improvements of over 45% and 35% respectively. Note also how the worst case prediction is significantly improved as well.

4.3. Scenario 3: small model size and original dataset

We investigated the performance of our various frameworks with a reduced model size. New ROMs were generated by halving the number of channels at each NN layer. As reported in Appendix B, the overall model size was reduced by a factor of 4, resulting in an equivalent reduction of GPU memory requirements. Figs. 18 to 20 report the error on the integrated loads for the baseline DD and the best two GG approaches. Focusing on the pitching moment error (right panels), for the baseline data-driven model, Fig. 18, the accuracy of the prediction diminishes at higher angles of attack, where a larger nonlinear response occurs. Again, larger errors can be expected since the size of the model (number of regression parameters) is significantly smaller, limiting the capacity of the model to learn the physics. In contrast to the previous setting involving limited training data, where worst predictions comprised test samples (triangles), now we also observe training samples being wrongly predicted (red and amber circles). This is consequence of the smaller model not having enough capacity to accommodate the nonlinear response across the envelope. Nevertheless, with the gradient-based methods, the C_{M_y} error is visibly improved, Figs. 19 and 20 respectively. With either model, no training samples and fewer test samples were found with error larger than 10%. Embedding the gradients into the data-driven framework enhances a model of smaller capacity to perform closer to a larger model.

As with the previous scenarios, Table 4 reports the statistical comparison on the prediction error for the test set among the selected modelling frameworks. Again, the results for the weighted scheme are also added, further justifying the choice for the gradient loss and the masking approaches. In this configuration, we observe similar trends compared to the previous analyses. The best two gradient-based schemes were found to improve the pitching moment C_{M_y} error statistics on the test set by 29% and 11% respectively. Therefore, the output gradient loss model was found to outperform the other frameworks. Although the masking scheme showed a slight improvement, the weighted scheme, on the contrary, did not provide a significant increment over the baseline DD model. This indicates that a simpler boolean masking is more effective for highlighting high-gradient regions compared to a variable-field mask.

4.4. Sensitivity to training set size

To further demonstrate the advantage of gradient-guided data-driven approaches while reducing the dataset size, we generated new models for the selected approaches with an interim scenario of 30 training samples. In combination with the first two scenarios, Tables 2 and 3, we addressed a sensitivity analysis of the model accuracy with the training dataset size for the various modelling approaches. Fig. 21 illustrates the evolution of the prediction errors for the C_L (left panel), C_D (mid panel) and C_{M_y} (right panel). The average errors on the test set are in black and on the left axis, while the worst predictions are in red and on the right axis. Results are reported for the baseline data-driven model (○), the gradient loss approach (□) and the masking approach (△). For the various coefficients, the prediction accuracy generally improves with increasing the dataset, which is expected. Observe how the slope of the curves, particularly the average values, are deeper with the baseline model. This confirms that the purely data-driven model is more sensitive to dataset size. By contrast, the gradient models, particularly the gradient-loss approach, present slightly shallower curves, which means that there is less degradation of the accuracy with reducing the data. This result demonstrates the adequacy of our proposed physics-enhanced approach. Furthermore, when comparing against the DD model errors, improvements are consistent for the gradient loss approach (GG 2) across the various training set cases, particularly for the lift and pitching moment coefficients. On the other hand, the masking approach (GG 5) was found only to improve the C_L predictions at the smallest train set size, whereas improvements on C_{M_y} were found in every case.

4.5. Aerodynamic field predictions and gradients

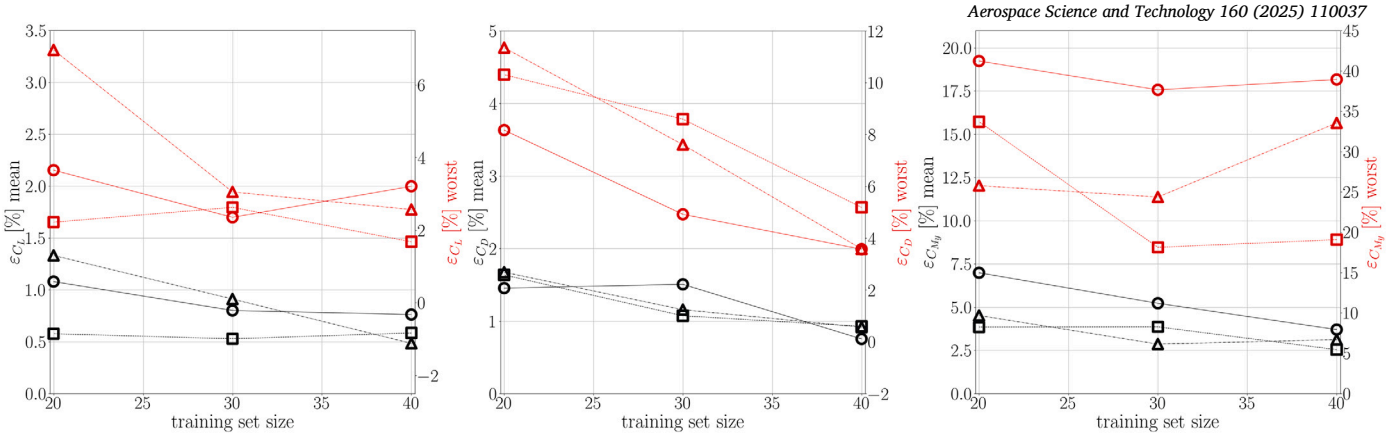
The analysis for the predicted integrated loads is useful to broadly assess the performance across the operating envelope and compare among the various frameworks and scenarios. However, analysis of the distributed fields is more interesting, as these are actually the quantities predicted by the models. Distributed fields are more challenging to predict accurately compared to scalar loads and are more useful for aircraft design development. For brevity, only the results for the pressure predictions are reported here. Shear stress results are provided in Appendix D as their contribution to the resulting aerodynamic forces is less significant. In addition, we report analysis of only the approaches already selected in the previous results.

4.5.1. Pressure field predictions

Figs. 22 to 24 illustrate comparisons of the pressure coefficient distribution obtained with the various models for each scenario, respectively. The flight condition corresponds to sample *e* labelled in Fig. 6(b). This sample is interesting for being the worst predicted, due to the lack of training samples in the vicinity. In each panel, the various modelling approaches are arranged in columns, with the ground-truth CFD solution on the left. The upper panels are for the C_P prediction, while the lower panels are for the error (difference) against the CFD reference. The error values on the integrated forces are also reported in the captions, corresponding to the values shown within the respective dataset error plots in the previous sections.

For the first scenario, Fig. 22, we observe a discrepancy on the shock-wave prediction with the data-driven model compared to the CFD reference. These results are the benchmark for the two selected GG-based frameworks. We found that the prediction error improved along the shock line (more faded blue) and at the root of the wing (diminished red portions). The C_P remained well captured elsewhere. Fig. 23 is for the reduced dataset scenario. With the purely data-driven implementation, the shock wave was found slightly better captured from $\sim 30\%$ of the span but worse at the root. The gradient loss model (GG approach 2) was again found to improve the shock-wave prediction. Furthermore, the increment with the masking method (GG approach 5) was smaller than with the gradient-loss approach and a degradation is observed towards the trailing edge of the wing. The reason for this is that the masking filter prioritises regions containing large gradients rather than smooth solution variations, i.e. shock waves and the trailing edge, respectively. Observing the comparisons for the last scenario, Fig. 24, similar trends are observed but the improvements are less visible than with the first scenario. Despite with the GG masking method the worst-case prediction was not significantly improved, the analysis on the integrated forces across the entire dataset demonstrated to outperform the DD implementation. Regardless, these comparisons confirmed the gradient-loss as the best predictive method.

The GG models were found to provide significantly greater improvements in the scenario of limited training samples, Section 4.2. Fig. 25 reports the pressure coefficient prediction comparison for sample *f* in Fig. 6(b) with the large model and reduced training dataset. The same panel row (prediction and error) and column arrangements (modelling approaches) is adopted. We observe that with limited data, the DD model (second column) predicted the location of the shock wave excessively downstream. The prediction was found considerably improved with the gradient-loss model (third column). By contrast, the masking model overpredicted the suction peak towards the leading of the wing, despite the pitching moment error is successfully improved.

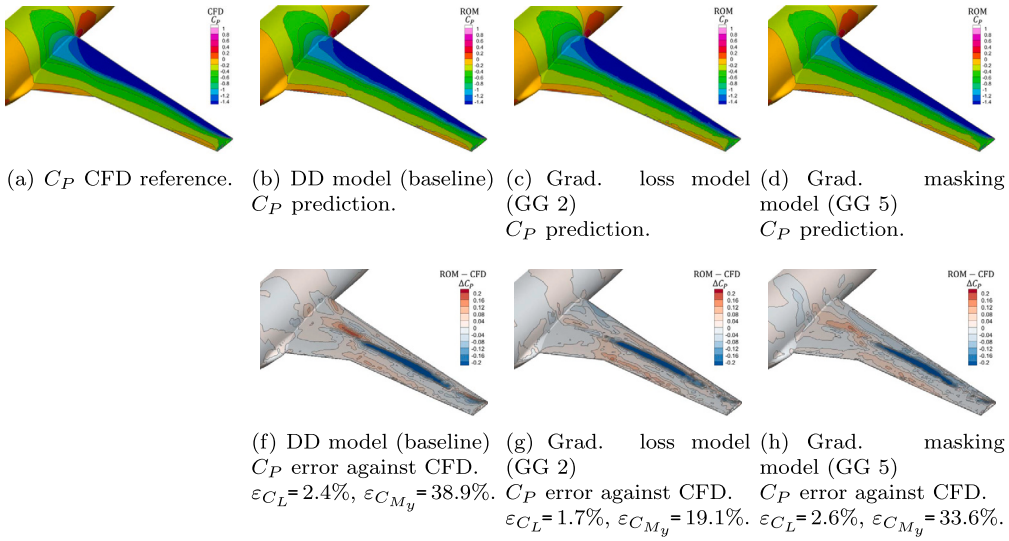


(a) Lift coefficient mean and worst prediction errors.

(b) Drag coefficient mean and worst prediction errors.

(c) Pitching moment coefficient mean and worst prediction errors.

Fig. 21. Sensitivity of the prediction error on the test set to the number of training samples. Mean errors are in black on left axis, and worst predictions are in red on right axis. Models correspond to data-driven (BL) in ○, gradient loss (GG 2) in □ and gradient mask (GG 5) in △.



(a) C_P CFD reference.

(b) DD model (baseline) C_P prediction.

(c) Grad. loss model (GG 2) C_P prediction.

(d) Grad. masking model (GG 5) C_P prediction.

(f) DD model (baseline) C_P error against CFD. ε_{C_L}= 2.4%, $\varepsilon_{C_{M_y}}$= 38.9%.

(g) Grad. loss model (GG 2) C_P error against CFD. ε_{C_L}= 1.7%, $\varepsilon_{C_{M_y}}$= 19.1%.

(h) Grad. masking model (GG 5) C_P error against CFD. ε_{C_L}= 2.6%, $\varepsilon_{C_{M_y}}$= 33.6%.

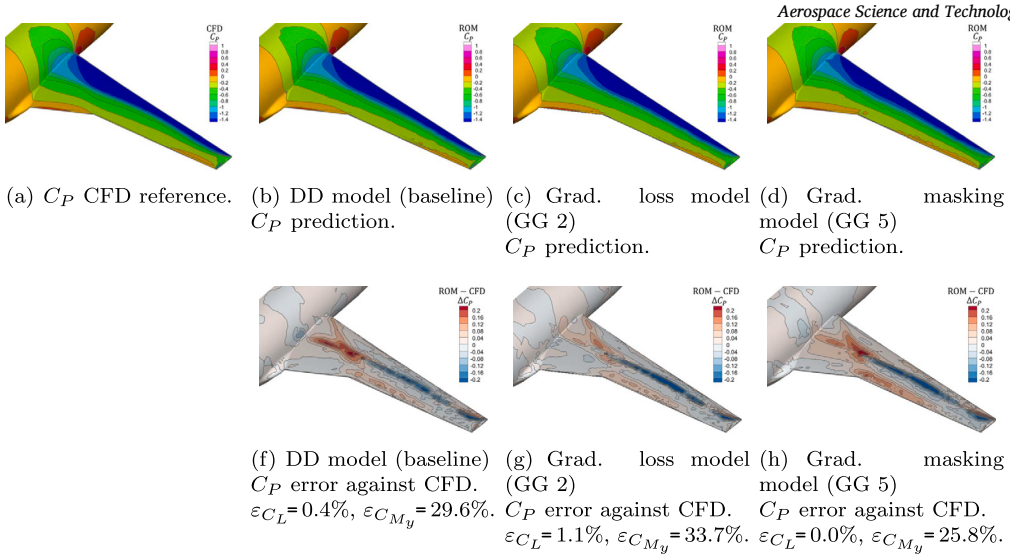
Fig. 22. Pressure coefficient comparison of the various methods against CFD reference at $M_\infty = 0.76$, $\alpha_\infty = 4.42$ deg, $C_{L_{\text{CFD}}} = 0.414$ and $C_{M_{y\text{CFD}}} = -0.019$ (sample *e* in Fig. 6(b)), for the large model and original dataset scenario.

To complete the analysis on the C_P distribution, we analysed the performance of the GG models in the case of limited model size, Section 4.3. Fig. 26 illustrates a similar comparison for training sample *g* in Fig. 6(b) with the small model and the original 40 training samples. The DD model in second column, predicted the shock wave slightly more upstream. By contrast, the gradient-loss and the masking models were both found a significant improvement.

4.5.2. Streamwise pressure gradient predictions

For post-processing of the results, gradients are computed with the `Pyvista` package from the modelled quantities. We analyse here the streamwise derivative of the pressure coefficient only, as Section 3.2 demonstrated being the most relevant gradient component. Figs. 27 to 29 provide a similar analysis of the various ROM approaches with the different scenarios arranged as in the previous section, again for sample *e*. Observing the CFD reference, the shock-wave location is clearly identified along the span of the upper surface of the wing. In Fig. 27 regarding the first scenario, the error distribution clearly shows the location of the shock wave predicted further downstream by the data-driven model compared to the CFD ground truth (mid left panels). Regarding the gradient loss embedding (mid right), the $\frac{\partial C_P}{\partial x}$ field was found better captured (more faded blues and reds), particularly on the inboard side. The streamwise derivative was also marginally better predicted by the masking model (right) but to a lower extend than with the gradient-loss approach. These observations are in agreement with the C_P results above. The small dataset scenario of Fig. 28 evidences similar trends as before. The gradient along the shock-wave is slightly improved except towards the root. Improvements are observed with the gradient-loss method but a larger discrepancy was observed towards the trailing edge of the wing for the masking method, in line with the C_P results. Fig. 29 is for the limited model memory case, suggesting that the increments are not as obvious as with the other scenarios. The masking method was considerably easier to implement than the output-gradient scheme (see Section 4.6) and it still proved capable of marginally outperforming the regular DD approach. Nevertheless, the gradient-loss method was found to consistently outperform all the other frameworks in all the scenarios.

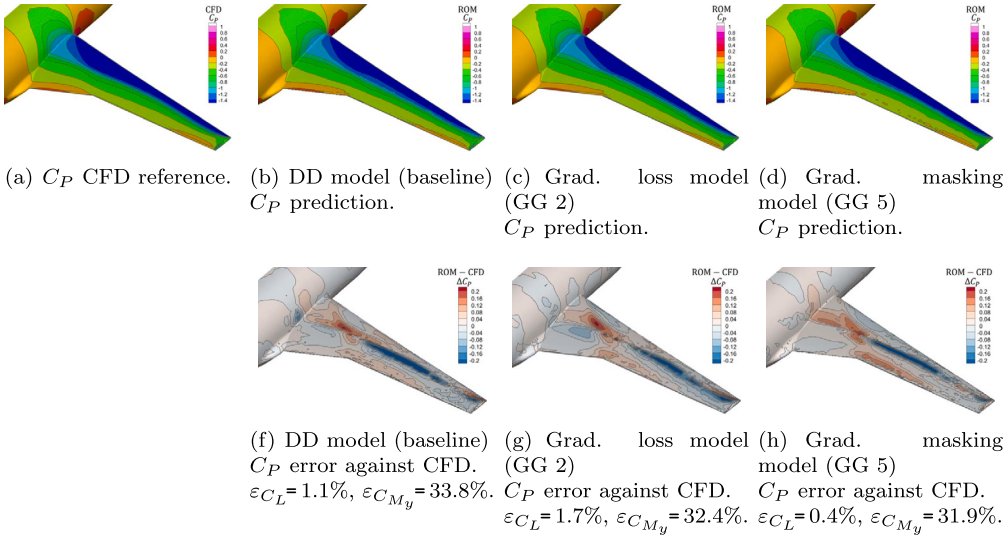
Fig. 30 illustrates the streamwise derivative comparison for sample *f* with the reduced dataset scenario. The second column evidences how the regular DD model mispredicted the shock wave location further downstream. This discrepancy is improved with the GG models (third and last columns). However, the masking model introduced derivative



(a) C_P CFD reference. (b) DD model (baseline) C_P prediction. (c) Grad. loss model (GG 2) C_P prediction. (d) Grad. masking model (GG 5) C_P prediction.

(f) DD model (baseline) C_P error against CFD. ε_{C_L}= 0.4%, $\varepsilon_{C_{M_y}}$= 29.6%. (g) Grad. loss model (GG 2) C_P error against CFD. ε_{C_L}= 1.1%, $\varepsilon_{C_{M_y}}$= 33.7%. (h) Grad. masking model (GG 5) C_P error against CFD. ε_{C_L}= 0.0%, $\varepsilon_{C_{M_y}}$= 25.8%.

Fig. 23. Pressure coefficient comparison of the various methods against CFD reference at $M_\infty = 0.76$, $\alpha_\infty = 4.42$ deg, $C_{L_{\text{CFD}}} = 0.414$ and $C_{M_{y\text{CFD}}} = -0.019$ (sample *e* in Fig. 6(b)), for the large model and reduced dataset scenario.



(a) C_P CFD reference. (b) DD model (baseline) C_P prediction. (c) Grad. loss model (GG 2) C_P prediction. (d) Grad. masking model (GG 5) C_P prediction.

(f) DD model (baseline) C_P error against CFD. ε_{C_L}= 1.1%, $\varepsilon_{C_{M_y}}$= 33.8%. (g) Grad. loss model (GG 2) C_P error against CFD. ε_{C_L}= 1.7%, $\varepsilon_{C_{M_y}}$= 32.4%. (h) Grad. masking model (GG 5) C_P error against CFD. ε_{C_L}= 0.4%, $\varepsilon_{C_{M_y}}$= 31.9%.

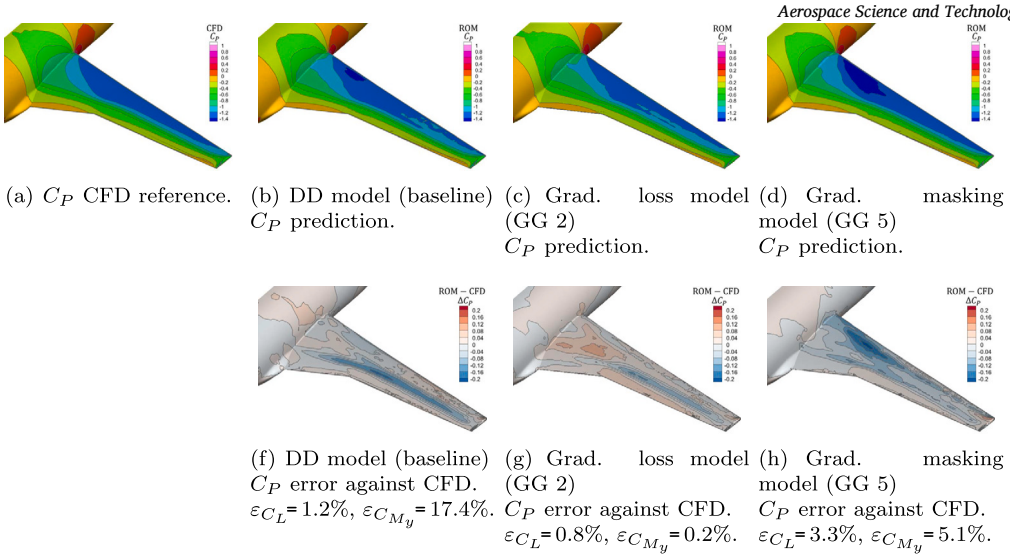
Fig. 24. Pressure coefficient comparison of the various methods against CFD reference at $M_\infty = 0.76$, $\alpha_\infty = 4.42$ deg, $C_{L_{\text{CFD}}} = 0.414$ and $C_{M_{y\text{CFD}}} = -0.019$ (sample *e* in Fig. 6(b)), for the small model and original dataset scenario.

error towards the leading edge of the wing. Regarding sample *g* for the limited model size case, Fig. 31 illustrates the $\frac{\partial C_P}{\partial x}$ comparison for the various models. The plots evidence the upstream prediction of the shock wave with the baseline DD model. Both GG models again manage to rectify this error. In these distinct scenarios, especially with a limited number of training samples, there is a significant sensitivity of prediction accuracy to the data distribution across the envelope. The incorporation of physics contributions has been found to significantly alleviate this effect. These results confirm that embedding of the gradients into the data-driven framework becomes significantly more convenient with limited data available and/or limited model capacity.

The results herein presented are for surrogate modelling of CFD-based solutions. By contrast, in experimental testing (e.g. wind tunnel) complete flow-field information is generally unavailable but discrete sensor measurements are more common instead. Reconstruction methods can be considered to obtain complete distributed fields from sensor measurements. Adequate approaches must be adopted to ensure smooth solution fields. Our proposed GCN-MM-AE framework, embedding a multi-mesh scheme, can be adapted to address this problem. Using the reconstructed fields, it is then possible to use the gradient-guided approaches proposed herein, with the inclusion of a low-pass filter to exclude regions of noisy gradient values.

4.6. Computing cost remarks

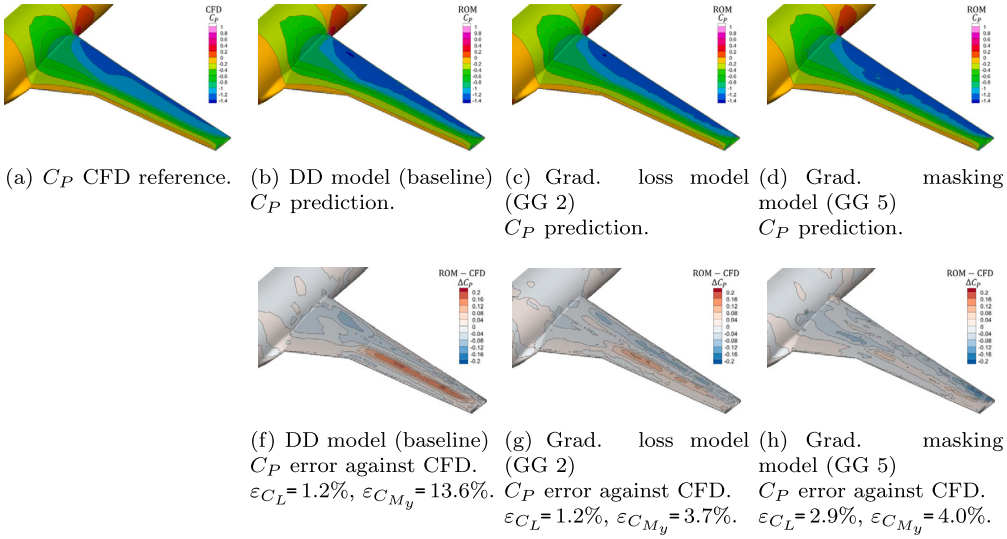
We conclude with a remark on the computing efficiency for the various modelling approaches. Table 5 provides a summary of the computing cost involved with the generation of the ROM framework and the benefit against high-fidelity simulations. Steady-state CFD simulations in Ref. [15] were solved with a 120-core HPC, requiring about 28,000 CPU-h for the 70 CFD dataset samples. The ROMs were generated using a conventional 8GB GPU. Tables B.7 and B.8 of the Appendix provide details of the model architecture and the training strategy. Once the models are generated, simulation time is invariant to the various



(a) C_P CFD reference. (b) DD model (baseline) C_P prediction. (c) Grad. loss model (GG 2) C_P prediction. (d) Grad. masking model (GG 5) C_P prediction.

(f) DD model (baseline) C_P error against CFD. ε_{C_L}= 1.2%, $\varepsilon_{C_{M_y}}$= 17.4%. (g) Grad. loss model (GG 2) C_P error against CFD. ε_{C_L}= 0.8%, $\varepsilon_{C_{M_y}}$= 0.2%. (h) Grad. masking model (GG 5) C_P error against CFD. ε_{C_L}= 3.3%, $\varepsilon_{C_{M_y}}$= 5.1%.

Fig. 25. Pressure coefficient comparison of the various methods against CFD reference at $M_\infty = 0.81$, $\alpha_\infty = 4.34$ deg, $C_{L_{\text{CFD}}} = 0.417$ and $C_{M_{y\text{CFD}}} = -0.029$ (sample *f* in Fig. 6(b)), for the large model and reduced dataset scenario.



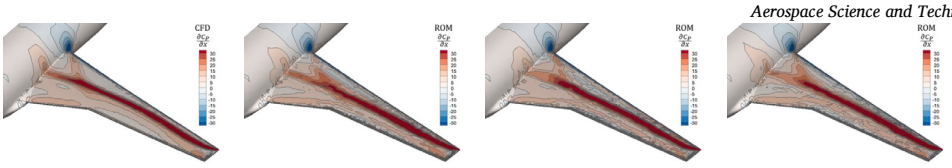
(a) C_P CFD reference. (b) DD model (baseline) C_P prediction. (c) Grad. loss model (GG 2) C_P prediction. (d) Grad. masking model (GG 5) C_P prediction.

(f) DD model (baseline) C_P error against CFD. ε_{C_L}= 1.2%, $\varepsilon_{C_{M_y}}$= 13.6%. (g) Grad. loss model (GG 2) C_P error against CFD. ε_{C_L}= 1.2%, $\varepsilon_{C_{M_y}}$= 3.7%. (h) Grad. masking model (GG 5) C_P error against CFD. ε_{C_L}= 2.9%, $\varepsilon_{C_{M_y}}$= 4.0%.

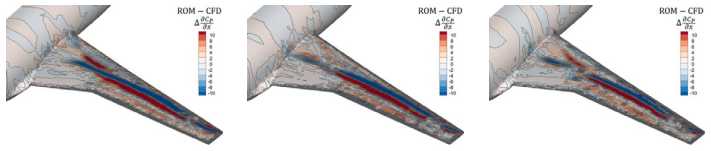
Fig. 26. Pressure coefficient comparison of the various methods against CFD reference at $M_\infty = 0.79$, $\alpha_\infty = 3.24$ deg, $C_{L_{\text{CFD}}} = 0.384$ and $C_{M_{y\text{CFD}}} = -0.038$ (sample *g* in Fig. 6(b)), for the small model and original dataset scenario.

Table 5
Summary of the computing costs involved in steady-state simulations of the CRM.

Computing Task	Cost
1 steady-state CFD run [CPU-h]	~ 400
Dataset of 70 CFD runs [CPU-h]	$\sim 28,000$
ROM training (see Table 6) [GPU-h]	$\sim 2.9 - 7.0$
1 steady-state ROM prediction [GPU-h]	$\sim 1.0 \times 10^{-4}$
Dataset of 70 ROM predictions [GPU-h]	~ 0.006

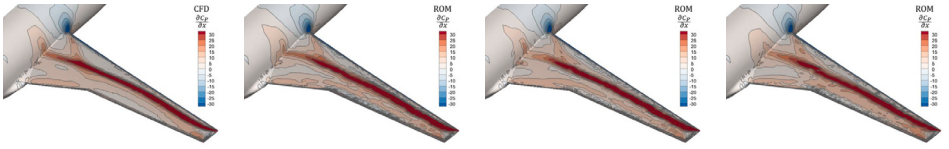


(a) $\frac{\partial C_P}{\partial x}$ CFD reference. (b) DD model (baseline) $\frac{\partial C_P}{\partial x}$ prediction. (c) Grad. loss model (GG 2) $\frac{\partial C_P}{\partial x}$ prediction. (d) Grad. masking model (GG 5) $\frac{\partial C_P}{\partial x}$ prediction.

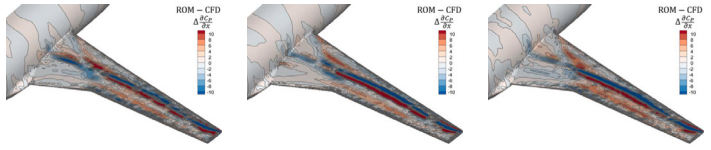


(f) DD model (baseline) $\frac{\partial C_P}{\partial x}$ error against CFD. ε_{C_L}= 2.4%, $\varepsilon_{C_{M_y}}$= 38.9%. (g) Grad. loss model (GG 2) $\frac{\partial C_P}{\partial x}$ error against CFD. ε_{C_L}= 1.7%, $\varepsilon_{C_{M_y}}$= 19.1%. (h) Grad. masking model (GG 5) $\frac{\partial C_P}{\partial x}$ error against CFD. ε_{C_L}= 2.6%, $\varepsilon_{C_{M_y}}$= 33.6%.

Fig. 27. Pressure coefficient streamwise derivative comparison of the various methods against CFD reference at M_∞ = 0.76, α_∞ = 4.42 deg, $C_{L_{\text{CFD}}}$ = 0.414 and $C_{M_{y\text{CFD}}}$ = -0.019 (sample *e* in Fig. 6(b)), for the large model and original dataset scenario.

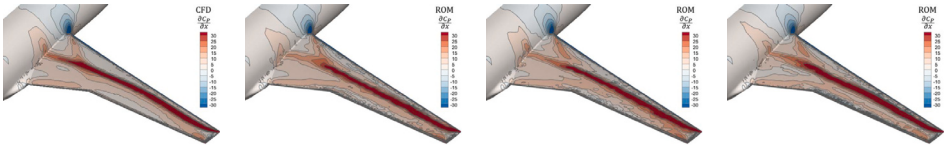


(a) $\frac{\partial C_P}{\partial x}$ CFD reference. (b) DD model (baseline) $\frac{\partial C_P}{\partial x}$ prediction. (c) Grad. loss model (GG 2) $\frac{\partial C_P}{\partial x}$ prediction. (d) Grad. masking model (GG 5) $\frac{\partial C_P}{\partial x}$ prediction.

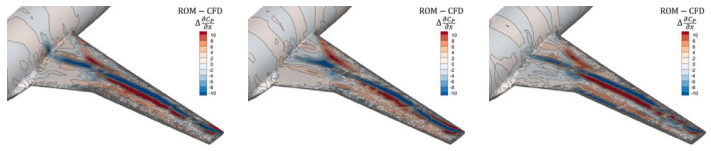


(f) DD model (baseline) $\frac{\partial C_P}{\partial x}$ error against CFD. ε_{C_L}= 0.4%, $\varepsilon_{C_{M_y}}$= 29.6%. (g) Grad. loss model (GG 2) $\frac{\partial C_P}{\partial x}$ error against CFD. ε_{C_L}= 1.1%, $\varepsilon_{C_{M_y}}$= 33.7%. (h) Grad. masking model (GG 5) $\frac{\partial C_P}{\partial x}$ error against CFD. ε_{C_L}= 0.0%, $\varepsilon_{C_{M_y}}$= 25.8%.

Fig. 28. Pressure coefficient streamwise derivative comparison of the various methods against CFD reference at M_∞ = 0.76, α_∞ = 4.42 deg, $C_{L_{\text{CFD}}}$ = 0.414 and $C_{M_{y\text{CFD}}}$ = -0.019 (sample *e* in Fig. 6(b)), for the large model and reduced dataset scenario.

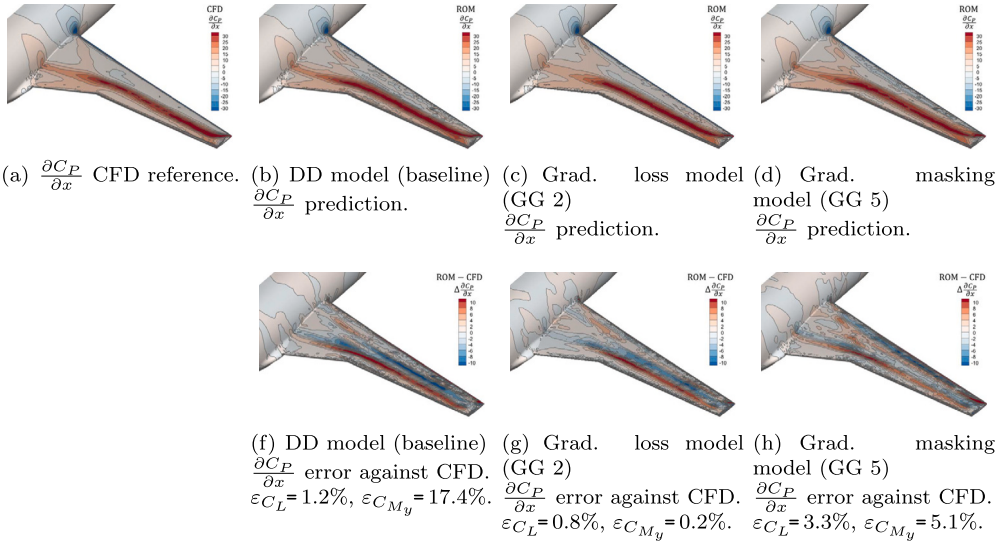


(a) $\frac{\partial C_P}{\partial x}$ CFD reference. (b) DD model (baseline) $\frac{\partial C_P}{\partial x}$ prediction. (c) Grad. loss model (GG 2) $\frac{\partial C_P}{\partial x}$ prediction. (d) Grad. masking model (GG 5) $\frac{\partial C_P}{\partial x}$ prediction.



(f) DD model (baseline) $\frac{\partial C_P}{\partial x}$ error against CFD. ε_{C_L}= 1.1%, $\varepsilon_{C_{M_y}}$= 33.8%. (g) Grad. loss model (GG 2) $\frac{\partial C_P}{\partial x}$ error against CFD. ε_{C_L}= 1.7%, $\varepsilon_{C_{M_y}}$= 32.4%. (h) Grad. masking model (GG 5) $\frac{\partial C_P}{\partial x}$ error against CFD. ε_{C_L}= 0.4%, $\varepsilon_{C_{M_y}}$= 31.9%.

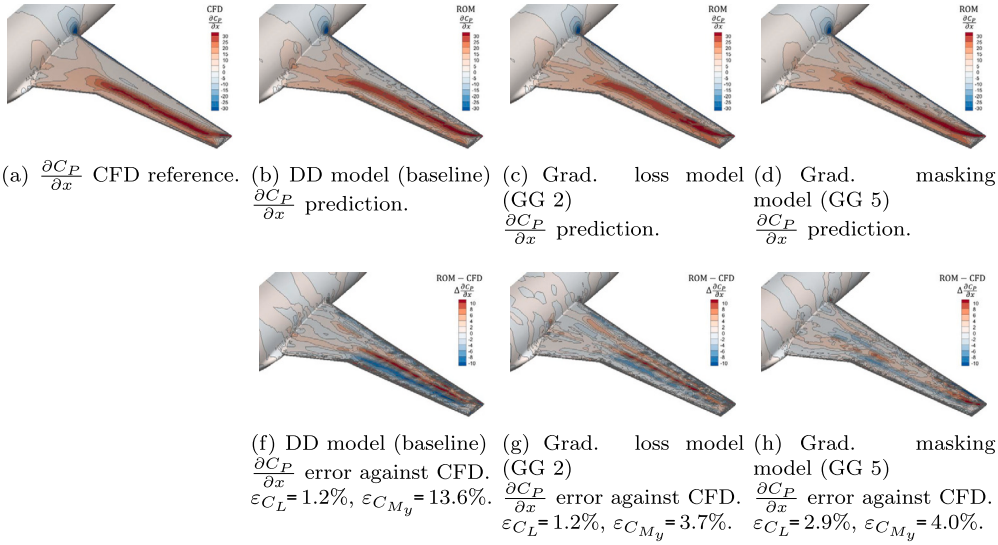
Fig. 29. Pressure coefficient streamwise derivative comparison of the various methods against CFD reference at M_∞ = 0.76, α_∞ = 4.42 deg, $C_{L_{\text{CFD}}}$ = 0.414 and $C_{M_{y\text{CFD}}}$ = -0.019 (sample *e* in Fig. 6(b)), for the small model and original dataset scenario.



(a) $\frac{\partial C_P}{\partial x}$ CFD reference. (b) DD model (baseline) $\frac{\partial C_P}{\partial x}$ prediction. (c) Grad. loss model (GG 2) $\frac{\partial C_P}{\partial x}$ prediction. (d) Grad. masking model (GG 5) $\frac{\partial C_P}{\partial x}$ prediction.

(f) DD model (baseline) $\frac{\partial C_P}{\partial x}$ error against CFD. $\varepsilon_{C_L} = 1.2\%$, $\varepsilon_{C_{M_y}} = 17.4\%$. (g) Grad. loss model (GG 2) $\frac{\partial C_P}{\partial x}$ error against CFD. $\varepsilon_{C_L} = 0.8\%$, $\varepsilon_{C_{M_y}} = 0.2\%$. (h) Grad. masking model (GG 5) $\frac{\partial C_P}{\partial x}$ error against CFD. $\varepsilon_{C_L} = 3.3\%$, $\varepsilon_{C_{M_y}} = 5.1\%$.

Fig. 30. Pressure coefficient streamwise derivative comparison of the various methods against CFD reference at $M_\infty = 0.81$, $\alpha_\infty = 4.34$ deg, $C_{L_{\text{CFD}}} = 0.417$ and $C_{M_{y\text{CFD}}} = -0.029$ (sample *f* in Fig. 6(b)), for the large model and reduced dataset scenario.



(a) $\frac{\partial C_P}{\partial x}$ CFD reference. (b) DD model (baseline) $\frac{\partial C_P}{\partial x}$ prediction. (c) Grad. loss model (GG 2) $\frac{\partial C_P}{\partial x}$ prediction. (d) Grad. masking model (GG 5) $\frac{\partial C_P}{\partial x}$ prediction.

(f) DD model (baseline) $\frac{\partial C_P}{\partial x}$ error against CFD. $\varepsilon_{C_L} = 1.2\%$, $\varepsilon_{C_{M_y}} = 13.6\%$. (g) Grad. loss model (GG 2) $\frac{\partial C_P}{\partial x}$ error against CFD. $\varepsilon_{C_L} = 1.2\%$, $\varepsilon_{C_{M_y}} = 3.7\%$. (h) Grad. masking model (GG 5) $\frac{\partial C_P}{\partial x}$ error against CFD. $\varepsilon_{C_L} = 2.9\%$, $\varepsilon_{C_{M_y}} = 4.0\%$.

Fig. 31. Pressure coefficient streamwise derivative comparison of the various methods against CFD reference at $M_\infty = 0.79$, $\alpha_\infty = 3.24$ deg, $C_{L_{\text{CFD}}} = 0.384$ and $C_{M_{y\text{CFD}}} = -0.038$ (sample *g* in Fig. 6(b)), for the small model and original dataset scenario.

frameworks, as the gradient-guided embedding is just involved in the training process and only the DD model is called in new predictions. As a result, each new aerodynamic prediction is obtained in less than a second, rather than approximately 3 hours in CFD, leading to a speed-up of well over 99.9% for the pure predictive step.

The training cost varied depending on the dataset, model size and GG approach. Table 6 reports details of the cost involved for the distinct ROM approaches. The reported values are based on the worst case, consisting of the large model and the original training samples (scenario 1). The number of epochs were dictated by ensuring the convergence of each model training. Note how the better performing gradient-loss method (GG approach 2) was also found more computationally intensive than the others. However, the additional burden from the GG embedding is only involved during the training phase. Finally, the computing cost to train the models for scenarios 2 (large model and reduced dataset) and 3 (small model and original dataset) were in general reduced to $\sim 48\%$ and $\sim 74\%$, respectively, compared to the cost required in the first scenario.

5. Conclusions

Deep learning-based data-driven models are renowned for their inference capabilities in highly nonlinear spaces. However, these methods depend heavily on sufficient data to adequately capture trends across the entire design space. In engineering analyses, for example CFD simulations, generating the preliminary data required to train these models is computationally expensive. Incorporating physics terms into data-driven models enhances their ability to understand physical solutions, thereby reducing the need for extensive preliminary data without compromising accuracy. To achieve this, we developed five formulations to embed solution gradients into our previously implemented geometric deep-learning multi-resolution framework. Leveraging gradients is ad-

Table 6
Training strategies for the various GG modelling approaches. Training cost is for the large model size and original dataset. The training process comprised 800 epochs unless otherwise stated.

Parameter	Value
DD model (baseline)	
training cost [GPU-h]	~ 2.9
GG model 1 (grad. targets)	
training cost [GPU-h]	~ 4.5
GG model 2 (grad. loss)	
Training epochs	1150
training cost [GPU-h]	~ 7.0
GG model 3 (grad. latent state)	
training cost [GPU-h]	~ 3.0
GG model 4 (grad. weighting)	
training cost [GPU-h]	~ 3.0
GG model 5 (grad. masking)	
training cost [GPU-h]	~ 3.0

vantageous because they efficiently extract critical surface flow features while remaining computationally cost-effective.

To investigate the five distinct gradient-informed models and assess their predictive capability, we considered three scenarios differing in dataset size and model architecture. The first scenario involved a substantial dataset (in the context of costly preliminary simulations) and a large model architecture. In the second scenario, the number of training samples was significantly reduced while maintaining the same model architecture as in the first scenario. The third scenario featured a reduced model size compared to the first scenario. As a relevant engineering application, we addressed the aerodynamic prediction of the NASA CRM wing/body configuration under a range of transonic flow conditions. Nonetheless, our methodology is advantageous for its adequacy to a wide range of problems, particularly in spatial domain discretisations.

The interpretation of the results led to the following conclusions. In scenarios with access to a large dataset and the capability to build a large model architecture, a good predictive model can be obtained without the necessity of embedding gradient terms into the model architecture. This aligns with expectations and is consistent with many applications involving significantly more extensive datasets (millions of data). However, in typical engineering design environments, where large datasets are rare, the findings from the other two scenarios become more relevant to the aeronautical sector. Significant improvements were observed with the inclusion of gradient terms in cases with limited data (second scenario) or constrained GPU memory (third scenario). Regular data-driven models exhibited poor performance under these conditions, whereas models incorporating gradient terms performed satisfactorily.

Among the various gradient-informed models, two approaches stood out: the output-gradient loss (approach 2) and the loss masking (approach 5). The gradient-loss model consistently outperformed all other models, demonstrating significant prediction improvements across the three scenarios, albeit at a higher training cost. Consequently, the gradient-loss method is preferred when optimal prediction accuracy is of primary concern. The loss masking approach can provide slightly better accuracy than the purely data-driven model, particularly for the pitching moment, although clearly not to the same extent as the gradient-loss model. Due to the lower computational demand for model generation, the loss masking approach is the recommended choice when a balance between improved prediction accuracy (with respect to the purely data-driven model) and rapid implementation (with respect to the gradient-loss model) is desired.

This work demonstrates on a relevant use case the practical benefits of integrating data-driven models with physical terms. The enhanced predictive accuracy highlights how the incorporation of physical information can compensate for smaller datasets. As an exploratory study, it sets the stage for further investigations into other forms of physics-based terms or simplified equations, for application in fluid dynamics, structural analysis or any field where mesh-based simulations are involved. Although these approaches may currently seem elusive, continued efforts in this direction are well justified.

CRediT authorship contribution statement

David Massegur: Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Andrea Da Ronch:** Writing – review & editing, Supervision, Resources, Project administration.

Funding sources

This work received no funding sources.

Declaration of competing interest

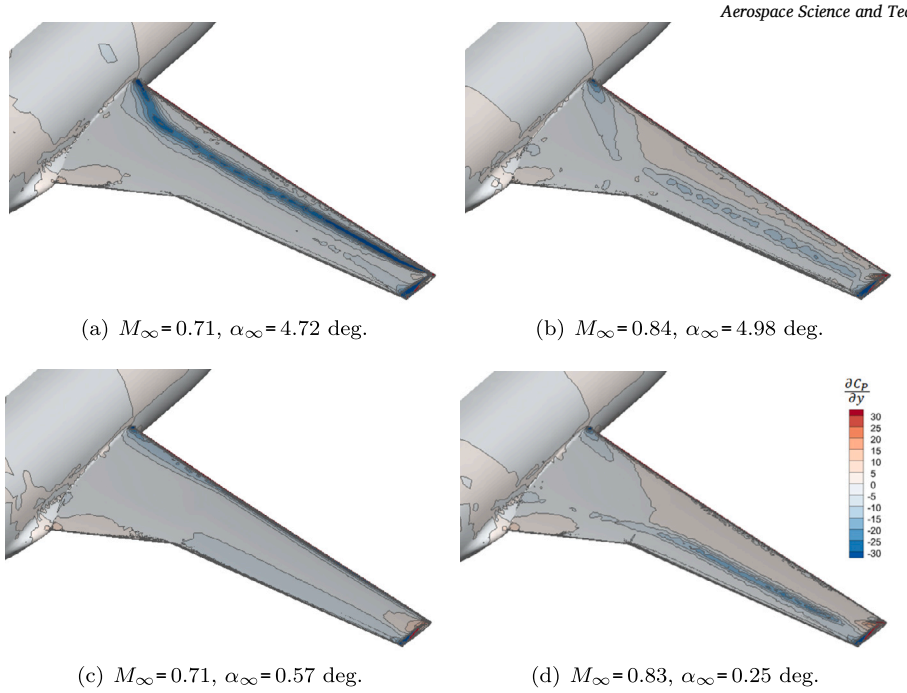
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors would like to thank Gabriele Immordino for providing the CRM reference dataset.

Appendix A. Extended CFD gradient analysis

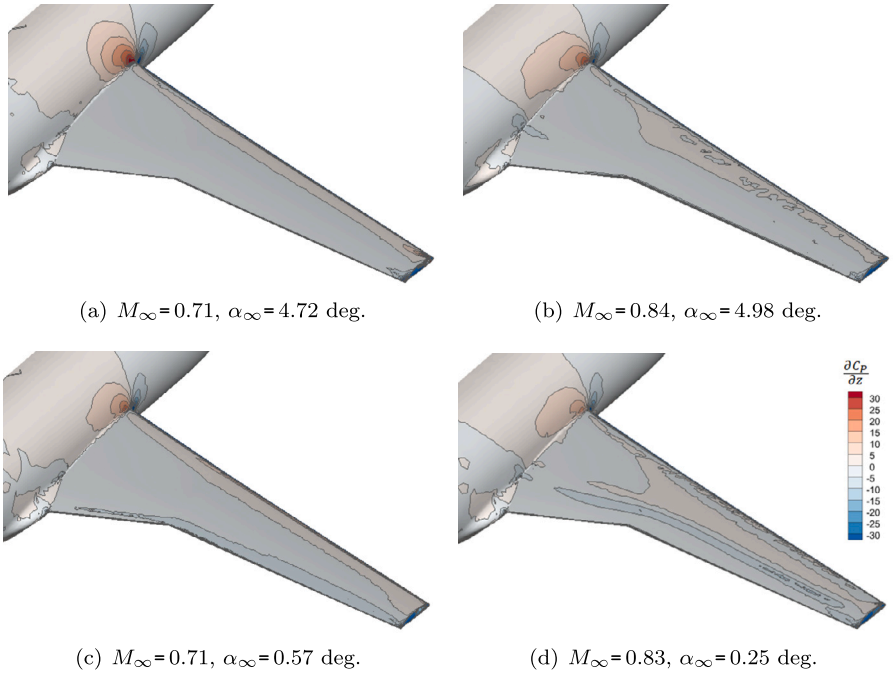
Figs. A.32 and A.33 illustrate the crosswise $\frac{\partial C_P}{\partial y}$ and vertical $\frac{\partial C_P}{\partial z}$ derivatives for the first four selected samples of the reference dataset. These correspond to the other two components of the pressure gradient on the CRM surface. The streamwise $\frac{\partial C_P}{\partial x}$ derivative was reported in Fig. 8. We observe how the fields for these two derivatives are less interesting, especially on the wing. This comparison justifies the choice of the streamwise derivative for the analysis of the results.



(a) $M_\infty = 0.71$, $\alpha_\infty = 4.72$ deg. (b) $M_\infty = 0.84$, $\alpha_\infty = 4.98$ deg.

(c) $M_\infty = 0.71$, $\alpha_\infty = 0.57$ deg. (d) $M_\infty = 0.83$, $\alpha_\infty = 0.25$ deg.

Fig. A.32. Crosswise derivative of the pressure coefficient $\frac{\partial C_P}{\partial y}$ distribution for the CFD samples labelled with the same letters in Fig. 6 Panel (b).



(a) $M_\infty = 0.71$, $\alpha_\infty = 4.72$ deg. (b) $M_\infty = 0.84$, $\alpha_\infty = 4.98$ deg.

(c) $M_\infty = 0.71$, $\alpha_\infty = 0.57$ deg. (d) $M_\infty = 0.83$, $\alpha_\infty = 0.25$ deg.

Fig. A.33. Vertical derivative of the pressure coefficient $\frac{\partial C_P}{\partial z}$ distribution for the same selected CFD cases.

Table B.7
Architecture of the steady-state GCN-MM-AE model to construct the aerodynamic envelope for the CRM model. Refer to Fig. 3 for the diagram of this NN architecture.

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

Table B.8
Training strategy adopted to generate the steady-state GCN-MM-AE model.

Parameter	Value
Trainable parameters	174,460, 44,464
Dataset samples	70
Training set	40 , 20
Batch size	1
Loss function	MSE
Optimiser	Adam
Starting learning rate	0.0009
Learning rate schedule	0.333 / 100 epochs
GPU machine	`NVIDIA GeForce RTX 2070`

Appendix B. Data-driven model architecture and hyper-parameters

The baseline data-driven model architecture was based on the GCN-MM-AE developed in Ref. [25]. In Table B.7 we report the final architecture adopted for the analysis of the CRM test case. A diagram of the model architecture is reported in Fig. 3. The layers involving sequential GCN and MM blocks for the encoder and decoder branches are laid out. The PReLU nonlinear activation function was used at the output of each GCN layer, except for the last layer, which outputs the target fields. The tensor dimensions at each layer output are also provided, arranged in $m \times n_{\text{mml}} \times c_l$, with m the batch size, n_{mml} the grid size at the corresponding MM level and c_l the state size (or channels) from each layer output.

The learnable parameters (kernel size) involved in each NN layer (either GCN or PReLU) are for the original model size adopted in the results sections 4.1 and 4.2. The reduced model adopted in section 4.3 was constructed by halving the number of channels at each NN layer. The total number of ROM parameters decreased from 174,460 to 44,464, resulting in a reduction of the GPU memory requirement by a factor of ~ 4 approximately.

Table B.8 reports more details on the training and optimisation strategies adopted. The first-order gradient-based algorithm Adam [18] was chosen to minimise the mean squared error (MSE) loss functions [3]. Both the inputs and the gradients were standardised with the mean fields and normalised with the standard deviation to ensure better training convergence.

Table C.9
Training strategies for the various GG modelling approaches. Training cost is for the large model size and original dataset. The training process comprised 800 epochs unless otherwise stated.

Parameter	Value
GG model 2 (grad. loss)	
lagrange multiplier λ_2, Eq. (7)	0.001
GG model 3 (grad. latent state)	
lagrange multiplier λ_3, Eq. (9)	0.001
GG model 4 (grad. weighting)	
lagrange multiplier λ_4, Eq. (10)	1.0
GG model 5 (grad. masking)	
lagrange multiplier λ_5, Eq. (11)	0.01
mask threshold δ_5, Eq. (12)	2.0

Appendix C. Gradient-based hyper-parameters

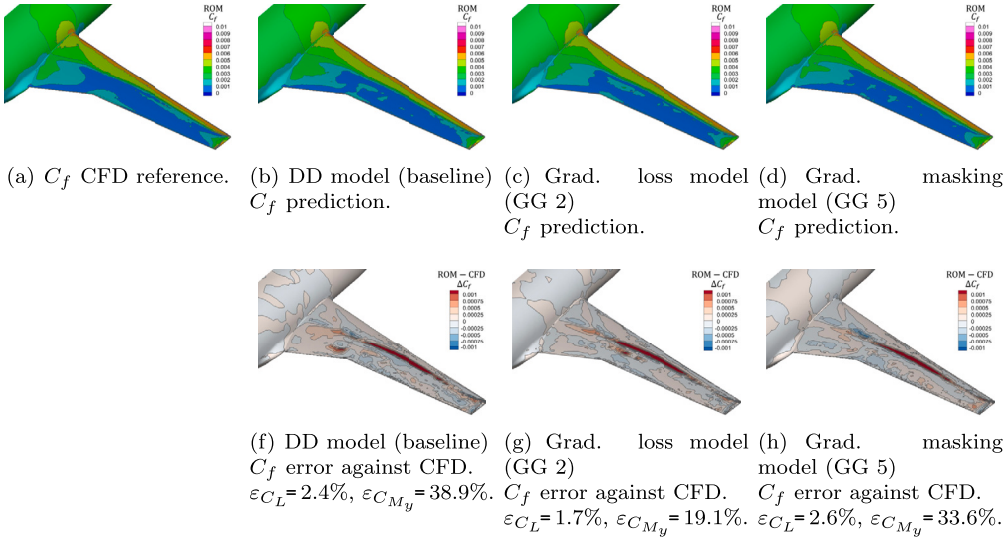
The only hyper-parameters involved in each GG approach are the lagrange multipliers to combine the physics and the data-driven terms in the respective loss functions, as described in Section 2.2, as well as the number of epochs reported in Section 4.6. In particular, the lagrange multipliers were fine tuned in order that the values of the separate data-driven and physics loss terms were of similar magnitude. The adopted hyper-parameter values for each GG method are reported in Table C.9.

Appendix D. Skin-friction predictions

Section 4.5 presented a comprehensive analysis of the distributed pressure predictions, which is the most interesting aerodynamic quantity. We now present results for the skin friction, which is computed as the norm of the shear-stress components predicted by the models: $C_f = |||[C_{\tau_x}, C_{\tau_y}, C_{\tau_z}]||_2$. Fig. D.34 illustrates a comparison of the skin frictions for sample e again for the various modelling approaches and baseline scenario. Furthermore, Fig. D.35 is for the streamwise derivative of the predicted skin friction. For reference, the C_P analysis for the same configurations are in Figs. 22 and 27. We observe how the skin friction variations are two orders of magnitude lower than the pressure, which also reflect on the magnitude of the gradients. As a result, the skin friction predictions are in good agreement with the reference solution and no significant differences are observed among the various models. We conclude that, for this case, the shear-stress gradients have less influence in the model training because of being less significant than the pressure gradients.

Data availability

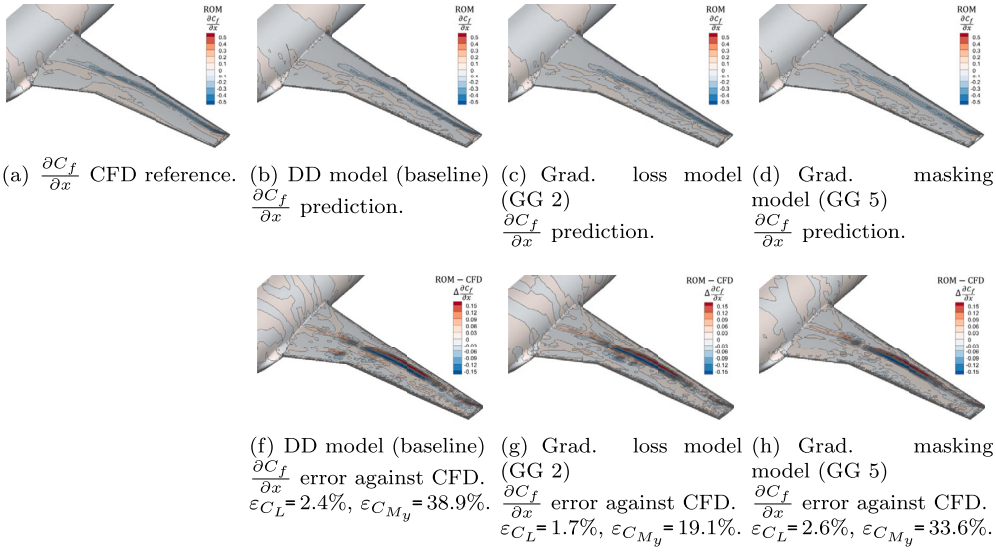
Data will be made available on request.



(a) C_f CFD reference. (b) DD model (baseline) C_f prediction. (c) Grad. loss model (GG 2) C_f prediction. (d) Grad. masking model (GG 5) C_f prediction.

(f) DD model (baseline) C_f error against CFD. ε_{C_L}= 2.4%, $\varepsilon_{C_{M_y}}$= 38.9%. (g) Grad. loss model (GG 2) C_f error against CFD. ε_{C_L}= 1.7%, $\varepsilon_{C_{M_y}}$= 19.1%. (h) Grad. masking model (GG 5) C_f error against CFD. ε_{C_L}= 2.6%, $\varepsilon_{C_{M_y}}$= 33.6%.

Fig. D.34. Skin friction coefficient comparison of the various methods against CFD reference at $M_\infty = 0.76$, $\alpha_\infty = 4.42$ deg, $C_{L_{\text{CFD}}} = 0.414$ and $C_{M_{y\text{CFD}}} = -0.019$ (sample *e* in Fig. 6(b)), for the large model and original dataset scenario.



(a) $\frac{\partial C_f}{\partial x}$ CFD reference. (b) DD model (baseline) $\frac{\partial C_f}{\partial x}$ prediction. (c) Grad. loss model (GG 2) $\frac{\partial C_f}{\partial x}$ prediction. (d) Grad. masking model (GG 5) $\frac{\partial C_f}{\partial x}$ prediction.

(f) DD model (baseline) $\frac{\partial C_f}{\partial x}$ error against CFD. ε_{C_L}= 2.4%, $\varepsilon_{C_{M_y}}$= 38.9%. (g) Grad. loss model (GG 2) $\frac{\partial C_f}{\partial x}$ error against CFD. ε_{C_L}= 1.7%, $\varepsilon_{C_{M_y}}$= 19.1%. (h) Grad. masking model (GG 5) $\frac{\partial C_f}{\partial x}$ error against CFD. ε_{C_L}= 2.6%, $\varepsilon_{C_{M_y}}$= 33.6%.

Fig. D.35. Skin friction coefficient streamwise derivative comparison of the various methods against CFD reference at $M_\infty = 0.76$, $\alpha_\infty = 4.42$ deg, $C_{L_{\text{CFD}}} = 0.414$ and $C_{M_{y\text{CFD}}} = -0.019$ (sample *e* in Fig. 6(b)), for the large model and original dataset scenario.

References

[1] J. Anderson, Fundamentals of Aerodynamics, McGraw-Hill Education, 2016, https://books.google.co.uk/books?id=D1ZojgEACAAJ.

[2] E. Ang, B.F. Ng, Physics-informed neural networks for flow around airfoil, in: AIAA SCITECH 2022 Forum, American Institute of Aeronautics and Astronautics Inc, AIAA, 2022, https://arc.aiaa.org/doi/abs/10.2514/6.2022-0187, https://arc.aiaa.org/doi/pdf/10.2514/6.2022-0187.

[3] P. Bickel, K. Doksum, Mathematical Statistics: Basic Ideas and Selected Topics, Volume I, second edition, Chapman & Hall/CRC Texts in Statistical Science, CRC Press, 2015, https://books.google.co.uk/books?id=y5i9BwAAQBAJ.

[4] M.M. Bronstein, J. Bruna, T. Cohen, P. Veličković, Geometric deep learning grids, groups, graphs, geodesics, and gauges, https://arxiv.org/abs/2104.13478, 2021.

[5] M.M. Bronstein, J. Bruna, Y. Lecun, A. Szlam, P. Vandergheynst, Geometric deep learning: going beyond Euclidean data, IEEE Signal Process. Mag. 34 (2017) 18–42, https://doi.org/10.1109/MSP.2017.2693418.

[6] D. Clifford, A. Hossein Modarres Aval, A. Da Ronch, Model Order Reduction for Nonlinear Aeroelastic Dynamical Systems Using Automatic Differentiation, in: International Forum on Aeroelasticity and Structural Dynamics, IFASD, The Hague, the Netherlands, 2024.

[7] L. Cun, J. Henderson, Y. Le Cun, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Handwritten Digit Recognition with a Back-Propagation Network, Technical Report, 1989.

[8] J.P. Fielding, Introduction to Aircraft Design, 2 ed., Cambridge Aerospace Series, Cambridge University Press, 2017.

[9] H. Gao, L. Sun, J.X. Wang, Phygeonet: physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain, J. Comput. Phys. 428 (2021) 110079, https://doi.org/10.1016/j.jcp.2020.110079, https://www.sciencedirect.com/science/article/pii/S0021999120308536.

[10] B. Glaz, L. Liu, P.P. Friedmann, Reduced-order nonlinear unsteady aerodynamic modeling using a surrogate-based recurrence framework, AIAA J. 48 (2010) 2418–2429, https://doi.org/10.2514/1.J050471.

[11] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: surpassing human-level performance on ImageNet classification, http://arxiv.org/abs/1502.01852, 2015.

[12] D. Hines, P. Bekemeyer, Graph neural networks for the prediction of aircraft surface pressure distributions, Aerosp. Sci. Technol. 137 (2023), https://doi.org/10.1016/j.ast.2023.108268.

[13] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (2006) 502–504, https://doi.org/10.1126/science.1129198.

[14] J. Hu, W. Zhang, Flow field modeling of airfoil based on convolutional neural networks from transform domain perspective, Aerosp. Sci. Technol. 136 (2023) 108198, https://doi.org/10.1016/j.ast.2023.108198, https://www.sciencedirect.com/science/article/pii/S1270963823000950.

[15] G. Immordino, A. Da Ronch, M. Righi, Steady-state transonic flowfield prediction via deep-learning framework, AIAA J. 62 (2024) 1915–1931, https://doi.org/10.2514/1.J063545.

[16] G.R. Joldes, H.A. Chowdhury, A. Wittek, B. Doyle, K. Miller, Modified moving least squares with polynomial bases for scattered data approximation, Appl. Math. Comput. 266 (2015) 893–902, https://doi.org/10.1016/j.amc.2015.05.150, https://www.sciencedirect.com/science/article/pii/S0096300315007924.

[17] G. Kenway, G. Kennedy, J. Martins, Aerostructural optimization of the common research model configuration, https://doi.org/10.2514/6.2014-3274, 2014.

[18] D.P. Kingma, J.L. Ba, Adam: a method for stochastic optimization, in: 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 2015, pp. 1–15.

[19] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, in: 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings, 2017, pp. 1–14.

[20] D.M. Luchtenburg, B.R. Noack, M. Schlegel, An introduction to the POD Galerkin method for fluid flows with analytical examples and MATLAB source codes, Technical Report, Berlin Institute of Technology, Department for Fluid Dynamics and Engineering Acoustics, Berlin, 2009.

[21] J.L. Lumley, in: John L. Lumley (Ed.), Stochastic Tools in Turbulence, Academic Press New York, 1970.

[22] C. Mancinelli, M. Livesu, E. Puppo, Gradient field estimation on triangle meshes, in: Italian Chapter Conference 2018 - Smart Tools and Apps in Computer Graphics, STAG 2018, Eurographics Association, 2018, pp. 87–96.

[23] D. Massegur, D. Clifford, A. Da Ronch, R. Lombardi, M. Panzeri, Low-dimensional models for aerofoil icing predictions, Aerosp. J. 10 (2023) 444, https://doi.org/10.3390/aerospace10050444, https://www.mdpi.com/2226-4310/10/5/444.

[24] D. Massegur, D. Clifford, A. Da Ronch, S. Symon, Comparing Reduced Order Model Forms for Nonlinear Dynamical Systems, in: 33rd Congress of the International Council of the Aeronautical Sciences, ICAS, Stockholm, Sweden, 2022, https://www.icas.org/ICAS_ARCHIVE/ICAS2022/data/papers/ICAS2022_0399_paper.pdf.

[25] D. Massegur, A. Da Ronch, Graph convolutional multi-mesh autoencoder for steady transonic aircraft aerodynamics, Mach. Learn.: Sci. Technol. 5 (2024) 025006, https://doi.org/10.1088/2632-2153/ad36ad, https://iopscience.iop.org/article/10.1088/2632-2153/ad36ad.

[26] D. Massegur, A. Da Ronch, Recurrent graph convolutional multi-mesh autoencoder for unsteady transonic aerodynamics, J. Fluids Struct. 131 (2024) 104202, https://doi.org/10.1016/j.jfluidstructs.2024.104202, https://www.sciencedirect.com/science/article/pii/S0889974624001373.

[27] S.F. McCormick, Multigrid Methods, Society for Industrial and Applied Mathematics, 1987, https://epubs.siam.org/doi/abs/10.1137/1.9781611971057, https://epubs.siam.org/doi/pdf/10.1137/1.9781611971057.

[28] M. Morimoto, K. Fukami, K. Zhang, A.G. Nair, K. Fukagata, Convolutional neural networks for fluid flow analysis: toward effective metamodeling and low dimensionalization, Theor. Comput. Fluid Dyn. 35 (2021) 633–658, https://doi.org/10.1007/s00162-021-00580-0.

[29] A.Y. Ng, L1 and L2 regularisation comparisation, in: Proceedings of the 21 st International Conference on Machine Learning, 2004.

[30] C. Ning, W. Zhang, Mha-net: multi-source heterogeneous aerodynamic data fusion neural network embedding reduced-dimension features, Aerosp. Sci. Technol. 145 (2024) 108908, https://doi.org/10.1016/j.ast.2024.108908, https://www.sciencedirect.com/science/article/pii/S1270963824000415.

[31] S.B. Pope, Turbulent Flows, Cambridge University Press, 2000.

[32] G. Quaranta, P. Masarati, P. Mantegazza, A conservative mesh-free approach for fluid-structure interface problems, in: CIMNE (Ed.), Int. Conf. on Computational Methods for Coupled Problems in Science and Engineering, Coupled Problems 2005, ECCOMAS, Barcelona, 2005, pp. 1–22.

[33] M. Raissi, P. Perdikaris, G.E. Karniadakis, Physics informed deep learning (Part I): data-driven solutions of nonlinear partial differential equations, http://arxiv.org/abs/1711.10561, 2017.

[34] M.B. Rivers, NASA Common Research Model: A History and Future Plans, in: AIAA Scitech 2019 Forum, NASA Langley Research Center, AIAA, San Diego, CA, 2019.

[35] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (1986) 533–536, https://doi.org/10.1038/323533a0, http://www.nature.com/articles/323533a0.

[36] D.E. Rumelhart, J.L. McClelland, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations, MIT Press, 1986.

[37] J. Schmidhuber, Deep learning in neural networks: an overview, https://doi.org/10.1016/j.neunet.2014.09.003, http://arxiv.org/abs/1404.7828, 2014.

[38] N. Srivastava, G. Hinton, A. Krizhevsky, R. Salakhutdinov, Dropout: a Simple Way to Prevent Neural Networks from Overfitting, Technical Report, 2014.

[39] S. Sureshbabu, F. Tejero, F. Sanchez-Moreno, D.G. MacManus, C. Sheaf, Deep-learning methods for non-linear transonic flow-field prediction, in: AIAA AVIATION 2023 Forum, American Institute of Aeronautics and Astronautics, San Diego, California, 2023.

[40] N.J. Taylor, M. Gammon, J.C. Vassberg, The NASA common research model: a geometry-handling perspective, in: 46th AIAA Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics Inc, AIAA, 2016.

[41] N. Thuerey, P. Holl, M. Mueller, P. Schnell, F. Trost, K. Um, Physics-based deep learning, https://physicsbaseddeeplearning.org, 2021.

[42] F.A. Viana, R.G. Nascimento, A. Dourado, Y.A. Yucesan, Estimating model inadequacy in ordinary differential equations with physics-informed neural networks, Comput. Struct. 245 (2021) 106458, https://doi.org/10.1016/j.compstruc.2020.106458.

[43] J. Willard, X. Jia, S. Xu, M. Steinbach, V. Kumar, Integrating scientific knowledge with machine learning for engineering and environmental systems, ACM Comput. Surv. 55 (4) (2022) 1–37, https://doi.org/10.1145/3514228.

[44] Y.A. Yucesan, F.A.C. Viana, A Physics-Informed Neural Network for Wind Turbine Main Bearing Fatigue, Int. J. Progn. Health Manage. 11 (1) (2020) 1–17, https://doi.org/10.36001/ijphm.2020.v11i1.2594, https://papers.phmsociety.org/index.php/ijphm/article/view/2594.

[45] W. Zhang, X. Peng, J. Kou, X. Wang, Heterogeneous data-driven aerodynamic modeling based on physical feature embedding, Chin. J. Aeronaut. 37 (2024) 1–6, https://doi.org/10.1016/j.cja.2023.11.010, https://www.sciencedirect.com/science/article/pii/S1000936123003941.