Aerospace Science and Technology 165 (2025) 110516

Contents lists available at ScienceDirect

Aerospace Science and Technology

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

Original article

Spatio-temporal graph convolutional autoencoder for transonic wing pressure distribution forecasting

Gabriele Immordino [a,b,*], Andrea Vaiuso [b], Andrea Da Ronch [a], Marcello Righi [b]

[a] Faculty of Engineering and Physical Sciences, University of Southampton, Southampton, United Kingdom
[b] School of Engineering, Zurich University of Applied Sciences ZHAW, Winterthur, Switzerland

ARTICLE INFO

Communicated by Mehdi Ghoreyshi

ABSTRACT

This study presents a framework for predicting unsteady transonic wing pressure distributions due to pitch and plunge movement, integrating an autoencoder architecture with graph convolutional networks and graph-based temporal layers to model time dependencies. The framework compresses high-dimensional pressure distribution data into a lower-dimensional latent space using an autoencoder, ensuring efficient data representation while preserving essential features. Within this latent space, graph-based temporal layers are employed to predict future wing pressures based on past data, effectively capturing temporal dependencies and improving predictive accuracy. Four different temporal schemes have been tested, where the spatio-temporal graph convolutional network achieved the best accuracy thanks to convolution in both time and space. This combined approach leverages the strengths of autoencoders for dimensionality reduction, graph convolutional networks for handling unstructured grid data, and temporal layers for modeling time-based sequences. To benchmark the efficacy of the framework, a comparison with the Dynamic Mode Decomposition with control technique is performed. Validation is conducted using the Benchmark Super Critical Wing test case at Mach 0.74, demonstrating that the proposed approach achieves accuracy comparable to high-fidelity computational fluid dynamics simulations while significantly reducing prediction time. This work underscores the potential of the developed framework as a scalable, efficient, and robust solution for the analysis of nonlinear unsteady aerodynamic phenomena.

1. Introduction

The complexity of aerodynamic analysis poses a significant challenge across various engineering applications. Accurately predicting challenging physical phenomena involves capturing detailed variations that arise from the complex interaction of multiple forces. Traditional computational fluid dynamics (CFD) methods are effective in many scenarios, but often require substantial computational resources and may struggle with accurately representing dynamic and unsteady phenomena under specific flow conditions [1]. These limitations highlight the need for more efficient and robust approaches.

To alleviate these challenges, reduced-order models (ROMs) have been widely adopted. Among these, Proper Orthogonal Decomposition (POD) [2] and Dynamic Mode Decomposition (DMD) [3] offer efficient approximations of high-dimensional flowfields through low-rank representations. The DMD framework, in particular, excels at extracting coherent spatio-temporal patterns from unsteady flows. Its extension, Dynamic Mode Decomposition with control (DMDc) [4], explicitly incorporates motion inputs, enabling effective reconstruction and forecasting of flow responses under dynamic excitation. DMDc has been successfully applied in various fluid dynamic contexts, including aerodynamic [5–7] and aeroelastic [8,9] applications.

Recent advancements in machine learning (ML) offer alternative paths to surrogate modeling by enabling data-driven learning of complex nonlinear patterns. Initial efforts employed deep neural networks to reconstruct or forecast aerodynamic fields [10–13]. However, aerospace engineering problems often rely on non-homogeneous and unstructured grids modeling, which necessitate more advanced ML architectures that can handle this complex data structures.

Geometric deep learning, introduced around 2017 [14], utilizes graph neural networks (GNNs) for graph-structured data [15,16]. GNNs excel in capturing relationships and dependencies within graph nodes, making them ideal for tasks involving topological information [17–19]. Graph Convolutional Networks (GCNs), a specific type of GNN, leverage convolution operations on graphs [20]. GCNs are particularly promising in aerospace engineering, as they can handle data with spatial

* Corresponding author at: Faculty of Engineering and Physical Sciences, University of Southampton, Southampton, United Kingdom.
E-mail address: G.Immordino@soton.ac.uk (G. Immordino).

https://doi.org/10.1016/j.ast.2025.110516
Received 27 January 2025; Received in revised form 16 June 2025; Accepted 18 June 2025
Available online 26 June 2025

Nomenclature

Acronyms

AE	autoencoder
CFD	computational fluid dynamics
GCN	graph convolutional network
GNN	graph neural network
GRU	gated recurrent unit
$LSTM$	long short-term memory
ML	machine learning
MAE	mean absolute error
$MAPE$	mean absolute percentage error
$MWLS$	moving weighted least squares
$RMSE$	root mean square error
$STGCN$	spatio-temporal GCN

Symbols

AoA_∞	freestream angle of attack	deg
C_L	lift coefficient	
C_M	pitching moment coefficient	
C_P	pressure coefficient	
M	Mach number	
θ	pitch angle	rad
$\dot{\theta}$	pitch rate	rad/s
$\ddot{\theta}$	pitch acceleration	rad/s^2
ξ	plunge	m
$\dot{\xi}$	plunge rate	m/s
$\ddot{\xi}$	plunge acceleration	m/s^2

structures and are suitable for modeling complex aerodynamic geometries [21–25]. In fact, while Convolutional Neural Networks (CNNs) perform well on regular grid data like images and texts, GCNs are better suited for irregular domains, such as mesh grids, by applying convolution operations directly on graphs [20]. Indeed, GCNs can directly input raw 3D model mesh data without pre-computation or feature extraction, enhancing predictive capabilities without bias or information loss [17].

Another important challenge concerns the high dimensionality of model input data. As with CNN architectures for image recognition tasks, deep and complex architectures struggle with propagating information over a large number of features. CFD simulations typically involve the use of fine meshes, consisting of a significantly large number of points, increasing both complexity and computational requirements. To manage this, careful data compression is needed to retain only the essential features without losing critical information. Previous studies have demonstrated that dimensionality reduction can be effectively achieved using an autoencoder (AE) architecture [26,25,27,28]. AEs, through their encoding and decoding processes, can learn a compact and efficient representation of the data, ensuring that critical information is preserved while reducing the computational burden [29,30].

Building on our previous study [31], which focused on steady-state problems, we now extend our methodology to address unsteady phenomena. Predicting time-varying pressure distributions relies primarily on capturing temporal dependencies within the data. Recurrent Neural Networks (RNNs), with their ability to track evolving patterns through a hidden state, are particularly well-suited for this task. Their effectiveness in modeling unsteady behaviors and dynamic responses makes them an ideal choice for forecasting time series in aerodynamic applications [32,33]. However, RNNs often struggle with long-term dependencies due to challenges like vanishing gradients [34]. To address these limitations, Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs), both RNN variants, have been developed. LSTMs introduce gates that control the flow of information, making them more effective at learning long-term dependencies [32]. GRUs offer a simpler structure than LSTMs, using fewer gates while still managing to capture long-term dependencies, often with faster training times [33]. LSTMs have been extensively applied in aerodynamic modeling, such as predicting the dynamic response of aeroelastic systems and turbulence [35,36], while GRUs have also proven effective for similar tasks [37–39]. More recently, attention mechanisms have revolutionized time series forecasting by enabling models to focus on the most relevant parts of the input sequence [40,41]. This capability leads to more accurate and robust predictions, as demonstrated for instance by improvements in maintenance scheduling through estimating icing probabilities on wind turbine blades [42], stable long-term fluid dynamics predictions using transformer-style temporal attention [43], and enhanced design and control of hypersonic vehicles by capturing spatiotemporal turbulence characteristics [44]. Attention mechanisms enable models to weigh the importance of different time steps dynamically, thereby improving the ability to model complex temporal patterns. Similarly, Spatio-Temporal Graph Convolution Networks (STGCNs) have shown strong performance in modeling such patterns by processing entire sequences in parallel and applying filters across the time dimension, capturing both short- and long-term dependencies efficiently [45].

To fully harness the potential of these temporal modeling techniques, it is important to recognize that each approach offers distinct benefits depending on the nature of the data and the specific application. With this in mind, our methodology investigates and evaluates several temporal layers—LSTMs, GRUs, attention mechanisms, and STGCNs—to effectively capture the temporal dependencies in our case study. By integrating graph convolutional networks with AEs and temporal layers, our proposed approach leverages the strengths of each method to enhance the prediction of unsteady surface pressure distributions over a transonic wing. This integrated framework not only handles the unstructured grids typical in aerodynamic data through GCNs but also ensures efficient dimensionality reduction through AEs. By comparing different temporal approaches, we aim to provide a comprehensive and scalable solution for complex aerodynamic analyses, delivering more accurate and computationally efficient predictions for unsteady phenomena. To provide a clear reference for performance gains, we benchmarked the proposed framework against a DMDc model due to its relevance and maturity in this field.

The structure of the paper is as follows: Section 2 details the implemented methodology, providing a comprehensive explanation of the architecture and its components. Section 3 describes the test case used to validate the model, focusing on its aerodynamic challenges. Section 4 presents the results, comparing the performance of different architectures designed to address temporal challenges, while evaluating the impact of various temporal layers. Finally, Section 5 summarizes the conclusions drawn from the study and suggests potential future directions for improving the model accuracy and scalability in different aerodynamic scenarios.

2. Methodology

This section outlines the methodology used in developing the model. It begins with an overview of the spatio-temporal graph convolutional autoencoder framework, denoted as AeroNet, then provides an in-depth description of each component of the model.

2.1. AeroNet

The proposed AeroNet framework integrates a GCN-based AE architecture with a temporal prediction layer to model and forecast wing pressure distributions for subsequent timesteps. The encoding and

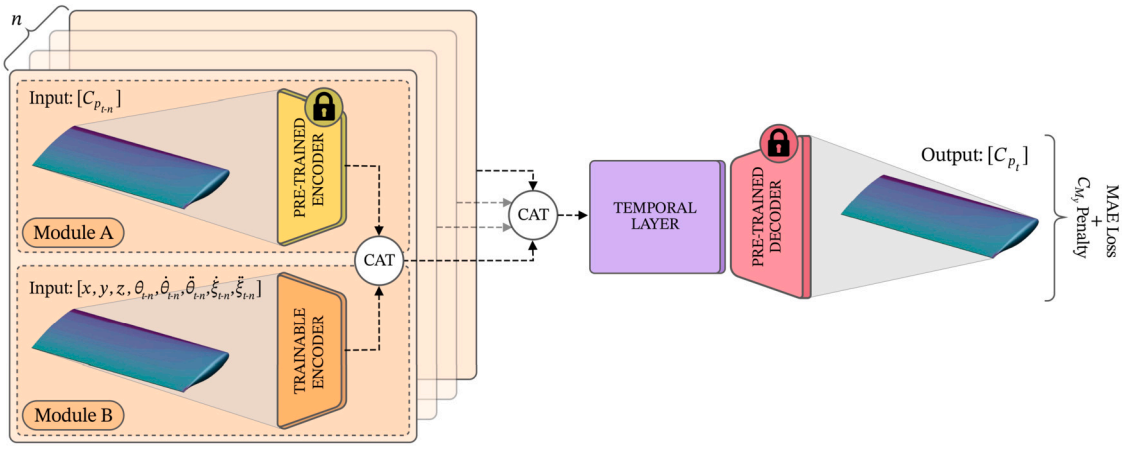


Fig. 1. Overview of the AeroNet architecture for predicting wing pressure distributions. Module A represents the autoregressive component, incorporating previously predicted C_P values, while Module B processes spatial coordinates and motion data from previous timesteps.

decoding modules operate with graph nodes based on the pressure-gradient distribution values across the wing surface, performing pooling and unpooling operations on the input, respectively. Initially, a pre-trained AE is used to compress the pressure distribution data into a lower-dimensional representation, preserving fundamental features while reducing computational complexity. This pre-training step reduces the full model training time and computational costs, enhancing the overall efficiency of the prediction process. After the pooling operation, the reduced-space representation is passed through a temporal prediction layer. This layer is designed to capture the temporal dependencies in the data and forecast wing pressure from a series of previous timesteps to a future one. To account for the complexities caused by shock waves and boundary layer separation, which affect force and moment calculations, a penalty term for the pitching moment coefficient C_{M_y} is added to the Mean Absolute Error (MAE) loss function. This addition is represented as $Loss = \texttt{MAE} + \lambda \cdot C_{M_y}$, with $\lambda = 0.01$ for dimensional consistency. The combination of AE, GCN layers, and temporal modeling enables the framework to provide precise and reliable pressure predictions, which are crucial for analyzing aerodynamic performance.

A visual overview of the model architecture is presented in Fig. 1. The model input features include data from the n previous timesteps ($t-1$, ..., $t-n$): spatial coordinates (x, y, z); pitch $(\theta_{t-n}, \dot{\theta}_{t-n}, \ddot{\theta}_{t-n})$; plunge $(\xi_{t-n}, \ddot{\xi}_{t-n})$; and pressure coefficient $(C_{P_{t-n}})$, with $n = 3$. Finally, the output of the model is represented by the pressure coefficient (C_{P_t}) at the current timestep t. The choice of this sequence length ensures that the model has access to sufficient temporal context to capture the evolution of unsteady aerodynamic features, such as flow separation and shock dynamics, while avoiding the inclusion of redundant or excessive data, which would increase computational complexity without significantly improving accuracy.

Building on this, we developed two different types of architecture: a feedforward model and an autoregressive–moving-average (ARMA) model. In the case of the feedforward model, the inputs consist solely of the coordinates of the wing surface and the prescribed motion at n previous timesteps casted on each graph node (using only Module B in Fig. 1). This model does not incorporate any past predicted pressures into its input, relying purely on the historical spatial and motion data of the wing to make its predictions. Conversely, the ARMA model includes additional information in its input by integrating the pressures predicted at prior timesteps (utilizing both Module A and Module B in Fig. 1). This autoregressive component allows the ARMA model to potentially capture more complex temporal dependencies by considering the history of its own predictions, aiming to enhance the accuracy of the pressure forecasts. Implementing both models serves to evaluate the trade-offs between simplicity and predictive depth: while the feedforward model offers a simpler, stable approach less prone to error accumulation, the ARMA model is designed to capture complex temporal dependencies and unsteady behaviors, potentially enhancing accuracy under dynamic conditions.

To limit error accumulation in the time-marching scheme, we employed a Back-Propagation Through Time (BPTT) algorithm [34] for the total loss calculation, dividing the dataset into mini-sequences. The model processes each sequence consecutively, accumulating error over time. After processing each sequence, the loss function is applied to update the model parameters through backpropagation. A sequence length of three was chosen based on its performance, yielding the best results.

2.1.1. Graph representation

A graph G is defined by a set of nodes N and edges E, where each edge (i, j) represents a directed link from node i to node j. Self-loops occur when $(i, i) \in E$. These connections are represented by an adjacency matrix $\mathbf{A}$, where $\mathbf{A}_{ij} = 1$ if $(i, j) \in E$, and 0 otherwise. Costs for edges can be included by replacing 1 with the cost and using ∞ for absent connections. A path $p(i \to j)$ is a series of steps from i to j, where each step $(h, k) \in E$. A graph is acyclic if no path returns to a starting node, otherwise, it is cyclic.

In our case, the mesh can be represented as a cyclic graph G, where each grid point i serves as a node. Each node carries features such as spatial coordinates (x, y, z), pitch $(\theta, \dot{\theta}, \ddot{\theta})$, plunge $(\dot{\xi}, \ddot{\xi})$, and the pressure coefficient C_P from the previous n timesteps. We call node features y_i, while edge weights e_{ij}.

Graph connectivity is represented by the adjacency matrix $\mathbf{A}$, where each weight e_{ij} is the Euclidean distance between grid points i and j: $e_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|_2$. To normalize the weights to $(0, 1]$ and include self-loops ($e_{ii} = 1$), the adjacency matrix is augmented: $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$. Since edges are bidirectional ($e_{ij} = e_{ji}$), $\tilde{\mathbf{A}}$ is symmetric.

Due to the graph sparsity, the adjacency matrix is stored in a memory-efficient Coordinate List (COO) format, where the edge-index matrix contains pairs of node indices, and the edge-weight matrix stores the corresponding weights, with n_e being the number of edges.

2.1.2. Graph convolutional networks

GCNs are a particular type of ML algorithms that are based on graph theory. GCNs extract features from graphs by aggregating information from neighboring nodes using a graph convolutional operator. This operator, originally introduced by Duvenaud et al. in 2015 for molecular fingerprints [46], was later extended by Kipf et al. in 2016 [20] and is now implemented in `PyTorch-Geometric` library [47]. GCNs effectively generate node embeddings that capture structural information, making them ideal for tasks requiring an understanding of relationships between nodes.

The GCN operator follows the layer-wise propagation rule that is defined by the equation:

$$H^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{1}$$

Here, $H^{(l)}$ and $H^{(l+1)}$ are the node feature matrices at layers l and $l+1$, $\tilde{\mathbf{A}}$ is the adjacency matrix with self-loops, $\tilde{\mathbf{D}}$ is the degree matrix calculated on $\tilde{\mathbf{A}}$, $W^{(l)}$ is the matrix of trainable weights, and σ is the activation function. This rule propagates information from a node to its neighbors, allowing nodes to gather information from larger neighborhoods as layers are stacked. Equation (1) is a first-order approximation of trainable localized spectral filters g_θ on graphs [20].

A spectral convolution $g_\theta * x$ of an input graph $\tilde{x}$ with a filter g_θ in the Fourier domain is defined as:

$$g_\theta * x = \mathbf{U}g_\theta\mathbf{U}^T x \tag{2}$$

where $\mathbf{U}$ contains the eigenvectors of the graph Laplacian, $\mathbf{L}$. By approximating the spectral filter g_θ using Chebyshev polynomials [48], GCNs perform efficient localized filtering on graph data. This approximation simplifies the convolution process, making it feasible for large-scale graphs while preserving the ability to extract meaningful node features.

2.1.3. Temporal layers

In our framework, we explored various layers for temporal modeling: GRUs, LSTMs, attention mechanisms, and STGCN layers. Each method offers a distinct way to capture temporal dependencies, with varying level of complexity and performance suited to different contexts. In this section, we provide a brief overview of these methods, highlighting their key features and how they are integrated into our model. This comparison helps evaluate their effectiveness in handling temporal sequences.

Gated Recurrent Unit

The combination of GCNs with GRU [33] offers several key advantages when dealing with spatio-temporal data. GRUs are widely used and well-suited for modeling temporal dependencies, but they can not directly used with non-Cartesian domains like graphs, where spatial relationships are irregular. By incorporating graph convolution operators on GRUs, it is possible to improve the generalization capability of the model by replacing traditional convolution with a graph convolution, which can handle arbitrary graph structures and effectively learn from unstructured data.

Based on the approach in [49] where recurrent networks for fixed grid-structured sequence are introduced, Seo et al. [50] proposed a Graph Convolutional Recurrent Network (GCRN) architecture for building a generalized GRU that works with unstructured sequence. To generalize the model to graphs, the 2D convolution is replaced by the graph convolution operator, here $*_g$, introduced in (2). In particular, GRU cell in GCRN is defined by:

$$\begin{aligned}
z_t &= \sigma(W_{xz} *_g x_t + U_{hz} *_g h_{t-1} + b_z)\\
r_t &= \sigma(W_{xr} *_g x_t + U_{hr} *_g h_{t-1} + b_r)\\
\hat{h}_t &= \phi(W_{xh} *_g x_t + U_{hh} *_g (r_t \odot h_{t-1}) + b_h)\\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t
\end{aligned}$$

Here, $W_{xz} *_g x_t$ refers to the graph convolution operation of x_t with spectral filters which are functions of the graph Laplacian L parametrized by K Chebyshev coefficients. z_t is the update gate vector, r_t is the reset gate vector, $\hat{h}_t$ is the candidate activation vector, h_t is the hidden state at time step t, W and U are the trainable weight matrices for the input and hidden states, respectively, σ is the logistic sigmoid function, and ϕ is the hyperbolic tangent function (or other possible activation functions). The operator $\odot$ denotes the Hadamard product, while b represents the biases.

Long Short-Term Memory

LSTM networks [32] are particularly useful when the data involves long-term dependencies, as they include memory units that can store information across multiple timesteps. This makes them potentially suitable to model unsteady aerodynamic flows where past behavior influences future forecasts over long periods of time. While both LSTM and GRU address the vanishing gradient problem and are designed to capture temporal relationships, LSTM includes additional memory structures that enable it to retain information over longer time periods, by sacrificing computational power and increasing the number of parameters. The implementation of a convolutional graph based LSTM follows a similar approach presented before with GRU [50], by creating a model that replaces the 2D convolution with the graph convolution operator $*_g$. In particular:

$$\begin{aligned}
i_t &= \sigma(W_{xi} *_g x_t + W_{hi} *_g h_{t-1} + w_{ci} \odot c_{t-1} + b_i)\\
f_t &= \sigma(W_{xf} *_g x_t + W_{hf} *_g h_{t-1} + w_{cf} \odot c_{t-1} + b_f)\\
c_t &= f_t \odot c_{t-1} + i_t \odot \phi(W_{xc} *_g x_t + W_{hc} *_g h_{t-1} + b_c)\\
o_t &= \sigma(W_{xo} *_g x_t + W_{ho} *_g h_{t-1} + w_{co} \odot c_t + b_o)\\
h_t &= o_t \odot \phi(c_t)
\end{aligned}$$

Where i_t is the input gate, f_t the forget gate, c_t the cell state, o_t the output gate and h_t the hidden state, which is the output of the LSTM at time step t. As before, σ is the logistic sigmoid function, ϕ is the hyperbolic tangent function, W represents the trainable weight matrix, and the support K of the graph convolutional kernels is defined by the Chebyshev coefficients. This extension of the standard LSTM architecture enables the model to learn temporal dependencies while also taking into account the spatial structure of the data.

Attention Mechanisms

Incorporating attention mechanisms allows for dynamically assigning importance to different time steps in a temporal sequence, which is especially useful in graph-based models where both complex spatial and temporal dependencies must be captured. Following the work of Bai et al. [45], attention mechanisms can be employed to re-weight the influence of hidden states of a GCRN across time, enabling the model to focus on the most relevant time points for prediction, rather than treating each equally. The model was constructed by combining GCN and GRU to compute both the spatial and temporal domains of the graph, by using the graph convolution operator $*_g$ introduced before. In addition, the attention mechanism is used to compute a context vector that selectively weighs the hidden states of the GCRN.

First, for each time step, the hidden states of the GRU h_t are passed through an attention layer, where attention scores α_t are computed using a softmax function. These scores are then used to weigh the hidden states, resulting in the context vector C, which captures the global variation information.

In particular, given a series of hidden states calculated by the recurrent network for T time steps: $H = \{h_1, h_2, ..., h_T\}$, the attention weights $\alpha_t, 1 < t < T$ are computed using a softmax function on the scores derived from a multilayer perceptron (MLP):

$$e_t = w^{(2)}(w^{(1)}H + b^{(1)}) + b^{(2)}, \quad \alpha_t = \frac{\exp(e_t)}{\sum_{i=1}^{T}\exp(e_i)}$$

where $w^{(1)}$, $w^{(2)}$, $b^{(1)}$, and $b^{(2)}$ are trainable weights and biases in the MLP. The context vector C is then calculated by the weighted sum and used for implementing the attention mechanism on the GCRN hidden states:

$$C = \sum_{t=1}^{T}\alpha_t h_t$$

By combining GCNs for spatial feature extraction with GRUs and attention mechanisms for temporal modeling, the model can capture both short-term and long-term dependencies in the data.

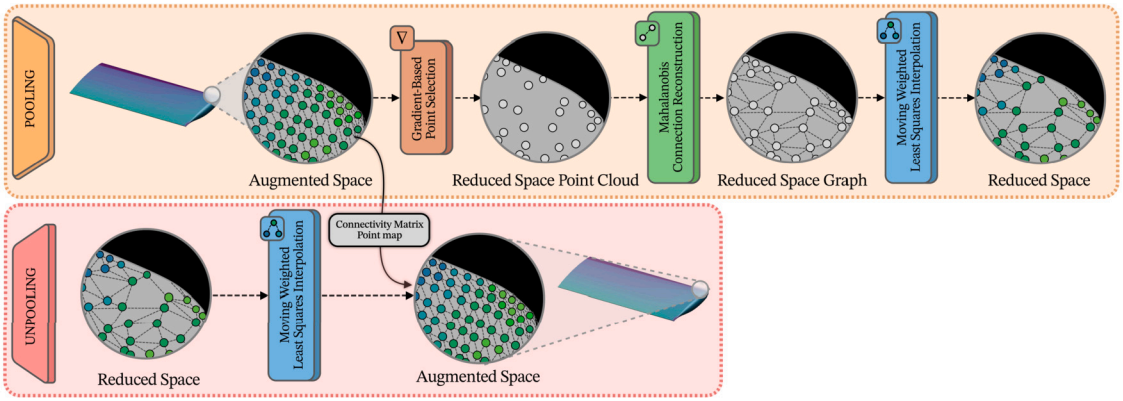


Fig. 2. Diagram of the pooling and unpooling modules used in the AE for dimensionality reduction and reconstruction.

Spatio-temporal Graph Convolution

STGCN is introduced by Yu et al. [51] as a method to capture temporal dependencies in spatio-temporal data by applying convolutions across the time dimension. Unlike RNNs, which process inputs sequentially, temporal convolutions handle entire sequences at once, allowing for parallelization and faster computation. The temporal convolutional block consists of 1-D causal convolutions followed by a Gated Linear Unit (GLU) to introduce non-linearity and control the flow of information.

For each node in the graph G, the temporal convolution layer explores K_t neighboring elements along the time axis. This approach does not require padding, and as a result, the length of the sequence decreases by $K_t - 1$ at each layer. Given an input sequence $Y \in \mathbb{R}^{M \times C_i}$ with M time steps and C_i channels, the convolution kernel $\Gamma \in \mathbb{R}^{K_t \times C_i \times 2C_o}$ maps the input to a single output element $[P\ Q] \in \mathbb{R}^{(M-K_t+1)\times(2C_o)}$. The GLU is then applied, splitting $[P\ Q]$ into two parts, and the output of the temporal convolution is given by:

$$\Gamma *_T Y = P \odot \sigma(Q) \in \mathbb{R}^{(M-K_t+1)\times C_o},$$

where P and Q are the inputs of the GLU, $\odot$ denotes the element-wise Hadamard product, and σ is the sigmoid function. The GLU selectively gates the information flow, determining which parts of the input sequence are relevant for capturing dynamic temporal dependencies. Stacking multiple layers of these temporal convolutions enables the model to capture both short- and long-term patterns effectively.

This approach can also be generalized to 3D tensors, where the same convolution kernel is applied to every node $Y_i \in \mathbb{R}^{M \times C_i}$ in the graph G, resulting in the operation $\Gamma *_T Y$ with $Y \in \mathbb{R}^{M \times n \times C_i}$.

2.1.4. Dimensionality reduction/expansion module

The dimensionality reduction and expansion process aims to simplify the computational load by eliminating redundant information and concentrating on key regions where nonlinear phenomena occur. This method is based on our previous work [31] and is visualized in Fig. 2, which illustrates both the pooling (reduction) and unpooling (expansion) phases. These phases form the core of the encoding and decoding operations in the AE architecture.

During the pooling phase, points are selected based on pressure gradients to create a reduced point cloud. This strategy ensures that key regions with high gradients are retained, while areas with lower gradients are simplified. The pressure gradient at each node i is calculated assuming that pressure p varies linearly in all spatial directions, as described by:

$$p_i - p_0 = \Delta p_i = \Delta x_i p_x + \Delta y_i p_y + \Delta z_i p_z \tag{3}$$

Here, p_0 represents the pressure at a reference node, while Δx_i, Δy_i, and Δz_i are the spatial differences between neighboring nodes i. Using a least-squares method, the gradient at each node is determined. Nodes are then selected for the reduced space based on a probability function driven by gradient magnitudes:

$$Pr(i) = 1 + \frac{1 - e^{-2i/n}}{1 - e^{-2}}(Pr_1 - Pr_n) + Pr_1 \quad \text{for} \quad i = 1, \ldots, n \tag{4}$$

where i refers to the node index ordered by gradient values, n is the total number of nodes, and Pr_1 and Pr_n are the probabilities assigned to the highest and lowest normalized gradient values, respectively set here to 0.2 and 1.0. Adjusting these two values allows the methodology to control the strength of the bias towards high-gradient regions. The mathematical form of Equation (4) may be adjusted to suit specific applications. The choice of this function in our case is based on a heuristic modeling approach that prioritizes higher gradient nodes while maintaining a smooth probabilistic transition across the ordered set. The exponential form provides a nonlinear but smooth decay in importance, offering more control over the sampling distribution than a linear or stepwise function. The function behavior can be adjusted by changing the range of the subtraction $(Pr_1 - Pr_n)$. Here, Pr_1 (lower probability value) corresponds to the nodes with the highest gradient magnitudes, which we aim to retain with higher priority, while Pr_n (higher probability value) is assigned to the nodes with the lowest gradients, making them more likely to be discarded. In practice, adjusting these two values allows the methodology to control the emphasis on sharp or smooth gradient features.

To reconnect the reduced point cloud, Mahalanobis distance (MD) is used [52], which accounts for the spread and covariance of the point distribution. MD between two points x and y is given by:

$$D_M(x, y) = \sqrt{(x - y)^T S^{-1} (x - y)} \tag{5}$$

where S is the covariance matrix of the original point distribution. MD is computed in the reduced point distribution, for each point to obtain a new set of nearest neighbors, from a reduced selection [31]. With MD, each point is connected according to the distribution of points in the finer mesh, using the covariance matrix computed in the original space. This method helps maintain accurate connectivity in the reduced graph, avoiding false connections caused by proximity errors when using Euclidean distance [31]. After MD is applied, the reduced point cloud becomes a new coarser graph.

Once the reduced graph is constructed, node values are interpolated using the Moving Weighted Least Squares (MWLS) method [53,54]. This methodology can be applied during both the encoding and decoding phases by inverting the interpolation matrix. Here we use the nomenclature *source* and *destination* grids, which represents the finer and coarser grids, respectively, during pooling, and vice versa during unpooling. The interpolation matrix $I_{S_s \to S_d}$ is calculated to map any scalar feature $\mathbf{y}$ values from the source grid S_s to the destination grid S_d. The final interpolated value at each node $\mathbf{x}_j$ for the specific feature $\mathbf{y}$ of the source grid is then computed as:

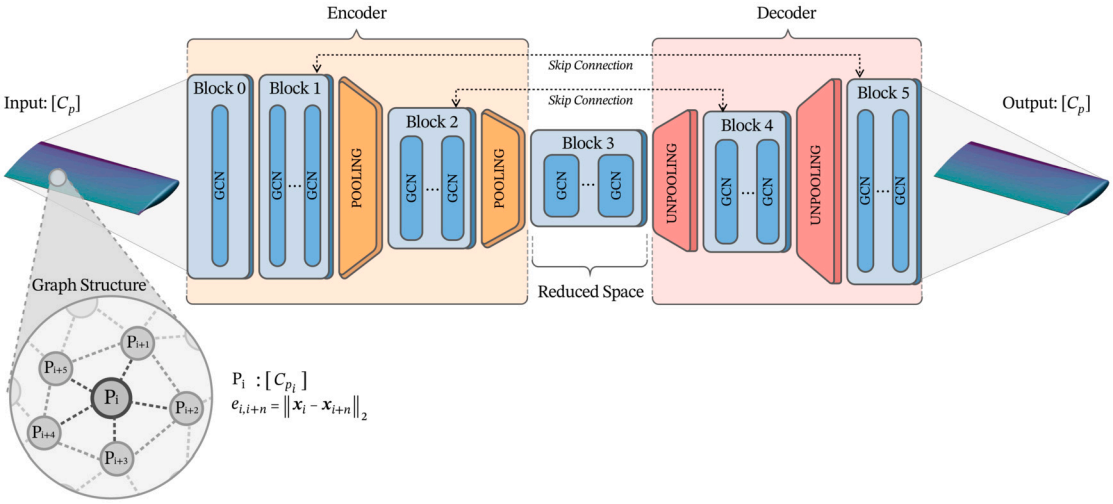


Fig. 3. Schematic of the pre-trained AE architecture for compressing and reconstructing the C_P data within the AeroNet framework.

Table 1
Parameters and types of training, test and validation signals. DS: damped Schroeder-phased harmonic, US: undamped Schroeder-phased harmonic, SH: single harmonic.

Signals	κ_θ [-]	a_θ [deg]	κ_ξ [-]	a_ξ [m]	Type
Training 1	0.114	0.80	0.152	-0.098	DS, $\theta > 0$, $\xi < 0$
Training 2	0.114	-0.80	0.152	0.098	DS, $\theta < 0$, $\xi > 0$
Training 3	0.148	1.00	0.181	-0.123	DS, $\theta > 0$, $\xi < 0$
Training 4	0.148	-1.00	0.181	0.123	DS, $\theta < 0$, $\xi > 0$
Test 1	0.091	0.70	0.123	0.074	DS, $\theta > 0$, $\xi > 0$
Test 2	0.104	0.90	0.089	0.061	DS, $\theta > 0$, $\xi < 0$
Test 3	0.104	-0.90	0.089	-0.061	DS, $\theta < 0$, $\xi < 0$
Test 4	0.092	0.75	0.081	-0.059	US, $\theta > 0$, $\xi < 0$
Test 5	0.147	-1.00	0.000	0.000	DS, $\theta < 0$
Test 6	0.000	0.00	0.072	0.049	DS, $\xi > 0$
Validation 1	0.147	-1.00	0.072	0.049	DS, $\theta < 0$, $\xi > 0$
Validation 2	0.106	3.00	0.089	-0.246	SH, $\theta > 0$, $\xi < 0$

$$u(\mathbf{x}_j) = \mathbf{\Phi}(\mathbf{x}_j)\,\mathbf{y}_{S_s}, \qquad \mathbf{y}_{S_s} = [\,y_1, \ldots, y_{n_s}\,]^T, \tag{6}$$

where:

$$\mathbf{\Phi}(\mathbf{x}_j) = \mathbf{m}^T(\mathbf{x}_j)(\mathbf{M}^T\mathbf{W}\mathbf{M})^{-1}\mathbf{M}^T\mathbf{W} \tag{7}$$

In this equation, $\mathbf{M}$ is the design matrix constructed from the source nodes, $\mathbf{m}(\mathbf{x})$ is a second-order polynomial basis, and $\mathbf{W}$ is a diagonal matrix of Gaussian weights $w(\mathbf{x}_i) = e^{-\|\mathbf{x_j}-\mathbf{x}_i\|_2}$. These weights control how strongly each source point influences the weighted least-squares fit, yielding a local approximation that is most accurate in the immediate vicinity of the interpolation point. The interpolation matrix $I_{S_s \to S_d}$ is used during pooling; since it is rectangular and therefore not invertible, a separate matrix $I_{S_d \to S_s}$ is calculated for the unpooling (decoder) step, by using the same methodology, swapping S_s and S_d.

For a detailed explanation of the entire encoding and decoding process, refer to the work of Immordino et al. [31].

2.1.5. Pre-trained autoencoder

Our proposed framework leverages an AE architecture, pre-trained for subsequent integration into the complete model. The pre-training phase involved using C_P data as both input and output to the AE, ensuring the model accurately captures the essential features of the pressure distribution over the wing surface. The training dataset comprised the four signals detailed in Table 1.

To enhance the robustness of the AE, a data augmentation technique was employed. Specifically, the dataset was augmented by 30% through the addition of Gaussian noise with 10% standard deviation of the input data. This augmentation strategy was designed to improve the model ability to generalize and handle variability in the pressure distribution data. Skip connections were integrated before each encoding module to facilitate the direct flow of information across the network. These connections allow the model to bypass certain layers, enabling the retention of critical features and mitigating the risk of information loss during the encoding and decoding processes. The network architecture has been optimized using a Bayesian optimization strategy, following the same approach of our previous work [31]. A schematic of the pre-trained AE architecture is shown in Fig. 3.

2.2. Dynamic mode decomposition with control

Dynamic Mode Decomposition with Control (DMDc) provides a systematic approach for deriving ROMs from high-dimensional data, explicitly incorporating external inputs. Since this is a well-known methodology in this field, we developed a baseline model to serve as a reference for our GCN-based framework.

Originally introduced by Proctor et al. [4], DMDc extends Dynamic Mode Decomposition to include control effects, yielding the following linear system:

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t, \tag{8}$$

where $\mathbf{x}_t \in \mathbb{R}^m$ is the state vector, $\mathbf{u}_t \in \mathbb{R}^p$ is the control input vector (modal amplitudes and derivatives), and $\mathbf{A}$, $\mathbf{B}$ are the dynamic and input matrices, respectively. In our case, state vector contains pressure coefficient (C_{P_t}), while the control vector contains pitch ($\theta_t, \dot{\theta}_t, \ddot{\theta}_t$) and plunge ($\dot{\xi}_t, \ddot{\xi}_t$).

Following Fonzi et al. [7], we define the state vector as:

$$\mathbf{x}_t = \begin{bmatrix} C_{p,1} & C_{p,2} & \ldots & C_{p,m} \end{bmatrix}^T, \tag{9}$$

where subscript m indicates the number of surface grid points. The input vector is:

$$\mathbf{u}_t = \begin{bmatrix} \mathbf{q}_t^T & \dot{\mathbf{q}}_t^T & \ddot{\mathbf{q}}_t^T \end{bmatrix}^T, \tag{10}$$

where $\mathbf{q}_t$ denotes structural modal amplitudes at time t.

To identify the system matrices, we collect snapshot matrices from the dataset:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1 & \ldots & \mathbf{x}_{n-1} \end{bmatrix}, \quad \mathbf{X}' = \begin{bmatrix} \mathbf{x}_2 & \ldots & \mathbf{x}_n \end{bmatrix}, \tag{11}$$

$$\mathbf{U} = \begin{bmatrix} \mathbf{u}_1 & \ldots & \mathbf{u}_{n-1} \end{bmatrix}. \tag{12}$$

Where subscript n denotes the number of timesteps for a single simulation. We define the augmented input matrix:

$$\mathbf{\Psi} = \begin{bmatrix} \mathbf{X} \\ \mathbf{U} \end{bmatrix}, \tag{13}$$

and compute the DMDc operators using the pseudoinverse:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \end{bmatrix} = \mathbf{X}'\mathbf{\Psi}^{\dagger}. \tag{14}$$

To reduce the model order, we apply truncated SVD to $\mathbf{X} \approx \mathbf{U}_r\mathbf{\Sigma}_r\mathbf{V}_r^*$, and project $\mathbf{A}$ and $\mathbf{B}$ onto the dominant modes:

$$\tilde{\mathbf{A}} = \mathbf{U}_r^T\mathbf{A}\mathbf{U}_r, \quad \tilde{\mathbf{B}} = \mathbf{U}_r^T\mathbf{B}. \tag{15}$$

Truncation is based on the Frobenius norm of $\mathbf{\Sigma}$, with the number of retained modes selected to minimize below a threshold the root mean square error between the original and the reconstructed state matrix. The resulting system is:

$$\tilde{\mathbf{x}}_{t+1} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}_t + \tilde{\mathbf{B}}\mathbf{u}_t, \quad \mathbf{x}_t \approx \mathbf{U}_r\tilde{\mathbf{x}}_t, \tag{16}$$

enabling the model to predict surface pressure evolution.

A stabilization procedure is applied to improve the conditioning of the reduced system, especially when unstable eigenvalues appear due to truncation artifact. This procedure modifies the eigenvalues of matrix $\tilde{\mathbf{A}}$, making them all lie inside the unit circle in the complex plane. When an unstable eigenvalue is obtained, a new eigenvalue is calculated using a flip method, which modifies the eigenvalue outside the unit circle and reflect it back inside while preserving phase and mode orientation (refer to [7]).

3. Test case

The selected case study for evaluating our framework is the Benchmark Super Critical Wing (BSCW), a transonic rigid semi-span wing featuring a rectangular planform and a supercritical airfoil profile, as detailed in the AIAA Aeroelastic Prediction Workshop [55]. The freestream conditions for this case are defined by a Mach number of 0.74, a Reynolds number of 4.49×10^6, and an initial angle of attack (AoA_∞) of 0 degree. The wing features a reference chord length of 0.4064 m and a surface area of 0.3303 m^2, with pitching motion occurring around 30% of the chord. It is mounted on a flexible support system that allows two degrees of freedom: pitch θ and plunge ξ. It is designed for flutter analysis, presenting challenges due to shock wave motion, shock-induced boundary-layer separation, and the interaction between the shock wave and the detached boundary layer. These nonlinear phenomena pose significant challenges for the framework predictions.

An unstructured mesh with 8.4×10^6 elements and 86,840 surface elements was created. A $y^+ = 1$ value was used, based on a preliminary mesh convergence study that confirmed adequate resolution of the boundary layer and shock wave. The computational domain extends 100 chord lengths from the solid wall to the farfield. Fig. 4 provides an impression of the mesh configuration.

The dataset used to train the model was generated with CFD unsteady responses using the Unsteady Reynolds-averaged Navier–Stokes (URANS) formulation with SU2 v7.5.1 [56]. The simulations employed the one-equation Spalart-Allmaras turbulence model for RANS closure. To accelerate convergence, a $1v$ multigrid scheme was used. The JST central scheme with artificial dissipation handled convective flow discretization, and the gradients of flow variables were calculated using the Green-Gauss method. The biconjugate gradient stabilization linear solver, along with the ILU preconditioner, was utilized. All URANS simulations started from a steady-state solution, with a timestep of 2×10^{-4} seconds and a total simulation time of 2 seconds. These values were chosen to ensure a high temporal resolution, capturing rapid aerodynamic variations while keeping the computational cost manageable over the full simulation period. However, due to computational constraints, the timestep was reduced to 2×10^{-3} seconds through downsampling for model training, resulting in 660 timesteps per signal. This ensures a balance between computational feasibility and quality of aerodynamic data.

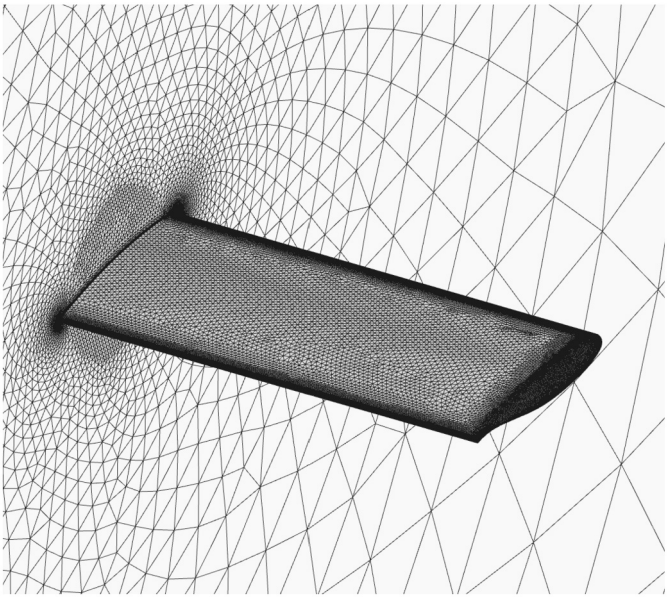


Fig. 4. Impression of the BSCW CFD grid.

Table 2
Number of samples for the training, test and validation datasets.

Dataset	Number of Samples
Training	2,640 (4 × 660)
Test	3,960 (6 × 660)
Validation	1,320 (2 × 660)

We ran a total of 12 time-varying simulations, divided into training, test, and validation sets as described in Table 1. The training set comprises damped Schroeder-phased harmonic (DS) simulations (see Appendix A for details on the DS signal formulation) with varied combinations of the parameters: reduced frequency for pitch (κ_θ), maximum pitch amplitude (a_θ), reduced frequency for plunge (κ_ξ), and maximum plunge amplitude (a_ξ), ensuring that the model learns from diverse conditions where these variables exhibit both positive and negative values (Fig. 5). This variation helps the model understand the complex interactions between the parameters. The test set extends this diversity by including additional combinations and signal types, specifically undamped Schroeder-phased harmonic (US) and cases focusing solely on θ or ξ variations. The signs associated with pitch and plunge indicate the initial directions of oscillation: a positive pitch corresponds to a nose-up rotation, while a positive plunge indicates downward motion. This approach allows us to accurately assess the model accuracy and sensitivity in predicting the effects of individual parameters, ensuring that it performs well across different conditions. The use of Schroeder signals is motivated by their ability to effectively cover a broad frequency spectrum while minimizing peak amplitudes, which enhances the model robustness and ability to generalize. The validation set is designed to evaluate the model generalizability to new and unseen data. It includes a scenario similar to those in the training set but with distinct parameter values, as well as a unique single harmonic (SH) signal type, which the model did not encounter during training. This ensures that the model can handle both familiar and novel situations effectively.

Table 2 summarizes the number of samples in the training, test, and validation datasets, providing a clear overview of the dataset structure used in this study. Each sample consists of three consecutive timesteps as input and one timestep as output.

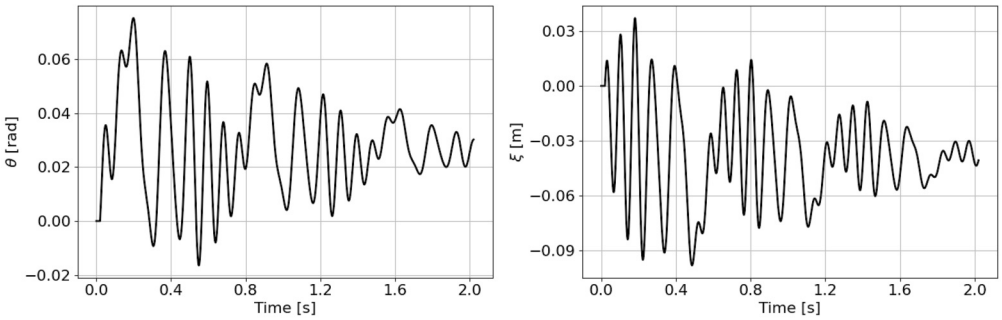


Fig. 5. Training signal 1: DS with $\kappa_\theta = 0.114$, $a_\theta = 0.80$ [deg], $\kappa_\xi = 0.152$, and $a_\xi = -0.098$ [m].

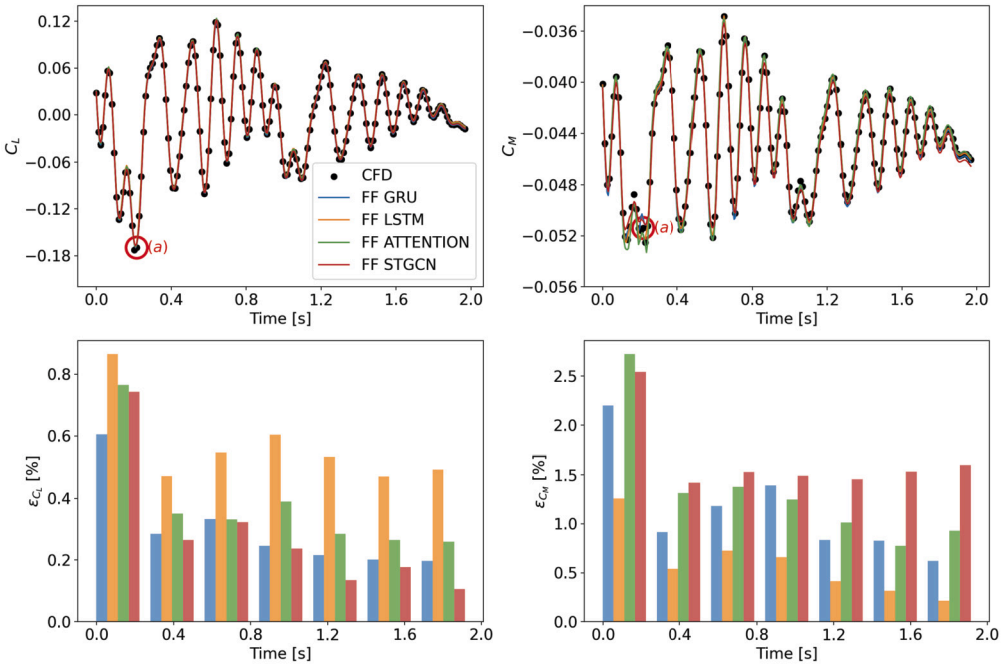


Fig. 6. Validation signal 1 - DS type: Effect of temporal layer selection on C_L and C_M predictions in the feedforward model. The red circle marks the point of maximum error, used for plotting the C_P distribution. FF: FeedForward. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

4. Results

In this section, we present the results of the reconstructed validation signals for two different types of architectures: feedforward model and ARMA model. By comparing the performance of these two architectures, we aim to illustrate the influence of incorporating predicted C_P into the model input on the overall prediction accuracy and robustness. The impact of the temporal layer on the accuracy of each architecture is also studied.

4.1. Feedforward model

The feedforward model utilizes spatial and temporal data from previous timesteps without relying on its own past predictions, thus preventing the accumulation of errors over time. The model performance across different temporal layers is evaluated using both DS and SH validation signals.

For the DS signal (Figs. 6 and 7), the predictions of the aerodynamic coefficients C_L and C_M align well with the CFD reference data in all temporal layers, as shown in Fig. 6. The STGCN layer demonstrates the lowest average error for C_L, while LSTM performs slightly better for C_M. A region of maximum error, indicated by the red circle, is consistent across all models and represents the point where the predictions for C_L and C_M show the greatest deviation from the reference data. This error appears to be related to the models difficulty in capturing sharp transitions or non-linearities in the aerodynamic flow, particularly near shock waves. Fig. 7 shows the C_P distribution at this maximum error point and highlights the models overall ability to predict the pressure distribution across the wing. While most temporal layers perform reasonably well, some discrepancies near the leading edge and areas affected by shock waves and flow separation are noticeable. These areas, characterized by strong flow gradients and non-linear aerodynamic behavior, are challenging for all models, although STGCN and LSTM exhibit slightly better accuracy compared to GRU and Attention mechanisms.

For the SH signal (Figs. 8 and 9), which involves higher-frequency oscillations, the models encounter increased errors, particularly in predicting peak values for C_L and C_M. LSTM and STGCN continue to provide the most accurate results, although both exhibit some phase and amplitude errors due to the rapid oscillations. The red-circled point, indicating the region of maximum error for C_L and C_M, again highlights the challenges of accurately capturing high-frequency aerodynamic loads. Fig. 9 provides a detailed view of the C_P distribution at this maximum error point, where the models struggle more to match the rapid fluctuations in C_P. Again, regions near the shock wave, which undergo significant

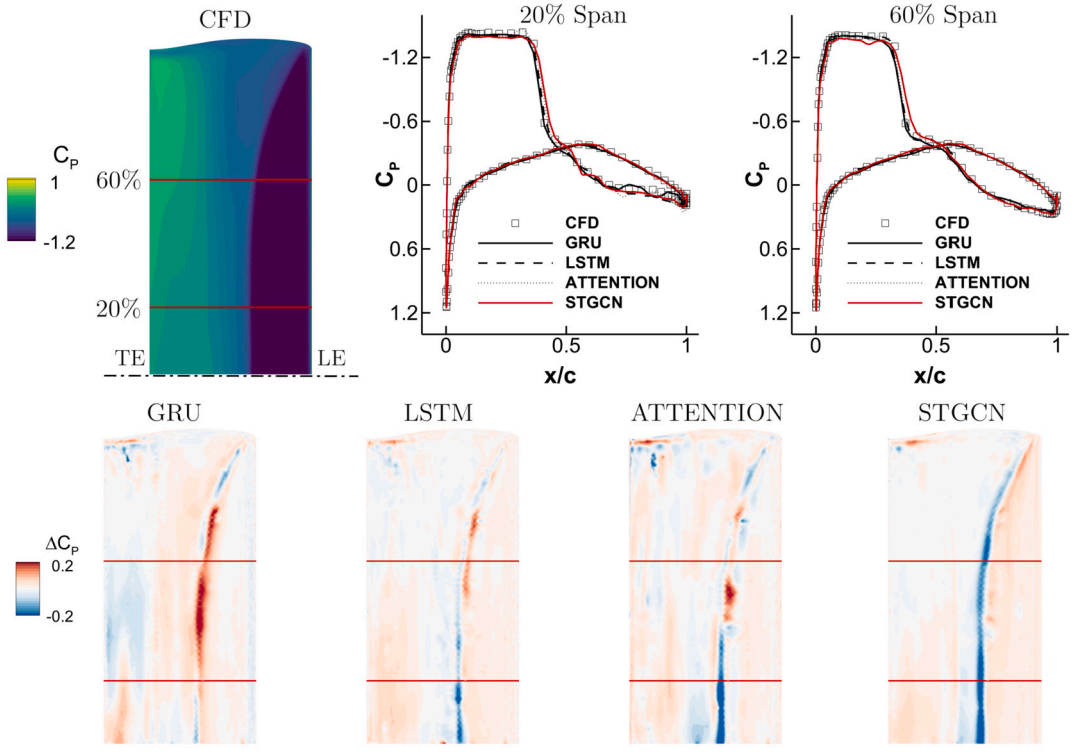


Fig. 7. Validation signal 1 - DS type: Effect of temporal layer selection on C_P prediction at the maximum error point in the feedforward model. The lower surface of the wing is shown. LE: Leading Edge. TE: Trailing Edge. Dash-dot line indicates the symmetry plane.

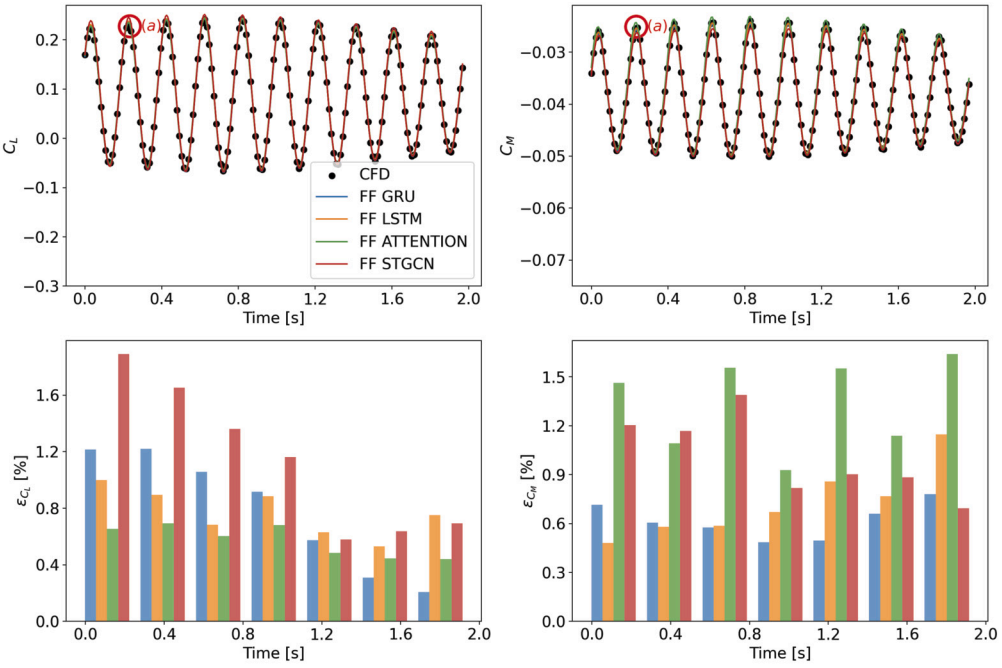


Fig. 8. Validation signal 2 - SH type: Effect of temporal layer selection on C_L and C_M predictions in the feedforward model. The red circle marks the point of maximum error, used for plotting the C_P distribution. FF: FeedForward.

temporal variation, show greater discrepancies. While the general C_P distribution is captured, the accuracy decreases in areas where shock-induced flow separation occurs, with LSTM and STGCN again showing marginally better performance in these challenging regions.

To quantitatively evaluate the models, we calculated three error metrics—Mean Absolute Percentage Error (MAPE), coefficient of determination (R2), and Root Mean Square Error (RMSE)—for C_P predictions across both validation signals, as shown in Table 3. These metrics are defined as:

$$\texttt{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|, \quad \texttt{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2},$$

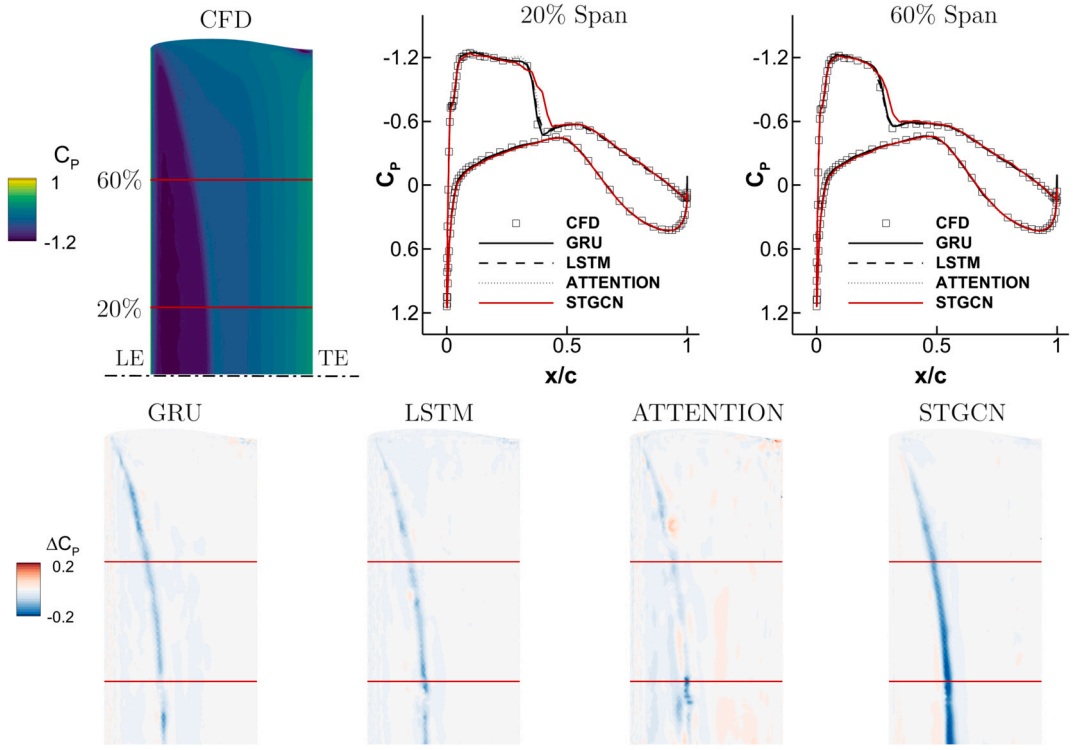


Fig. 9. Validation signal 2 - SH type: Effect of temporal layer selection on C_P prediction at the maximum error point in the feedforward model. The upper surface of the wing is shown. LE: Leading Edge. TE: Trailing Edge. Dash-dot line indicates the symmetry plane.

Table 3
Comparison of MAPE, R2, and RMSE for C_P predictions in the feedforward model with different temporal layers for DS and SH validation signals.

Temporal Layer	MAPE		R2		RMSE	
	DS	SH	DS	SH	DS	SH
GRU	1.2382	1.7851	0.9851	0.9830	0.0217	0.0229
LSTM	**0.7471**	**0.9695**	**0.9937**	**0.9909**	0.0194	0.0215
Attention	1.0470	1.5452	0.9887	0.9815	0.0238	0.0291
STGCN	0.8524	0.9975	0.9918	0.9897	**0.0163**	**0.0181**

$$\mathrm{R2} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

where y_i is the Aeronet prediction averaged over the surface mesh, weighted by the cell area, $\hat{y}_i$ is the corresponding CFD reference value computed under the same flow conditions and with the same weighting, n is the number of timesteps in the signal, and $\bar{y} = \frac{1}{n}\sum_{i=1}^{n}\hat{y}_i$ is the mean of the ground-truth CFD series.

MAPE reflects the average percentage error across predictions, showing that the LSTM model consistently achieves the lowest values, while the STGCN model follows closely. GRU and Attention mechanisms, on the other hand, display significantly higher MAPE, particularly for the high-frequency oscillations of the SH signal, indicating their difficulty in capturing rapid temporal variations and non-linearities. In terms of R2, which measures how well the model explains variance in the data, LSTM again performs the best, capturing the highest degree of variability, followed closely by STGCN, which also demonstrates strong performance in the DS signal but slightly lower accuracy for the SH signal. GRU and Attention layers show lower R2, further confirming their difficulty to fully capture the complexities in regions involving shock waves and high dynamic variability. Regarding RMSE, which emphasizes larger errors due to its quadratic nature, the STGCN model performs better with the lowest values, indicating a good robustness against significant deviations. LSTM shows similar performance but struggles slightly more with large deviations in dynamic conditions like the SH signal. Again, GRU and Attention mechanisms exhibit the highest RMSE. These results suggest that all models are well-suited for capturing both spatial and temporal dependencies with good overall performance, but GRU and Attention layers struggle more to capture the rapid temporal variations and non-linearities in the C_P distribution.

In conclusion, while all temporal layers provide reasonable predictions for unsteady aerodynamic phenomena, the LSTM model delivers the most robust performance across both validation signals. STGCN also performs well, particularly for C_L predictions, while GRU and Attention mechanisms exhibit larger errors, especially in more complex dynamics. The evaluation of C_P distribution highlights the importance of accurately capturing non-linear flow features, with regions near shock waves and flow separations proving to be the most challenging for all models.

4.2. ARMA model

The ARMA model, which integrates past predictions into its input, was evaluated using various temporal layers to assess its performance in predicting C_L, C_M, and C_P across DS and SH validation signals. As shown in Fig. 10, one of the biggest limitations of the ARMA model consists of accumulation of errors over time, particularly when using the GRU and Attention mechanisms. Again, LSTM and STGCN demonstrate better predictive performance, although both exhibit some deviations in

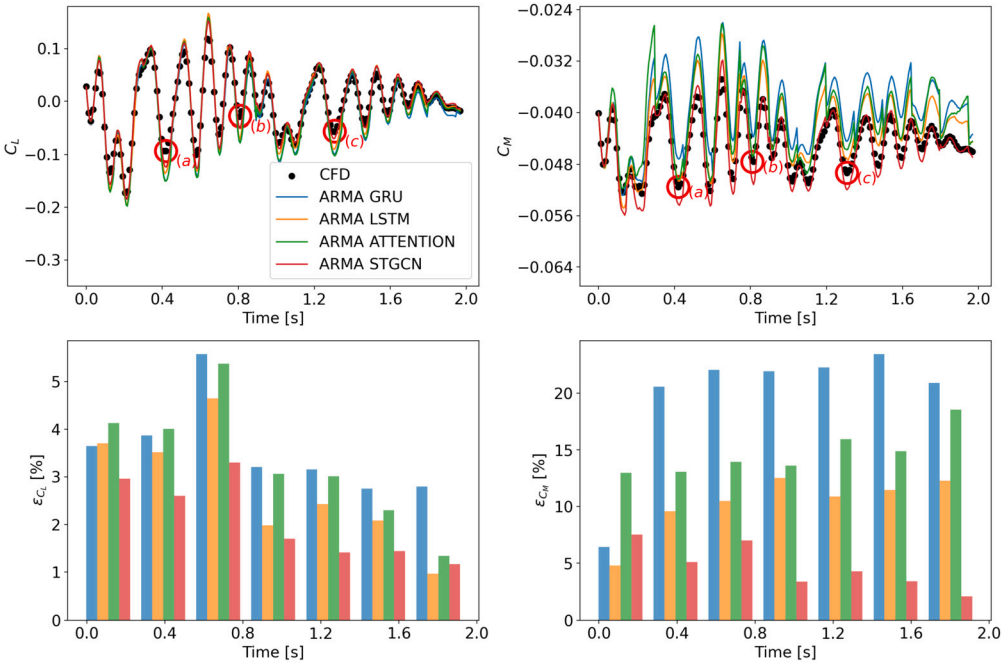


Fig. 10. Validation signal 1 - DS type: Impact of temporal layer selection on C_L and C_M predictions in the ARMA model. Red circles denote the points used for plotting the C_P distribution.

Table 4
Comparison of MAPE, R2, and RMSE for C_P predictions in the ARMA model with different temporal layers for DS and SH validation signals.

Temporal Layer	MAPE		R2		RMSE	
	DS	SH	DS	SH	DS	SH
GRU	8.5577	6.7837	0.7560	0.7993	0.1389	0.1439
LSTM	7.7511	6.4299	0.8426	0.8506	0.1002	0.0864
Attention	7.7985	5.8077	0.8167	0.7796	0.1233	0.1048
STGCN	**6.9381**	**5.7971**	**0.8571**	**0.8648**	**0.0938**	**0.0844**

complex flow regions. The red-circled points were selected to visualize the evolution of the error over time. These points were spaced at regular intervals along the signal to provide insight into how prediction accuracy changes throughout the sequence. This allows us to observe how the model handles different stages of prediction and highlights its difficulty in accurately capturing sharp transitions and non-linearities in the aerodynamic flow, particularly around shocks. This issue is further emphasized in Fig. 11, where the C_P at these selected points shows that the ARMA model has more difficulty maintaining accuracy near the leading edge and in shock-affected regions. Despite these challenges, LSTM and STGCN continue to provide the most reliable predictions despite the inherent error accumulation.

For the SH signal (Figs. 12 and 13), which involves higher frequency oscillations, the ARMA model exhibits increased errors, particularly for the GRU and Attention layers. The rapid oscillations introduce phase and amplitude discrepancies, with LSTM and STGCN handling these variations better, though both models still show increased errors compared to the DS signal. The C_P distribution in Fig. 13 reveals that the ARMA model is more susceptible to error propagation in highly dynamic regions near shock waves, where rapid changes in flow introduce greater inaccuracies. Although LSTM and STGCN mitigate these issues to some extent, they still experience some degradation in predictive accuracy due to the model autoregressive nature.

The superior performance of LSTM and STGCN can be attributed to their inherent design, which allows them to handle temporal dependencies more effectively. The LSTM architecture, with its complex gating mechanisms, enables the model to retain and manage both long- and short-term dependencies, preventing information loss over multiple timesteps and helping to mitigate the error accumulation issue inherent in the ARMA model. The STGCN layer combines both spatial and temporal convolutions, making it well-suited to handle non-uniform grid structures and effectively capture spatial correlations (such as pressure gradients across the wing) as well as temporal dependencies during rapid changes in aerodynamic conditions, as seen in the SH signal.

In contrast, the GRU simpler structure limits its capacity to capture long-term dependencies effectively. The Attention mechanism, while effective for capturing important temporal relationships in longer sequences, is not as well suited to the short input sequences used in this model, where the benefits of dynamically weighting timesteps are reduced, limiting the Attention layer effectiveness.

Table 4 quantifies the performance of the different temporal layers in terms of MAPE, R2, and RMSE for C_P predictions in the ARMA model. The STGCN model consistently achieves the lowest MAPE and RMSE values across both the DS and SH signals, with LSTM performing nearly as well. This suggests that both models are adept at handling temporal dependencies even in autoregressive contexts, although the STGCN slightly outperforms LSTM, particularly in the dynamic SH signal. GRU and Attention layers, on the other hand, show higher MAPE and RMSE, along with lower R2, indicating that they struggle to mitigate the error accumulation inherent in the ARMA model, particularly in high-frequency oscillations. The lower R2 values for GRU and Attention mechanisms highlight their reduced ability to capture the full variability in the data with rapid aerodynamic changes. Overall, the results suggest that while the ARMA model can capture general trends, its susceptibility to error propagation makes it crucial to use robust temporal layers, such as LSTM

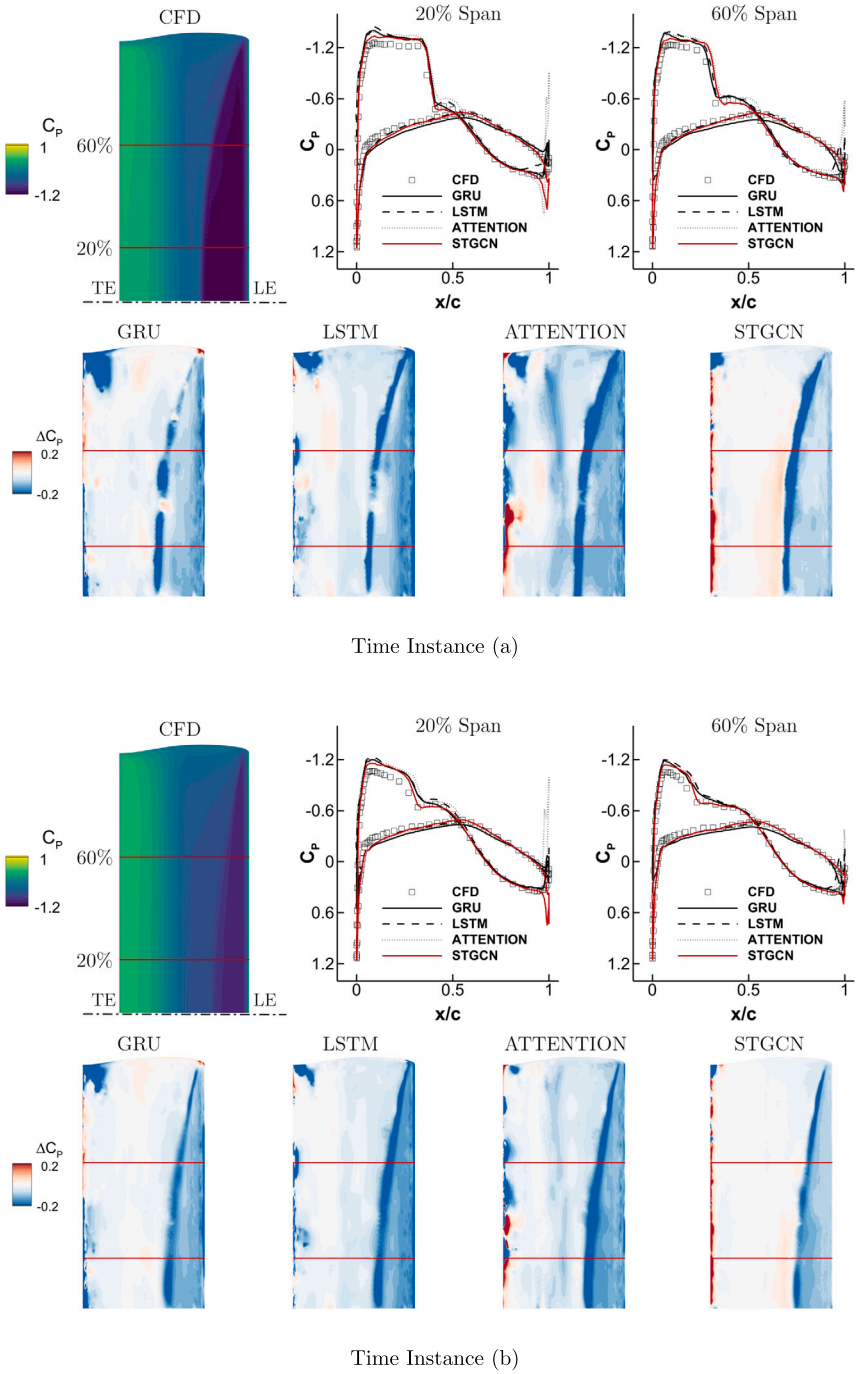
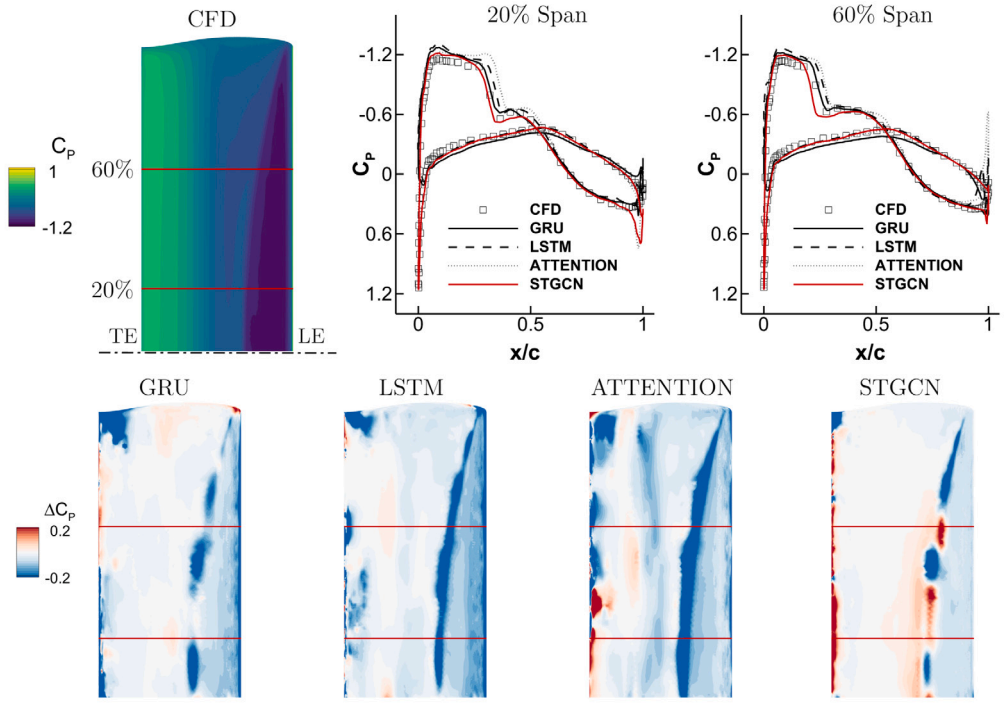


Fig. 11. Validation signal 1 - DS type: Impact of temporal layer selection on C_P prediction at the maximum error point in the ARMA model. The lower surface of the wing is shown. LE: Leading Edge. TE: Trailing Edge. Dash-dot line indicates the symmetry plane.

and STGCN, to minimize predictive inaccuracies in highly dynamic and non-linear aerodynamic conditions.

4.3. Comparison between feedforward and ARMA

The comparison between ARMA and feedforward models, both using the STGCN temporal layer, reveals notable differences in performance, especially regarding error accumulation and prediction stability. The STGCN temporal layer was selected for this comparison because it consistently yielded the most accurate results across both validation signals, as demonstrated in previous sections. In this case, ARMA model uses ground-truth C_P values for the first half of the signal, after which it switches to using its own predictions for subsequent timesteps. As shown in Fig. 14, the feedforward model provides more accurate predictions for C_L and C_M on the DS signal, avoiding the accumulation of errors observed in the ARMA model. The red circled points were chosen to be in the middle of the first and second halves of the signal, providing information on the behavior of the model during the transition from using ground truth to self-predicted values. This choice allows for a clearer comparison of model performance during both phases, highlighting the ARMA model difficulty in limiting error accumulation, while the feedforward model maintains closer alignment with the reference data. This trend is further confirmed in Fig. 15, where the evolution of the MAPE in C_P reveals that the feedforward model consistently maintains lower er-



Time Instance (c)

Fig. 11. (*continued*)

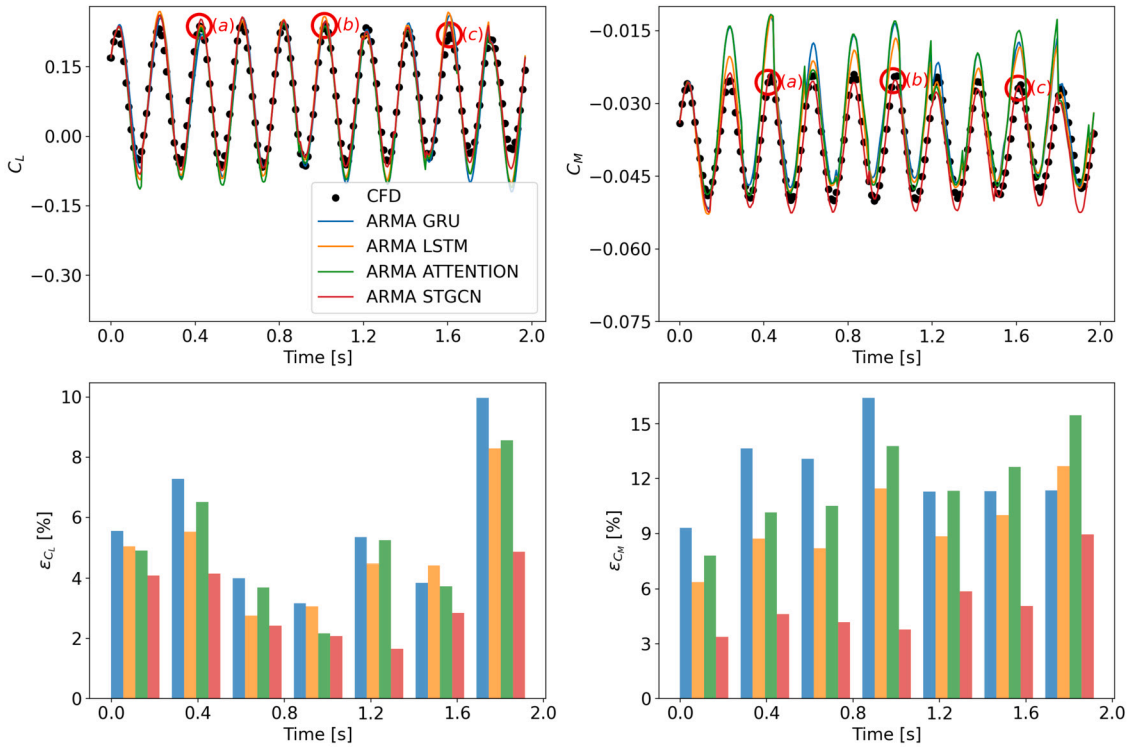


Fig. 12. Validation signal 2 - SH type: Impact of temporal layer selection on C_L and C_M predictions in the ARMA model. Red circles denote the points used for plotting the C_P distribution.

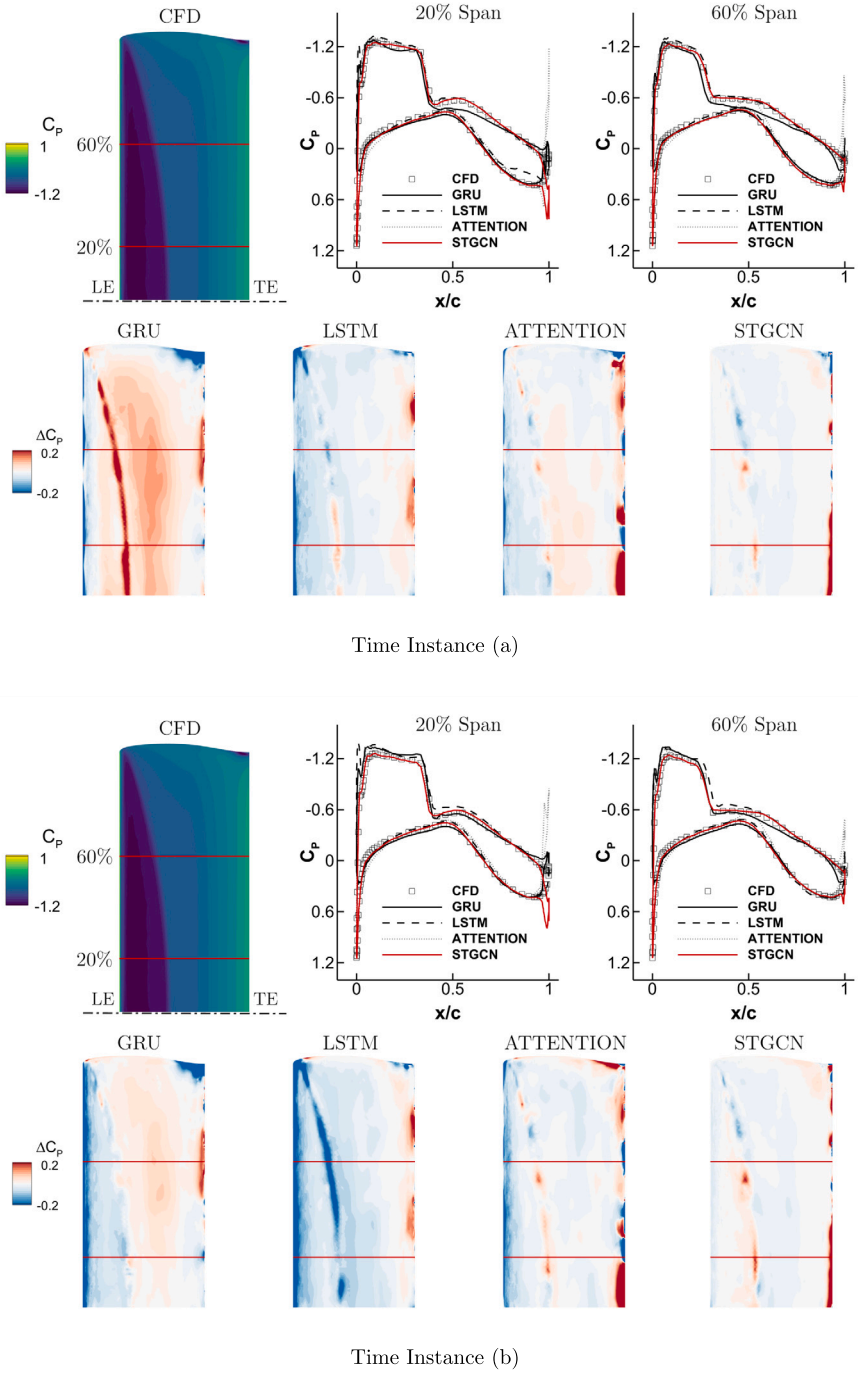


Fig. 13. Validation signal 2 - SH type: Impact of temporal layer selection on C_P prediction at the maximum error point in the ARMA model. The upper surface of the wing is shown. LE: Leading Edge. TE: Trailing Edge. Dash-dot line indicates the symmetry plane.

ror levels over time compared to the ARMA model, which, as expected, shows a clear pattern of accumulation of errors.

Interestingly, the ARMA model performs relatively well during the first half of the signals, when ground-truth C_P values are used as input. In this phase, the ARMA model can even outperform the feedforward model, as seen in the initial part of Figs. 14 and 15. However, once the model begins using its own predicted C_P values for subsequent timesteps, error accumulation begins, resulting in larger deviations from the reference data. This behavior is clearly seen in Fig. 16, where the ARMA model shows growing discrepancies in C_P predictions as the autoregressive process progresses. In contrast, the feedforward model avoids this problem by not relying on its past predictions, allowing it to maintain better accuracy in regions near the leading and trailing edges, where flow complexity is higher.

These trends are even more evident with the SH signal, which features rapid oscillations. Fig. 17 shows that the feedforward model handles these high-frequency aerodynamic variations more effectively, with significantly fewer phase and amplitude errors than the ARMA model. The ARMA model, due to its time-marching scheme, exhibits a greater sensitivity to reduced frequency. At higher frequencies, errors accumulate faster as small inaccuracies from previous steps compound. Once the ARMA model switches from using ground-truth inputs to its own predictions, it fails to keep pace with the rapid changes in aerodynamic forces, resulting in larger phase lags and more significant deviations from the

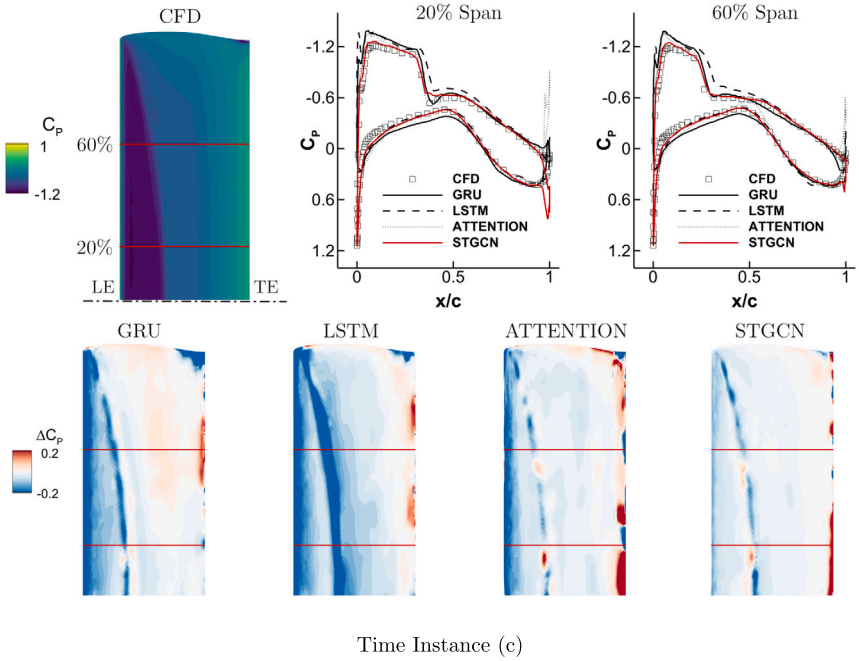


Fig. 13. (*continued*)

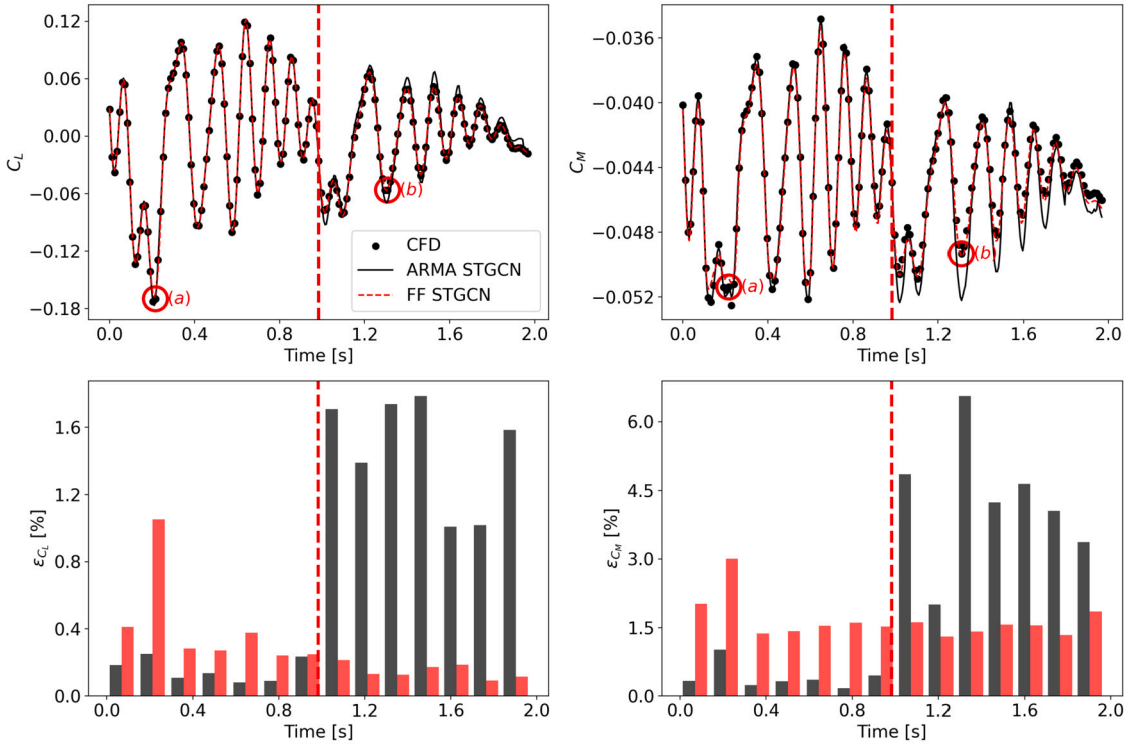


Fig. 14. Validation signal 1 - DS type: ARMA vs. Feedforward model using STGCN temporal layer on C_L and C_M predictions. Red circles denote the points used for plotting the C_P distribution. Vertical dashed line marks the time instance where the ARMA model transitions to autoregressive C_P predictions.

reference data. In contrast, as the feedforward model relies only on the wing spatial coordinates and prescribed motions at previous timesteps (without feeding back its own predictions), it results in a more stable error profile and in a more reliable and accurate predictions of unsteady phenomena.

As shown in Fig. 18, the feedforward model consistently outperforms the ARMA model in terms of MAPE on C_P for the SH signal. The ARMA model error accumulation is particularly pronounced in more dynamic conditions, such as rapid oscillations, which results in significantly higher MAPE. Fig. 19 further supports this observation, showing that the feedforward model is better at predicting the C_P distribution, where the ARMA model struggles due to the accumulation of prediction errors. However, it is important to note that when the ARMA model is feeded with ground-truth C_P values, it can produce very accurate predictions, outperforming the feedforward model. This highlights the ARMA potential in study cases where error on pressure values does not

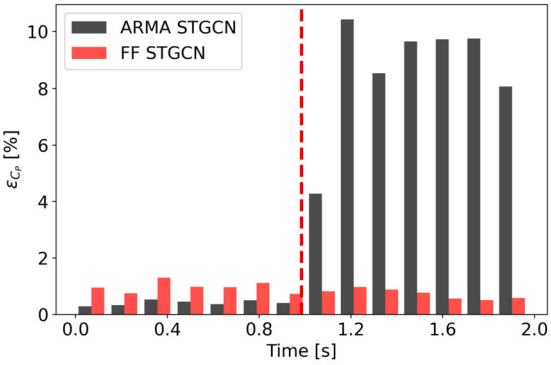


Fig. 15. Validation signal 1 - DS type: Evolution of MAPE of C_P with ARMA and Feedforward model. Vertical dashed line marks the time instance where the ARMA model transitions to autoregressive C_P predictions.

accumulate too fast, as in systems with lower reduced frequency. Also, insufficient convergence of the CFD solution may affect error accumulation in regions with complex flow patterns, suggesting that improving solution stability may help mitigate this problem.

4.4. Sensitivity analysis to elliptical and hyperbolic regions

The framework has the ability to independently detect the elliptic or hyperbolic nature of the flow, as demonstrated by a sensitivity analysis in which localized perturbations were applied separately in subsonic and supersonic regions (see Fig. 20). Specifically, we introduced a 100% instantaneous increase in C_P at a single surface node. Two cases are examined: (a) the perturbed node lies inside a subsonic area downstream of the shock, and (b) the node is inside the supersonic region immediately upstream of the shock. The perturbation is injected into the ARMA variant, and its spatial propagation is tracked for one time-step through the decoder. Our analysis showed distinct differences in the response to these perturbations: In the subsonic (elliptical) region, the perturbation affected the pressure distribution in all directions around the perturbed node, reflecting the expected elliptical behavior characterized by long-range spatial correlation. In the supersonic (hyperbolic) region, the perturbation only affected nodes downstream of the perturbed node, consistent with hyperbolic behavior where information propagates primarily in the downstream direction. Closer inspection reveals that the affected region is not only biased in the downstream direction, but also laterally constrained by the local Mach cone. In weakly supersonic pockets where the local Mach number lies between $M_{\text{loc}} = 1.0$ and 1.2, the corresponding Mach angle spans approximately 60° to 90°. As illustrated in Fig. 20, the high-ΔC_P response falls almost entirely inside this 2μ-wide cone-shaped sector. These results confirm that the convolution kernels in our graph-based architecture adaptively encode flow-specific properties without explicit a priori definitions, successfully capturing the essential physical behavior of transonic flows.

4.5. Extrapolation to off-design flow conditions

The Feedforward AeroNet was trained exclusively on URANS data obtained at the reference condition $M = 0.74$, but in practical applications the wing will often operate at significantly different freestream parameters. To assess how far the network can be pushed outside its training envelope, we confronted it with wind tunnel measurements of pitch-only motion from Piatak and Cleckner [57]. These experiments were designed to generate test cases for validation purposes and have since become standard benchmarks in the aeroelastic community. The corresponding datasets, developed as part of the AIAA Aeroelastic Prediction Workshop, have been widely studied both computationally and experimentally precisely because they involve challenging transonic phenomena, such as shock-induced separation, flow reattachment and nonlinear unsteady effects, which are difficult to predict reliably. This complexity has motivated several initiatives aimed at benchmarking reduced-order and high-fidelity models under standardized test conditions [55,58–60].

The experimental database provides mean pressure coefficient $(C_P)_{Mean}$ distributions extracted at 60% span for two off-design Mach numbers, $M = 0.70$ and $M = 0.85$, respectively, tested at mean incidences (AoA_∞) of 0 deg and 5 deg, while keeping the pitch amplitude ($a_\theta = 1.03$ [deg]) and reduced frequency ($k_\theta = 0.439$) identical for the two maneuvers. To ensure a consistent basis for comparison, the predicted surface pressure distributions from AeroNet were averaged in time over a full oscillation cycle, yielding mean values comparable to the experimental data. Note that the current implementation has no explicit input of flow parameters-the current network receives only the instantaneous wing kinematics $\{\theta, \dot{\theta}, \ddot{\theta}\}$ as input. Variations in M_∞ or Re therefore affect the model only indirectly through the rescaled timestep, which enters through the first and second order derivatives of the motion. In practice, this means that any prediction at an off-design Mach number is, strictly speaking, an extrapolation task. As with all data-driven models, robust performance can be expected only within the input space spanned by the training data; good generalization requires that the training set include sufficiently rich and diverse examples covering the desired range of operating conditions.

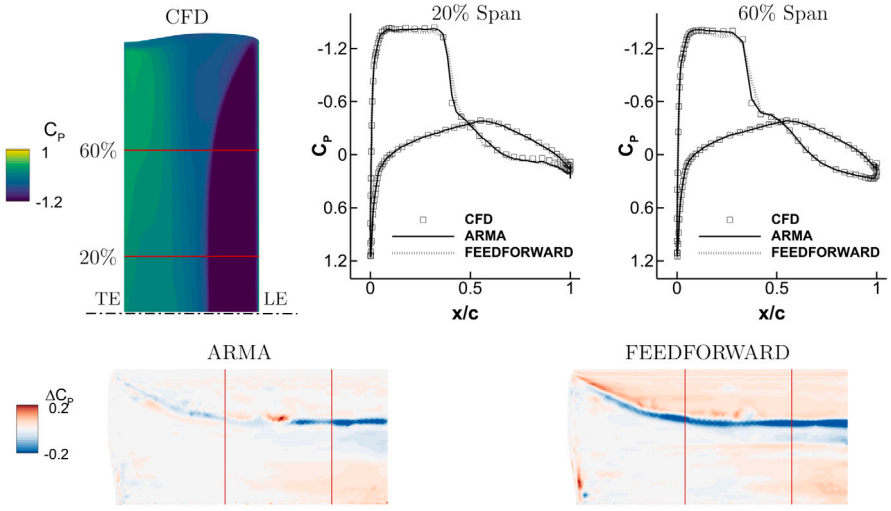
Fig. 21 contrasts the time-averaged AeroNet predictions (continuous curves) with the corresponding experimental data (symbols). For the subcritical case at $M = 0.70$, the network shows good agreement with the measurements, especially at $AoA_\infty = 0$ deg where the model follows the nearly flat pressure plateau. At higher angles of incidence, the distribution on the lower surface is correct, but the shock, which is relatively weak at this Mach number, appears at about $x/c \approx 0.35$ instead of 0.18, and the associated compression is slightly underpredicted. The trailing edge pressure levels are correctly matched, suggesting that this modest deviation from the design Mach number is well accommodated by the network's learned manifold. At $M = 0.85$, however, the discrepancies become more pronounced. The model tends to shift the shock aft compared to the experimental observations. For $AoA_\infty = 5$ deg the discrepancy becomes more significant: the network overestimates the suction on the lower surface downstream of the leading edge, probably due to an incomplete representation of the shock-induced separation, which becomes more relevant at this higher Mach number; a result consistent with the stronger and more upstream shock motion that the network never encountered during training.

The encouraging performance at $M = 0.70$ suggests that moderate deviations from the design point remain within the latent manifold learned by the network, while the larger deviations at $M = 0.85$ expose its limited awareness of transonic shock dynamics when freestream parameters are kept invisible. Making M and Re explicit conditioning variables is the next logical step for improving generalizability.

4.6. Comparison between AeroNet and DMDc

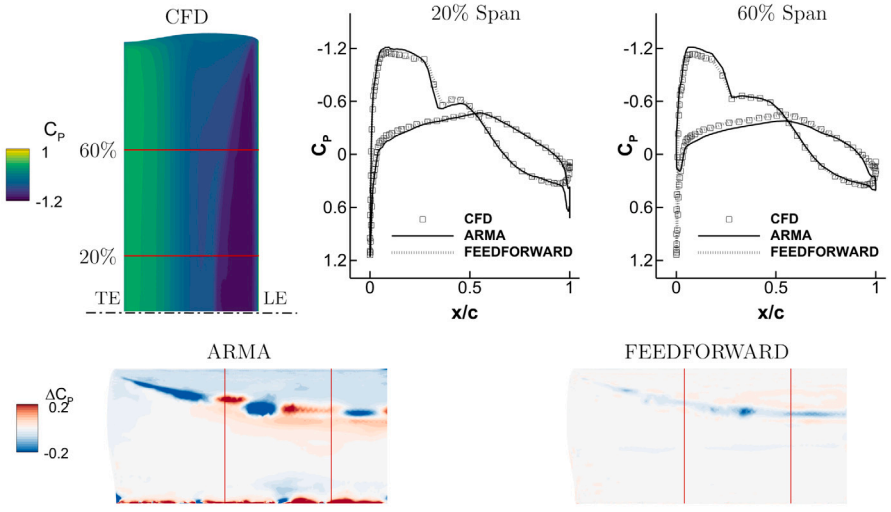
To contextualize the performance of the proposed AeroNet models, we benchmark them against a well-established reduced-order technique-the DMDc, given its similarity to the autoregressive mechanism inherent to our AeroNet ARMA variant. The DMDc implementation follows the procedure detailed by Fonzi et al. [7], which was previously employed successfully on the same benchmark wing under similar flow conditions. For optimal dimensionality reduction and computational efficiency, a total of 25 modes were retained based on a Frobenius norm threshold of 0.1, ensuring a low reconstruction error on the training dataset.

Figs. 22 and 23 compare the predicted C_L and C_M histories from the DMDc and AeroNet models against the CFD reference data for the two validation signals detailed in Table 1. For the DS validation signal (Fig. 22), all models exhibit good agreement with the CFD reference, with the AeroNet FF variant demonstrating notably superior accuracy. This observation is quantitatively supported by the metrics provided in



Time Instance (a)

Fig. 16. Validation signal 1 - DS type: ARMA vs. Feedforward model using STGCN temporal layer on C_P prediction. The lower surface of the wing is shown. LE: Leading Edge. TE: Trailing Edge. Dash-dot line indicates the symmetry plane.



Time Instance (b)

Fig. 16. (*continued*)

Table 5
Comparison of MAPE, R2, and RMSE for C_P predictions in the DMDc and AeroNet ARMA - FF models with STGCN temporal layer for DS and SH validation signals.

Model	MAPE		R2		RMSE	
	DS	SH	DS	SH	DS	SH
DMDc	4.7961	6.4841	0.9137	0.8574	0.0608	0.1362
Aeronet ARMA	6.9381	5.7971	0.8571	0.8648	0.0938	0.0844
Aeronet FF	0.8524	0.9975	0.9918	0.9897	0.0163	0.0181

Table 5. For the SH signal, characterized by higher-frequency oscillations (Fig. 23), both AeroNet variants outperform the DMDc method. In particular, AeroNet effectively captures dynamic peaks and mitigates the error propagation that noticeably affects the pitching moment predictions. These qualitative insights are further confirmed by the error metric summarized in Table 5.

To complement the error–propagation analysis, we quantified the stability of the learned latent-space dynamics by estimating the largest Lyapunov exponent, λ_1, for each model and excitation signal on the predicted C_P. The error is scaled with respect to the cell area. Table 6 lists the resulting values together with a qualitative stability assessment: positive λ_1 indicates exponential divergence (chaotic/unstable behavior), whereas a negative value reflects asymptotic convergence to an attractor.

For both signals, the feedforward model has a Lyapunov exponent of zero by definition, as the network never feeds back its own outputs into the next step, local perturbations neither grow nor decay and no systematic error build-up occurs, so the system remains neutrally stable. In contrast, for the DS signal, both ARMA and DMDc variants yield a positive λ_1, so that perturbations double every 1.3 s and 1.1 s respectively. This confirms that the autoregressive feedback loop amplifies numerical noise and modeling inaccuracies—a mechanism responsible for the error increase in Fig. 15. The SH case imposes a higher reduced

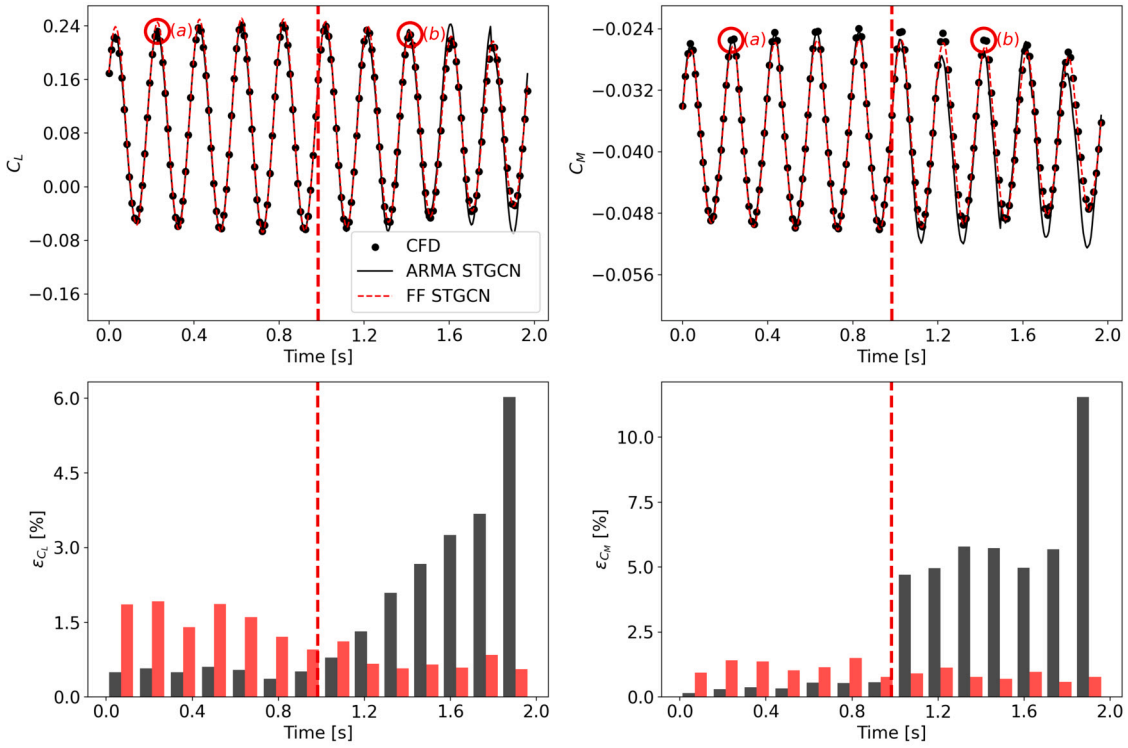


Fig. 17. Validation signal 2 - SH type: ARMA vs. Feedforward model using STGCN temporal layer on C_L and C_M predictions. Red circles denote the points used for plotting the C_P distribution. Vertical dashed line marks the time instance where the ARMA model transitions to autoregressive C_P predictions.

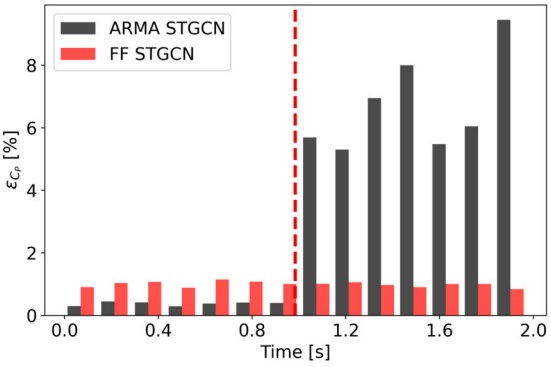


Fig. 18. Validation signal 2 - SH type: Evolution of MAPE of C_P with ARMA and Feedforward model. Vertical dashed line marks the time instance where the ARMA model transitions to autoregressive C_P predictions.

Table 6
Estimated largest Lyapunov exponents λ_1 for AeroNet and DMDc models under DS and SH excitations.

Signal Type	Model	λ_1 [s^{-1}]	Stability
DS	DMDc	+0.581	Unstable
DS	AeroNet ARMA	+0.513	Unstable
DS	AeroNet FF	0	Neutrally Stable
SH	DMDc	+0.844	Unstable
SH	AeroNet ARMA	+0.795	Unstable
SH	AeroNet FF	0	Neutrally Stable

frequency and a stronger shock oscillation. Here, the ARMA model becomes even more unstable, with $\lambda_1 \approx 0.80\ s^{-1}$, implying that trajectory separation doubles roughly every 0.9 s. Similar considerations can be drawn for DMDc model, with perturbations double every 0.8 s. The feedforward formulation maintains forecast fidelity over extended rollouts, whereas the autoregressive ARMA is prone to error accumulation unless external correction (e.g., periodic sensor feedback) is available. The DMDc model exhibits slightly faster prediction degradation. Unlike neural networks, which can learn to adaptively dampen or stabilize unsteady behavior, DMDc relies on a purely linear, open-loop operator. This makes it inherently more susceptible to noise amplification and model drift, especially when extrapolating beyond the training region or under higher frequency oscillations, as seen in the SH scenario.

The differences between AeroNet and DMDc models can be attributed to their underlying methodologies. DMDc inherently carries specific constraints, such as the requirement for explicit stabilization procedures and calibration within a relatively narrow operating envelope. Consequently, DMDc is highly sensitive to deviations from the training conditions, limiting its general robustness. In contrast, AeroNet models attain comparable or better accuracy without relying on such stringent stabilization and tuning steps. This allows them to perform robustly across a wider range of dynamic conditions.

The trade-off of employing AeroNet over traditional methods such as DMDc lies in interpretability. DMDc model expresses explicitly the system dynamics via A and B matrices and modal decompositions, thereby providing direct analysis of the system behavior and physical insights. In contrast, AeroNet employs deep neural architectures whose internal representations do not necessarily correspond to physically meaningful quantities. The network operates as a black box, optimizing its parameters solely to reduce predictive error, without explicitly encoding the governing flow mechanisms. Therefore, while AeroNet demonstrates superior adaptability and predictive accuracy in complex, dynamic scenarios, this comes at the cost of reduced interpretability.

4.7. Computing cost analysis

A comprehensive analysis of computational costs was performed to compare the efficiency of the proposed AeroNet model with that of a higher-order approach and DMDc technique, as shown in Table 7. A single CFD run on a high-performance computing system, using an Intel Skylake-based architecture with 3 nodes and 40 CPU cores per node, typically requires around 6,000 CPU hours. Generating the entire dataset

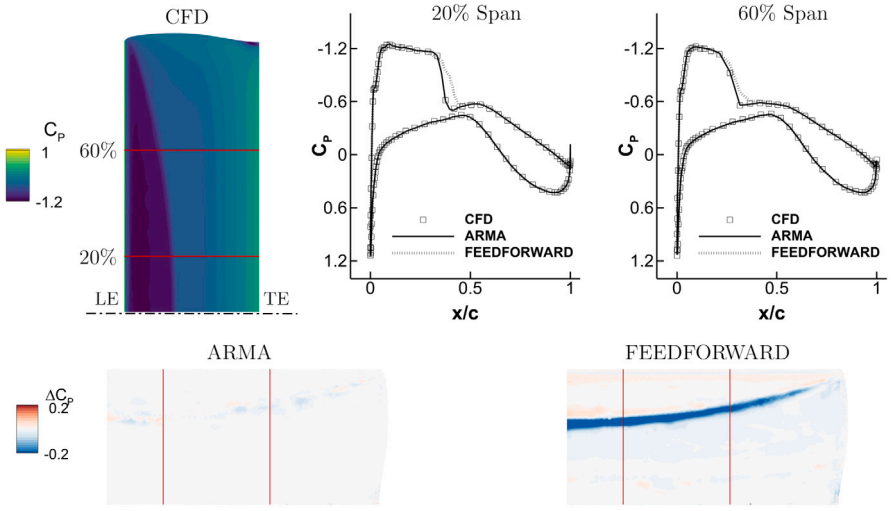


Fig. 19. Validation signal 2 - SH type: ARMA vs. Feedforward model using STGCN temporal layer on C_P prediction. The upper surface of the wing is shown. LE: Leading Edge. TE: Trailing Edge. Dash-dot line indicates the symmetry plane.

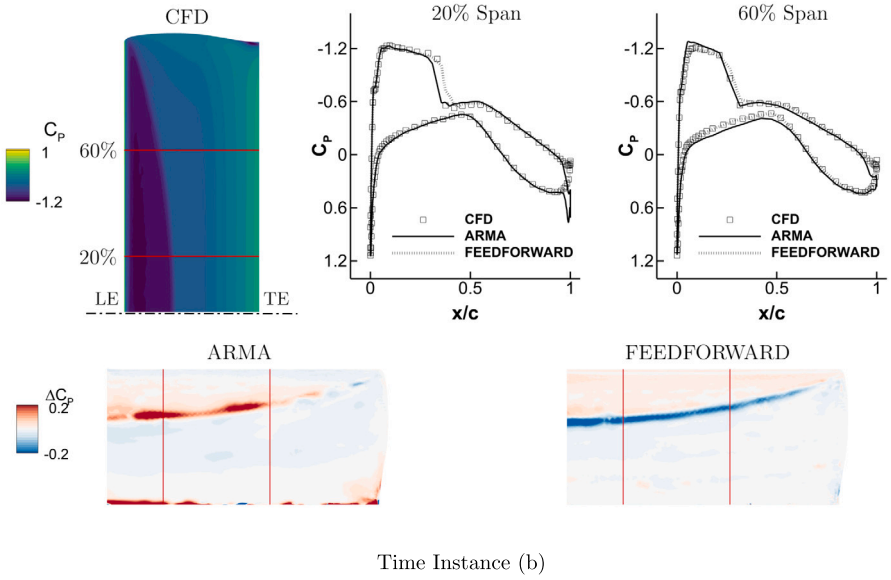


Fig. 19. (*continued*)

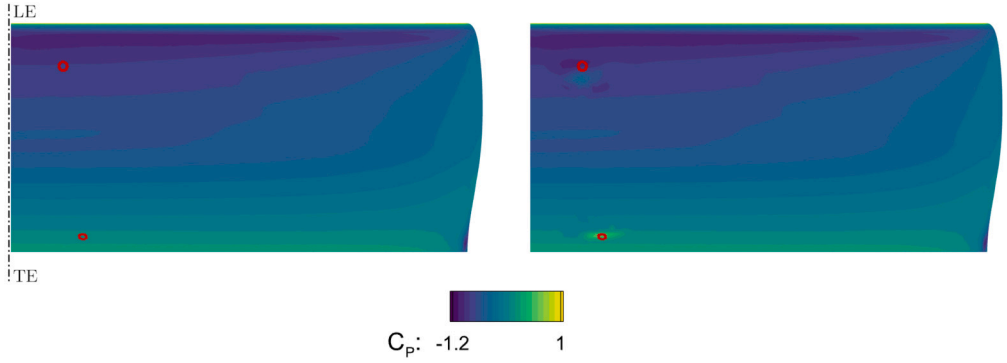


Fig. 20. Instantaneous C_P perturbation study demonstrating model sensitivity to elliptical and hyperbolic flow behavior.

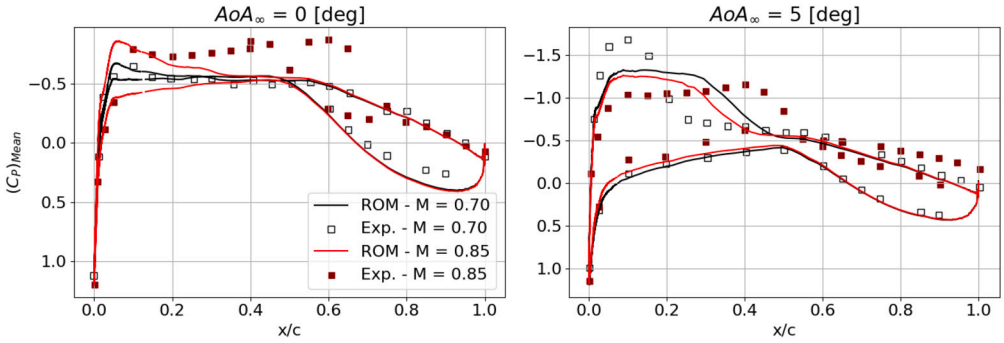


Fig. 21. Time-averaged surface pressure coefficient at 60% span for pitch-only oscillations with $a_\theta = 1.03$ [deg] and $k_\theta = 0.439$.

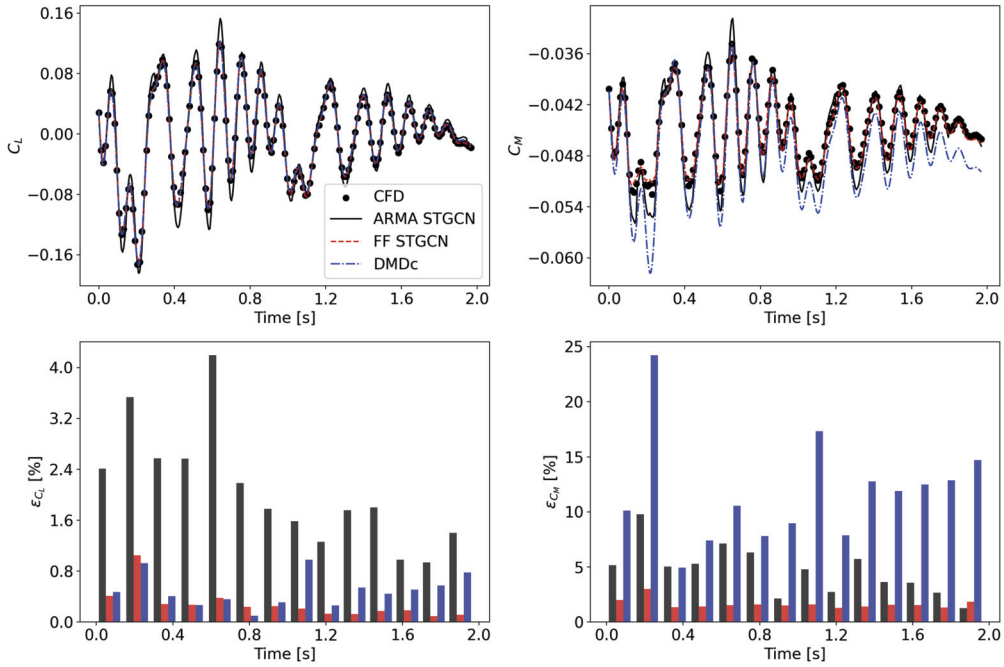


Fig. 22. Validation signal 1 - DS type: DMDc vs. AeroNet models using STGCN temporal layer on C_L and C_M predictions.

Table 7
Computing cost comparison between AeroNet models, DMDc and CFD. FF: FeedForward.

CFD (CPU hours)		AeroNet (GPU hours)				
Simulation		Pre-Trained AE	Training		Prediction	
(12 runs)	(1 run)	(Optimization + Training)	(ARMA)	(FF)	(ARMA)	(FF)
75,000	6,000	42.6	35.2	33.1	0.03	0.03

DMDc (CPU hours)	
Training	Prediction
0.15	0.003

demands approximately 75,000 CPU hours. In contrast, the proposed framework predicts a full signal in under two minutes on a local machine with a NVIDIA RTX A4000 GPU, with a per-timestep inference time of 0.15 seconds (6.67 frames-per-second), yielding over 99% computational savings with respect to CFD. The DMDc model, once trained, is still faster in wall-clock terms (60 frames-per-second), but (as shown in Section 4.6) at the cost of lower prediction accuracy. Finally, we note that the large up-front expense of producing high-fidelity training data motivates future work on transfer-learning or active-learning strategies.

5. Conclusions

This study introduced a framework for predicting unsteady transonic wing pressure distributions, integrating an AE architecture with GCN and graph-based temporal layers to capture time dependencies. Tested on a grid consisting of 86,840 surface points, the proposed model effectively compresses high-dimensional C_P distribution data into a lower-dimensional latent space using the AE, preserving essential features for accurate representation. The GCN layers are well-suited for handling

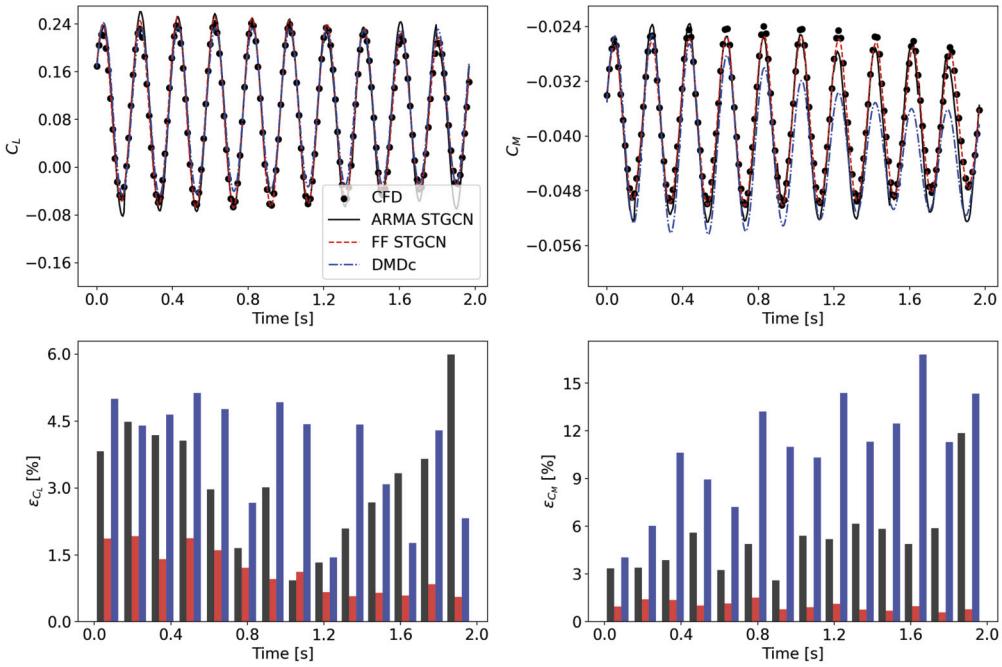


Fig. 23. Validation signal 2 - SH type: DMDc vs. AeroNet models using STGCN temporal layer on C_L and C_M predictions.

the unstructured grids characteristic of aerodynamic data, while the temporal layers capture and leverage temporal dependencies for robust forecasting of wing C_P distributions.

Our results demonstrated that this integrated approach can achieve an accuracy comparable to traditional CFD methods for the dynamic conditions included in the training and validation datasets, while significantly reducing computational costs, requiring initially about 76 GPU hours for training and then less than 2 minutes to predict an entire signal. We evaluated two architectures, the feedforward model and the ARMA model, using four different temporal layers (GRU, LSTM, STGCN, ATTENTION), with the STGCN layer consistently delivering the most accurate results across the validation signals. The feedforward model demonstrated clear advantages in terms of predictive accuracy and stability, particularly in avoiding the error propagation inherent in the ARMA model. By not relying on its own predictions for subsequent inputs, the feedforward model is better suited for dynamic simulations ranging from steady-state to high reduced frequency, showing greater accuracy in predicting complex flow features, such as shock waves and flow separation. The ARMA model, while capable of capturing general trends, is more prone to error accumulation, particularly in scenarios with high-frequency oscillations or rapid changes in aerodynamic forces. Nonetheless, when using ground-truth inputs, the ARMA model can yield highly accurate results, underscoring its potential in scenarios with reliable data inputs. Both AeroNet variants outperformed the established DMDc approach, particularly in high-frequency, dynamically complex situations. Notably, AeroNet achieves superior accuracy without requiring explicit stabilization or calibration, indicating greater robustness and adaptability.

Sensitivity analyses demonstrated that the proposed framework inherently distinguishes between elliptic and hyperbolic flow characteristics typical of transonic flows. Localized perturbations in subsonic (elliptical) regions affect global pressure distributions, while perturbations in supersonic (hyperbolic) regions propagate downstream within localized Mach cones. These findings confirm AeroNet capability to accurately capture the distinct physical behaviors associated with different aerodynamic flow regimes.

The extrapolation capacity to off-design conditions (e.g., varying freestream Mach or Reynolds number) remains limited and should be addressed in future work through explicit conditioning on flow parameters or multi-fidelity learning strategies. Additionally, scalability to larger grids and different graph structures will be explored. GCNs inherently support various graph configurations, but expanding the model to handle new spatial structures or larger meshes may require techniques like subgraph splitting or padding to ensure input compatibility. As grid size increases, deeper networks and additional pooling layers will be necessary to capture long-range dependencies efficiently, while maintaining computational feasibility.

CRediT authorship contribution statement

Gabriele Immordino: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Andrea Vaiuso:** Writing – original draft, Methodology, Conceptualization. **Andrea Da Ronch:** Writing – review & editing, Supervision, Resources, Project administration, Conceptualization. **Marcello Righi:** Writing – review & editing, Supervision, Resources, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Gabriele Immordino reports financial support was provided by ZHAW University of Applied Sciences. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by Digitalization Initiative of the Zurich Higher Education Institutions (DIZH) grant 9710.Z.12.P.0003.05 from Zurich University of Applied Sciences (ZHAW). The authors also acknowledge the University of Southampton for granting access to the IRIDIS High Performance Computing Facility and its associated support services.

Table B.8
Layer structure and output dimensions for the ARMA model, detailing the two encodings, temporal, and decoding layers used for predicting pressure distribution.

	Layer Type	Output Size
Encoding A	Input	$m \times 3 \times 86840 \times 8$
	GCN	$m \times 3 \times 86840 \times 256$
	GCN	$m \times 3 \times 86840 \times 224$
	GCN	$m \times 3 \times 86840 \times 96$
	Pooling 1	$m \times 3 \times 28600 \times 96$
	GCN	$m \times 3 \times 28600 \times 64$
	Pooling 2	$m \times 3 \times 9600 \times 64$
	GCN	$m \times 3 \times 9600 \times 368$
Encoding B	Input	$m \times 3 \times 86840 \times 1$
	GCN	$m \times 3 \times 86840 \times 256$
	GCN	$m \times 3 \times 86840 \times 224$
	GCN	$m \times 3 \times 86840 \times 96$
	Pooling 1	$m \times 3 \times 28600 \times 96$
	GCN	$m \times 3 \times 28600 \times 64$
	Pooling 2	$m \times 3 \times 9600 \times 64$
	GCN	$m \times 3 \times 9600 \times 368$
Concatenate Block – Output: $m \times 3 \times 9600 \times 736$		
Temporal Layer – Output: $m \times 9600 \times 368$		
Decoding	**Layer Type**	**Output Size**
	GCN	$m \times 9600 \times 368$
	Unpooling 2	$m \times 28600 \times 368$
	GCN	$m \times 28600 \times 64$
	Unpooling 1	$m \times 86840 \times 64$
	GCN	$m \times 86840 \times 96$
	GCN	$m \times 86840 \times 224$
	GCN	$m \times 86840 \times 256$
	Output	$m \times 86840 \times 1$

Appendix A. Schroeder-phased harmonic signal formulation

The Schroeder-phased harmonic signal is utilized in this study to improve the robustness and generalizability of the model by covering a wide frequency spectrum. These signals are constructed by summing sinusoidal components, where the phases are optimized to minimize the overall peak amplitude. This results in an evenly distributed energy spectrum, which is advantageous for training the model to handle various frequency interactions and reduces the risk of overfitting. To cover this broad frequency range with minimal peak amplitude, a total of 9 harmonics is selected, with $M = 9$.

Both damped Schroder-phased harmonic (DS) and undamped Schroder-phased harmonic (US) signals are used to model the wing displacement, whether it be pitch $\theta(t)$ or plunge $\xi(t)$. The US signal uniformly distributes energy across the frequency spectrum and is defined as:

$$\theta_{US}(t) = \sum_{m=1}^{M} a_m \sin\left((m+1)\omega_m t + \phi_m\right) \tag{A.1}$$

where a_m represents the amplitude of the m-th component, ω_m denotes the angular frequency, and ϕ_m corresponds to the phase of the m-th sinusoidal component.

For transient response analysis, the DS signal simulates amplitude decay over time, incorporating a damping function:

$$\theta_{DS}(t) = \sum_{m=1}^{M} \left(\left(\frac{a_{\text{end}} - a_0}{t_{\text{end}} - t_0}(t - t_0) + a_0 \right) \sin\left((m+1)\omega_m t + \phi_m\right) \right) \tag{A.2}$$

where a_0 and a_{end} denote the initial and final amplitudes, respectively, and t_0 and t_{end} are the corresponding time intervals. The damping is designed such that at the final time step, the amplitude is reduced to $0.1\,a_0$, ensuring transient behaviors are effectively captured.

The phases ϕ_m are calculated to minimize constructive interference between the sinusoidal components, flattening the overall spectrum:

$$\phi_m = -\frac{m(m+1)\pi}{M} \tag{A.3}$$

Table B.9
Layer structure and output dimensions for the Feedforward model, detailing the encoding, temporal, and decoding layers used for predicting pressure distribution.

	Layer Type	Output Size
Encoding A	Input	$m \times 3 \times 86840 \times 8$
	GCN	$m \times 3 \times 86840 \times 256$
	GCN	$m \times 3 \times 86840 \times 224$
	GCN	$m \times 3 \times 86840 \times 96$
	Pooling 1	$m \times 3 \times 28600 \times 96$
	GCN	$m \times 3 \times 28600 \times 64$
	Pooling 2	$m \times 3 \times 9600 \times 64$
	GCN	$m \times 3 \times 9600 \times 368$
Temporal Layer – Output: $m \times 9600 \times 368$		
Decoding	**Layer Type**	**Output Size**
	GCN	$m \times 9600 \times 368$
	Unpooling 2	$m \times 28600 \times 368$
	GCN	$m \times 28600 \times 64$
	Unpooling 1	$m \times 86840 \times 64$
	GCN	$m \times 86840 \times 96$
	GCN	$m \times 86840 \times 224$
	GCN	$m \times 86840 \times 256$
	Output	$m \times 86840 \times 1$

Appendix B. Models architecture

This section outlines the architecture and training process for both the ARMA and feedforward models used in this study, as detailed in Tables B.8 and B.9. The ARMA model in Table B.8 combines autoregressive components with GCN layers and STGCN temporal layer to capture both spatial and temporal dynamics, featuring 5,775,023 trainable weights. In contrast, the feedforward model in Table B.9 avoids using previous predictions, which helps prevent error accumulation over time. This model has 1,962,111 trainable weights. Both models utilize

a pre-trained AE for dimensionality reduction. Specifically, the Encoding A and Decoding layers in both architectures are optimized based on the pre-trained AE, while Encoding B mirrors the structure of Encoding A, ensuring consistent feature extraction across different model variants. Additionally, the concatenation block in both models concatenates the encodings from the previous three timesteps, enabling the temporal layer to effectively capture and process the sequential dependencies within the data.

During the backpropagation phase, the ADAptive Moment Estimation (Adam) optimizer [61] was employed to fine-tune the neural network weights and minimize the `MAE` loss function. The learning rate was set to 0.001. A batch size m of 1 was found to yield the most accurate results. The training process was carried out over 50 epochs.

Data availability

Data will be made available on request.

References

[1] J. Blazek, Computational Fluid Dynamics: Principles and Applications, Butterworth-Heinemann, 2015.

[2] A. Chatterjee, An introduction to the proper orthogonal decomposition, in: Current Science, 2000, pp. 808–817.

[3] J.H. Tu, Dynamic mode decomposition: Theory and applications, Ph.D. thesis, Princeton University, 2013.

[4] J.L. Proctor, S.L. Brunton, J.N. Kutz, Dynamic mode decomposition with control, SIAM J. Appl. Dyn. Syst. 15 (2016) 142–161.

[5] K.J. Asztalos, S.T. Dawson, D.R. Williams, Modeling the flow state sensitivity of actuation response on a stalled airfoil, AIAA J. 59 (2021) 2901–2915.

[6] B. Swaminathan, J.G. Manathara, A. Vinayagam, Application of dynamic mode decomposition with control (dmdc) for aircraft parameter estimation, IFAC-PapersOnLine 55 (2022) 789–794.

[7] N. Fonzi, S.L. Brunton, U. Fasel, Data-driven modeling for transonic aeroelastic analysis, J. Aircr. 61 (2024) 625–637.

[8] J.C. Rains, D. Huang, Y. Wang, Residual dynamic mode decomposition with control for nonlinear aeroservoelastic applications, in: AIAA SciTech 2024 Forum, 2024, p. 2264.

[9] N. Fonzi, S.L. Brunton, U. Fasel, Data-driven nonlinear aeroelastic models of morphing wings for control, Proc. Royal Soc. A 476 (2020) 20200079.

[10] C. Sabater, P. Stürmer, P. Bekemeyer, Fast predictions of aircraft aerodynamics using deep-learning techniques, AIAA J. 60 (2022) 5249–5261, https://doi.org/10.2514/1.J061234.

[11] R. Castellanos, J.B. Varela, A. Gorgues, E. Andrés, An assessment of reduced-order and machine learning models for steady transonic flow prediction on wings, in: ICAS 2022, 2022.

[12] G. Immordino, A. Da Ronch, M. Righi, Steady-state transonic flowfield prediction via deep-learning framework, AIAA J. (2024) 1–17, https://doi.org/10.2514/1.J063545.

[13] J. Tompson, K. Schlachter, P. Sprechmann, K. Perlin, Accelerating Eulerian fluid simulation with convolutional networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 3424–3433.

[14] 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.

[15] M. Gori, G. Monfardini, F. Scarselli, A new model for learning in graph domains, in: Proceedings, 2005 IEEE International Joint Conference on Neural Networks, 2005, vol. 2, IEEE, 2005, pp. 729–734.

[16] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE Trans. Neural Netw. 20 (2008) 61–80, https://doi.org/10.1109/TNN.2008.2005605.

[17] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (2020) 4–24, https://doi.org/10.1109/TNNLS.2020.2978386.

[18] Z. Zhang, P. Cui, W. Zhu, Deep learning on graphs: a survey, IEEE Trans. Knowl. Data Eng. 34 (2020) 249–270, https://doi.org/10.1109/TKDE.2020.2981333.

[19] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: a review of methods and applications, AI Open 1 (2020) 57–81, https://doi.org/10.1016/j.aiopen.2021.01.001.

[20] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907, 2016, https://doi.org/10.48550/arXiv.1609.02907.

[21] X. Jin, P. Cheng, W.-L. Chen, H. Li, Prediction model of velocity field around circular cylinder over various Reynolds numbers by fusion convolutional neural networks based on pressure on the cylinder, Phys. Fluids 30 (2018) 047105, https://doi.org/10.1063/1.5024595.

[22] K. Fukami, K. Fukagata, K. Taira, Super-resolution reconstruction of turbulent flows with machine learning, J. Fluid Mech. 870 (2019) 106–120, https://doi.org/10.1017/jfm.2019.238.

[23] N. Omata, S. Shirayama, A novel method of low-dimensional representation for temporal behavior of flow fields using deep autoencoder, AIP Adv. 9 (2019) 015006, https://doi.org/10.1063/1.5067313.

[24] J.-Z. Peng, S. Chen, N. Aubry, Z.-H. Chen, W.-T. Wu, Time-variant prediction of flow over an airfoil using deep neural network, Phys. Fluids 32 (2020) 123602, https://doi.org/10.1063/5.0022222.

[25] V. Rozov, C. Breitsamter, Data-driven prediction of unsteady pressure distributions based on deep learning, J. Fluids Struct. 104 (2021) 103316, https://doi.org/10.1016/j.jfluidstructs.2021.103316.

[26] R. Han, Y. Wang, Y. Zhang, G. Chen, A novel spatial-temporal prediction method for unsteady wake flows based on hybrid deep neural network, Phys. Fluids 31 (2019), https://doi.org/10.1063/1.5127247.

[27] E. Saetta, R. Tognaccini, G. Iaccarino, Machine learning to predict aerodynamic stall, Int. J. Comput. Fluid Dyn. 36 (2022) 641–654, https://doi.org/10.1080/10618562.2023.2171021.

[28] D. Massegur Sampietro, A. Da Ronch, Graph convolutional multi-mesh autoencoder for steady transonic aircraft aerodynamics, Mach. Learn.: Sci. Technol. (2023), https://doi.org/10.1088/2632-2153/ad36ad.

[29] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (2006) 504–507, https://doi.org/10.1126/science.1127647.

[30] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th International Conference on Machine Learning, 2008, pp. 1096–1103.

[31] G. Immordino, A. Vaiuso, A. Da Ronch, M. Righi, Predicting transonic flowfields in non–homogeneous unstructured grids using autoencoder graph convolutional networks, J. Comput. Phys. (2025) 113708.

[32] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780, https://doi.org/10.1162/neco.1997.9.8.1735.

[33] K. Cho, B. Van Merriënboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: encoder-decoder approaches, arXiv preprint arXiv:1409.1259, 2014, https://doi.org/10.3115/v1/W14-4012.

[34] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, S. Valaee, Recent advances in recurrent neural networks, arXiv preprint arXiv:1801.01078, 2017, https://doi.org/10.48550/arXiv.1801.01078.

[35] K. Li, J. Kou, W. Zhang, Deep neural network for unsteady aerodynamic and aeroelastic modeling across multiple Mach numbers, Nonlinear Dyn. 96 (2019) 2157–2177, https://doi.org/10.1007/s11071-019-04915-9.

[36] S.L. Brunton, B.R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, Annu. Rev. Fluid Mech. 52 (2020) 477–508, https://doi.org/10.1146/annurev-fluid-010719-060214.

[37] Q. Wang, C.E. Cesnik, K. Fidkowski, Multivariate recurrent neural network models for scalar and distribution predictions in unsteady aerodynamics, in: AIAA Scitech 2020 Forum, 2020, p. 1533.

[38] A. Mannarino, P. Mantegazza, Nonlinear aeroelastic reduced order modeling by recurrent neural networks, J. Fluids Struct. 48 (2014) 103–121, https://doi.org/10.1016/j.jfluidstructs.2014.02.016.

[39] D. Massegur, A. Da Ronch, Recurrent graph convolutional multi-mesh autoencoder for unsteady transonic aerodynamics, J. Fluids Struct. 131 (2024) 104202.

[40] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017), https://doi.org/10.48550/arXiv.1706.03762.

[41] J. Cheng, L. Dong, M. Lapata, Long short-term memory-networks for machine reading, arXiv preprint arXiv:1601.06733, 2016, https://doi.org/10.18653/v1/D16-1053.

[42] X. Cheng, F. Shi, M. Zhao, G. Li, H. Zhang, S. Chen, Temporal attention convolutional neural network for estimation of icing probability on wind turbine blades, IEEE Trans. Ind. Electron. 69 (2021) 6371–6380, https://doi.org/10.1109/TIE.2021.3090702.

[43] X. Han, H. Gao, T. Pfaff, J.-X. Wang, L.-P. Liu, Predicting physics in mesh-reduced space with temporal attention, arXiv preprint arXiv:2201.09113, 2022, https://doi.org/10.48550/arXiv.2201.09113.

[44] J. Du, X. Li, S. Dong, Z. Liu, G. Chen, A novel attention enhanced deep neural network for hypersonic spatiotemporal turbulence prediction, Phys. Fluids 36 (2024), https://doi.org/10.1063/5.0210966.

[45] J. Bai, J. Zhu, Y. Song, L. Zhao, Z. Hou, R. Du, H. Li, A3t-gcn: attention temporal graph convolutional network for traffic forecasting, ISPRS Int. J. Geo-Inf. 10 (2021) 485, https://doi.org/10.48550/arXiv.2006.11583.

[46] D.K. Duvenaud, D. Maclaurin, J. Iparraguirre, R. Bombarell, T. Hirzel, A. Aspuru-Guzik, R.P. Adams, Convolutional networks on graphs for learning molecular fingerprints, Adv. Neural Inf. Process. Syst. 28 (2015), https://doi.org/10.48550/arXiv.1509.09292.

[47] M. Fey, J.E. Lenssen, Fast graph representation learning with pytorch geometric, https://doi.org/10.48550/arXiv.1903.02428, arXiv:1903.02428, 2019.

[48] D.K. Hammond, P. Vandergheynst, R. Gribonval, Wavelets on graphs via spectral graph theory, Appl. Comput. Harmon. Anal. 30 (2011) 129–150, https://doi.org/10.1016/j.acha.2010.04.005.

[49] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, W.-c. Woo, Convolutional lstm network: a machine learning approach for precipitation nowcasting, in: C. Cortes, N.

Lawrence, D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems, vol. 28, Curran Associates, Inc., 2015.
[50] Y. Seo, M. Defferrard, P. Vandergheynst, X. Bresson, Structured sequence modeling with graph convolutional recurrent networks, in: Neural Information Processing: 25th International Conference, ICONIP 2018, Proceedings, Part I 25, Siem Reap, Cambodia, December 13–16, 2018, Springer, 2018, pp. 362–373.
[51] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting, arXiv preprint arXiv:1709.04875, 2017, https://doi.org/10.24963/ijcai.2018/505.
[52] R. De Maesschalck, D. Jouan-Rimbaud, D.L. Massart, The Mahalanobis distance, Chemom. Intell. Lab. Syst. 50 (2000) 1–18, https://doi.org/10.1016/j.patcog.2008.05.018.
[53] G. Quaranta, P. Masarati, P. Mantegazza, A conservative mesh-free approach for fluid structure problems in coupled problems, in: International Conference for Coupled Problems in Science and Engineering, Santorini, Greece, 2005, pp. 24–27.
[54] 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.
[55] J. Heeg, Overview of the aeroelastic prediction workshop, in: 51st AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition, 2013, p. 783.
[56] T.D. Economon, F. Palacios, S.R. Copeland, T.W. Lukaczyk, J.J. Alonso, Su2: an open-source suite for multiphysics simulation and design, AIAA J. 54 (2016) 828–846, https://doi.org/10.2514/1.J053813.
[57] D.J. Piatak, C.S. Cleckner, Oscillating turntable for the measurement of unsteady aerodynamic phenomena, J. Aircr. 40 (2003) 181–188.
[58] J. Heeg, Overview and lessons learned from the aeroelastic prediction workshop, in: 54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, 2013, p. 1798.
[59] J. Heeg, P. Chwalowski, D.E. Raveh, A. Jirasek, M. Dalenbring, Overview and data comparisons from the 2nd aeroelastic prediction workshop, in: 34th AIAA Applied Aerodynamics Conference, 2016, p. 3121.
[60] P. Chwalowski, B. Stanford, K. Jacobson, L. Poplingher, D.E. Raveh, A. Jirasek, G. Pagliuca, R. Marcello, Flutter prediction report in support of the high angle working group at the third aeroelastic prediction workshop, in: AIAA SciTech 2024 Forum, 2024, p. 0418.
[61] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint arXiv:1412.6980, 2014, https://doi.org/10.48550/arXiv.1412.6980.