Computers and Fluids 264 (2023) 105971

Contents lists available at ScienceDirect

Computers and Fluids

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

Data-driven correction of coarse grid CFD simulations

A. Kiener [*], S. Langer, P. Bekemeyer

German Aerospace Center (DLR), Institute of Aerodynamics and Flow Technology, Center for Computer Applications in AeroSpace Science and Engineering ($C^2A^2S^2E$), Lilienthalplatz 7, 38108 Braunschweig, Germany

ARTICLE INFO

Keywords:
Coarse grid
CFD
Error correction
Graph neural network

ABSTRACT

Computational fluid dynamics is a cornerstone of the modern aerospace industry, providing important insights through aerodynamic analysis while reducing the need for expensive experiments and tests. A consistent spatial discretization on so-called grids approximates the analytical solution of the partial differential equation with increasing number of discrete points. But computing a rigorous amount of CFD simulations on fine grids can be a daunting task due to the high computational cost involved. Thus, it is of interest to find methods which generate accurate results while keeping computational costs low. This work proofs that machine learning as a post-processing tool is capable of improving the accuracy of coarse grid simulations. The results show that three machine learning models with varying complexity, namely random forest, neural network, and graph neural network, are capable of finding patterns in coarse grid simulations. These patterns are used for a vertex-wise prediction of the discretization error to approximate the field variables of interest of the corresponding fine grid simulation mapped onto the coarse grid. Initial training and testing is performed on the two dimensional RAE2822 airfoil leading to corrected flow fields, improved surface integrals and coefficients, even when shocks are present. Additional tests are performed on the RAE5212 airfoil, showing the generalization limits of the trained models. Finally, the training and prediction is showcased on a three dimensional geometry. The proposed method promises to reduce computational expenses while increasing the accuracy of the coarse grid results which works locally, e.g. it corrects the error for each vertex individually and is therefore not restricted by the number of grid points. The presented results obtained by the machine learning models during post-processing are a promising baseline for more integrated developments, where the models will interact in a dynamic fashion with the flow solver to further improve coarse grid simulations.

1. Introduction

Aerodynamic analysis based on computational fluid dynamics (CFD) has become a cornerstone for the aerospace industry, whether it be for analysis or optimization. High-fidelity simulations provide insight into highly complex physical phenomena without the need for expensive wind tunnel experiments or flight tests. Nevertheless, these methods are far from perfect.

One of the crucial aspects when performing such simulations using the Finite Volume Method (FVM) is the need for a spatial discretization, a so-called computational grid to approximate solutions of the governing equations of interest. The design of such grids has in general a significant influence on the accuracy of the approximate solutions. Meshes with a high grid resolution are necessary to accurately capture phenomena such as the effects of turbulence at high Reynolds numbers, where the number of degrees of freedom (i.e. the number of computational points) has to increase close to the boundary layer to correctly approximate steep gradients.

Such a fine-grained resolution significantly increases computational cost as well as simulation time and ultimately becomes a bottleneck for performing a large number of high-fidelity simulations. Lately, due to an increasing availability of hardware and respective algorithms, data-driven methods based on Machine Learning (ML) have become an attractive alternative to be discovered to leverage CFD simulations [1–4]. The field of ML has potential to reduce the computational cost of simulations by taking advantage of data available from testing and simulation.

Previous research tries to circumvent the costly simulation iterations by completely replacing the solver with a Reduced Order Model (ROM) which directly returns the value of interest [5–7]. Hines et al. for example train various regression models on a data set of the XRF1 wing to predict the pressure coefficient based on the Mach number and angle of attack as input [8]. Sabater et al. predict the surface pressure distribution of a two dimensional airfoil case and the three dimensional NASA Common Research Model using Gaussian Processes,

* Corresponding author.
E-mail address: anna.kiener@dlr.de (A. Kiener).

https://doi.org/10.1016/j.compfluid.2023.105971
Received 23 December 2022; Received in revised form 27 April 2023; Accepted 12 June 2023
Available online 15 June 2023

proper orthogonal decomposition and deep neural networks. They have shown that the latter performs best at transonic flow conditions, where shock waves and separation phenomenon are present [9].

Instead of replacing the CFD solver, another approach consists of reconstructing high-fidelity flow fields from their coarse grid counterpart which are cheaper to compute [10–12]. This idea is based on the progress achieved in the field of computer vision, where the super-resolution problem tries to enhance the resolution of under-resolved images [13]. The most common tool for this reconstruction are Convolutional Neural Networks (CNN), which for images has shown the ability to map the input of a low-resolution image to a highly resolved output [14]. Conventional CNNs are readily applicable to structured data which is encountered in images but unfortunately not to unstructured data which is very common for CFD grids [4,15,16]. Thus, this work avoids their use. Beside two more common models (a random forest and a deep neural network), convolutions with graph neural networks are investigated, since they are recently gaining popularity for unstructured CFD data [17–24].

Furthermore, the here proposed approach avoids the reconstruction to higher-dimensional space but focuses on improving directly the accuracy of coarse grid simulations. By using data-driven methods, information from fine grid simulations are embedded into the trained regression models. Following the work of Hanna et al. [25], the ML methods are utilized as a post-processing tool to quantify the discretization error at each grid vertex after the coarse simulations are performed. Such a flexible vertex-wise correction approach does not restrict the models to the grid size seen during training. The main goal of this work is to identify how the error induced by the spatial discretization is quantifiable and subsequently if this error exhibits patterns which can be learned. Traditional approaches aiming to accelerate CFD simulations usually focus on how to speed-up already accurate but costly methods, as it is for example the case for Multigrid methods. This work proposes to reverse the idea by improving the accuracy of computations which are solved for very quickly. Compared to Hanna et al. [25], the proposed method in this work is tested on compressible external flows with transonic conditions and not only the field variables but also integral values are evaluated, which show the benefits of using a nested grid series. Additionally, the work proofs that graph neural networks are a natural choice to represent CFD data and improve the accuracy compared to the other models.

To achieve this, the proposed method is applied to a two dimensional airfoil data set as well as a three dimensional wing data set. For all CFD computations, the Navier–Stokes equations in conjunction with a one-equation turbulence model are solved at varying Mach numbers and angles of attack, including cases with varying shock sizes. The discretization error is corrected for different field variables (velocity, pressure and density). For the training data set, both a fine and a coarse grid simulation are needed for the supervised training approach. During inference, the trained models are able to predict the discretization error based only on a coarse grid simulation. To qualitatively assess the models, the field contours are visualized and for the quantitative assessment, coefficients such as the pressure and lift coefficient are computed from the corrected fields.

The results show that all data-driven models are able to capture the discretization error and improve the test set simulations to a certain extent for both airfoil and wing. On the two dimensional cases, a generalization test is performed using a geometry not seen during training which displays the limits of the proposed data-driven methods. Nevertheless, this work demonstrates the ability of ML models to quantify and return the discretization error of coarse grid CFD simulations with the perspective to improve their accuracy while retaining their computational efficiency.

2. Methodology

2.1. Governing equations

For the following, the notation of Mavripilis, Brandt and Trottenberg [26–28] commonly used for describing the multigrid method is adopted. Thus, different discretizations are described with subscript H and h, which represent the coarse and fine grid spacing, respectively.

Given a domain Ω with a fine grid spacing h, consider a discrete problem $L_h \phi_h = f_h$. An estimate of the solution denoted with a tilde can be obtained via iterative techniques, such that

$$L_h \tilde{\phi}_h = f_h + r_h \tag{1}$$

where r denotes the residual. Correspondingly, the solution on the same domain but with coarse grid spacing H is given as

$$L_H \tilde{\phi}_H = f_H + r_H. \tag{2}$$

The goal of this work is to quantify the discretization error between converged coarse and fine grid solutions and analyze whether this error can be learned by a data-driven modeling approach. To do so, a fine-to-coarse mapping operator $\mathcal{I}_h^H$ is introduced

$$\phi_{\mathcal{I}} = \mathcal{I}_h^H \phi_h \tag{3}$$

which transfers the fine grid solution from Eq. (1) to the coarsely discretized domain. Thus, the discretization error based on the fine grid solution can be defined on the coarse grid as

$$\epsilon_\phi = \phi_{\mathcal{I}} - \phi_H. \tag{4}$$

After being trained, an ML model predicts approximations $\tilde{\epsilon}$ of the discretization error ϵ. With $\tilde{\epsilon}$, the transferred fine grid solution on the coarse domain can be approximated by correcting the corresponding coarse grid solution:

$$\phi_H + \tilde{\epsilon}_\phi = \phi_{\mathcal{I}}. \tag{5}$$

This work intends to improve coarse grid solutions of the Reynolds Averaged Navier–Stokes (RANS) equations with turbulence being modeled using the negative Spalart–Allmaras one-equation model [29].

2.2. Machine learning

Given a set of training instances $\{(x_i, y_i)\}_{i=1}^N$, a supervised learning algorithm for a regression task searches a mapping function $f : X \rightarrow Y$ between the provided feature vector x_i and the output y_i. Here, the output y_i and a training instance i correspond to the discretization error $\epsilon_{\phi,i}$ and one grid point, respectively. Thus, each vertex corresponds to one training instance. The training process is depicted in the upper half of Fig. 1: the training data base is generated by conducting the same simulations on a coarse grid H and a fine grid h. This results in the field solution ϕ_H and ϕ_h of the variables of interest, e.g. in this study velocity in x-, y- and z-direction, pressure and density, such that $\phi = \{u, w, p, \rho\}$ for two and $\phi = \{u, v, w, p, \rho\}$ for three dimensions.

Since it is assumed that the discretization error correlates to steep gradients within a flow solution, the first and second derivatives of the variables of interest in each dimension are a natural choice as features x_i [25]. The gradients are computed using a weighted least squares approach with an inverse distance weight as described in [30]. A local cell Reynolds number defined as $Re_i = \frac{|U_i| d_{w,i}}{\nu_i}$ is another feature which uses the wall distance d_w as reference length.

Since the goal is to find the discretization error for each coarse grid vertex i via supervised learning, ϕ_h cannot directly be used as ground truth for the training. Instead, the variables of interest given on the fine grid h are mapped onto the coarse grid H using the fine-to-coarse mapping operator $\mathcal{I}_h^H$ introduced in Eq. (3). This mapping allows one

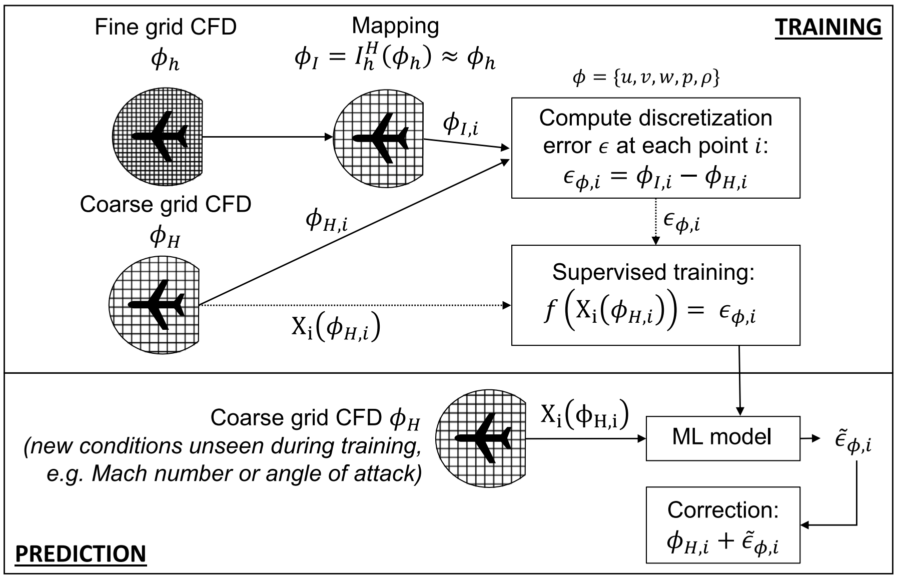

TRAINING
Fine grid CFD
ϕ_h
Mapping
$\phi_I = I_h^H(\phi_h) \approx \phi_h$
$\phi = \{u, v, w, p, \rho\}$
$\phi_{I,i}$
Compute discretization error ϵ at each point i:
$\epsilon_{\phi,i} = \phi_{I,i} - \phi_{H,i}$
Coarse grid CFD
ϕ_H
$\phi_{H,i}$
$\epsilon_{\phi,i}$
Supervised training:
$f\left(X_i(\phi_{H,i})\right) = \epsilon_{\phi,i}$
$X_i(\phi_{H,i})$
Coarse grid CFD ϕ_H
(new conditions unseen during training, e.g. Mach number or angle of attack)
$X_i(\phi_{H,i})$
ML model
$\tilde{\epsilon}_{\phi,i}$
Correction:
$\phi_{H,i} + \tilde{\epsilon}_{\phi,i}$
PREDICTION

Fig. 1. Training and prediction.

Table 1
Overview of features and output.

Input features	# 2D	# 3D	Output	# 2D	# 3D
$\frac{\partial\phi_i}{\partial x_j}$	8	15			
$\frac{\partial^2\phi_i}{\partial x_j \partial x_k}$	16	45	$\tilde{\varepsilon}_{\phi,i}$	4	5
Re_i	1	1			

to quantify the discretization error ε for each variable ϕ at each vertex i of the coarse grid H as

$$\varepsilon_{\phi,i} = \phi_{\mathcal{I},i} - \phi_{H,i} \tag{6}$$

The mapping operation poses the following limitations to the proposed procedure: (a) The choice of the mapping function $\mathcal{I}_h^H$ is non-trivial, especially around curvatures where the vertices of H and h are not overlapping if non-nested grids are used and (b) by reducing the number of data points it naturally decreases the information content and thus the accuracy. Since for the investigated test case the vertices of H are coinciding with their corresponding h vertices, the nearest neighbor interpolation method is a natural choice as a mapping operator $\mathcal{I}_h^H$ to find the coinciding nodes and transfer the values. Thus, the first mentioned limitation can be avoided.

After the training, the ML model is able to predict for previously unseen values of features x_i an approximation $\tilde{\varepsilon}_{\phi,i}$ of the true output variable $\varepsilon_{\phi,i}$, such that $f(x_i) = \tilde{\varepsilon}_{\phi,i} \approx \varepsilon_{\phi,i}$, as given in the lower half of Fig. 1. Table 1 summarizes the described feature inputs and outputs: The model takes as input for each cell i the first and second derivative of the variables of interest and returns the respective discretization error for each variable. The table also reports the number of inputs and outputs for a 2D and a 3D case scenario, since discarding the velocity in one direction leads to a reduced number of outputs and derivatives.

After the prediction of the discretization error $\tilde{\varepsilon}_{\phi,i}$ by the ML model, this value is used to correct the flow field variables on the coarse grid H in a vertex-wise manner as:

$$\phi_{H,i} + \tilde{\varepsilon}_{\phi,i} = \tilde{\phi}_{\mathcal{I},i} \approx \phi_{\mathcal{I},i} \tag{7}$$

To learn the function $f(x_i) = \tilde{\varepsilon}_{\phi,i}$ connecting the coarse grid features with the discretization error, three models with varying complexity are used and compared, namely a random forest, a feed forward neural network and a graph neural network. The models and the training and validation strategy are described in the following.

Random Forest One of the three models employed in this work is the Random Forest (RF) algorithm developed by Breiman [31]. It belongs to the class of ensemble algorithms. Ensemble models seek to improve the predictive performance by combining several weak learners. For a RF regression model, the results of many decision trees for a regression task are averaged to arrive at the final prediction. Let the prediction of a single decision tree (DT) be $f_{\mathrm{DT}}(x) = \tilde{y}_{\mathrm{DT}}$, then the RF prediction comprising N decision trees is

$$\tilde{y} = \frac{1}{N}\sum_{i=1}^{N} \tilde{y}_{\mathrm{DT}}. \tag{8}$$

Neural Network A fully connected deep Neural Network (NN) is implemented as the second model to be investigated. A NN consists of multiple layers of so-called neurons. Each layer can be described as a regression function $f_l(x) = \tilde{y}_l$, where x is the input to each layer l and $\tilde{y}$ the layers respective output. Each regression layer is given by

$$f_l(x) = g_l(W_l x + b_l) \tag{9}$$

where g_l denotes the activation function introducing a non-linearity to the equation. The weight matrix W_l and the bias vector b_l are the parameters to be optimized during the training phase. After training, the prediction $\tilde{y}$ is obtained via a nested function. Thus, each regression layer is applied successively such that a NN with for example three layers is written as

$$\tilde{y} = f_3(f_1(f_0(x))) \tag{10}$$

Graph Neural Network As final model, a Graph Neural Network (GNN) is employed. This model has been proposed as an extension of NNs for graph structured or non-Euclidean data [32]. Its advantage lies in its applicability to graphs $\mathcal{G}$ of arbitrary structure and its permutational invariance to the number of input nodes $\mathcal{V}$ and edges $\mathcal{E}$. GNNs take into account information of neighboring data points as well as the connectivity structure between all data points, thus leveraging the usable information content of a given data set. The function to describe a layer is written as

$$H_{l+1} = f(H_l, A) \tag{11}$$

where the input features correspond to the first layer $H_{l=0}$ and A denotes the adjacency matrix of the graph, which is needed to describe the graph's connectivity. Three operations are needed to update each

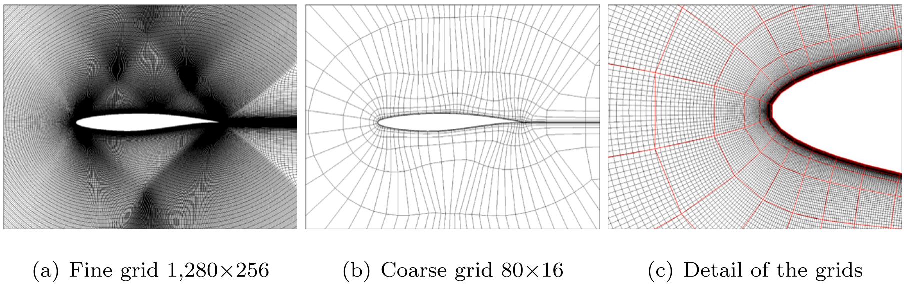


Fig. 2. Coarse and fine grid, RAE2822.

node: so called message passing, aggregation and the update. The first two steps, message passing and aggregation, can be more intuitively described as neighborhood aggregation, which is the process of embedding into each node the information of a certain amount of its respective neighboring nodes. Different layer implementations differ in their definition of the function f on the right hand side of Eq. (11). Denoting the node on which an operation is computed with superscript i and its neighbors with j, one layer on this node can be rewritten as:

$$H^i_{l+1} = g_l(W_1 H^i_l + \sum_j W_2 H^j_l) \tag{12}$$

As for NNs, g_l denotes a non-linear function and W_1 and W_2 the weight matrices. Eq. (12) shows an example where a sum is applied to aggregate the information of the neighboring nodes j, another popular variant of an aggregation function is to compute the mean. The final prediction using two layers (where each layer corresponds to one hop into the neighborhood to agglomerate information), can be written as another nested function similarly as for the NN:

$$\tilde{y} = f_1(f_0(H_{l=0}, A)) \tag{13}$$

For this work the node-level problem is solved, e.g. a regression task at each node $\mathcal{V}$ of the given graph $\mathcal{G}$. Furthermore, the graph convolutional operator proposed by Kipf et al. [33] is employed.

Validation and Hyperparameter Optimization All three models are trained using the mean squared error as loss function. To assess the quality of the models, another metric is evaluated on the set aside validation data during cross-validation. In this study, the coefficient of determination R^2 is used as validation metric:

$$R^2(\tilde{y}_i) = 1 - \frac{\sum_{i=1}^N (y_i - \tilde{y}_i)^2}{\sum_{i=1}^N (y_i - \bar{y})^2} \tag{14}$$

Here, $\bar{y}$ denotes the mean of the set of output variables $\{y_i\}_{i=1}^N$. A model predicting the exact output y will reach a coefficient of determination $R^2 = 1$, whereas $R^2 = 0$ corresponds to a model predicting the average $\bar{y}$ for any given instance.

During the tuning of the hyperparameters k of the models, the target is to maximize the R^2 score. These hyperparameters k can be optimized with respect to its performance on the validation data set evaluated for an objective function $J(k)$:

$$k_{\text{opt}} = \arg\max_{h\in\mathcal{K}} J(k) = \arg\max_{k\in\mathcal{K}} R^2(k) \tag{15}$$

Beside tedious manual hyperparameter tuning, common strategies for hyperparameter optimization are grid and random search, in which all possible combinations or a random set of combinations are explored for a given set of hyperparameters. A more effective strategy is Bayesian optimization, where a probabilistic surrogate model explores only the promising solution space based on priors. The subsequent hyperparameters are chosen in an informed manner, by taking into account already explored trials. In this work a sampler using a Tree-structured Parzen Estimator (TPE) algorithm [34] is used.

For the RF model, the number of decision tree estimators and the maximum number of features for a split are tuned. The hyperparameters to be tuned for the NN are the number of hidden layers, the number of neurons, the batch size, the optimizer, activation function and the learning rate schedule. The best optimizer and activation function found during the NN tuning were then employed for the GNN to reduce the hyperparameter search space. Only the number of stacked convolutional layers, their feature dimensionality as well as the learning rate schedule are tuned for the GNN. An overview of the search space used during the tuning and the final hyperparameters is given in the Appendix.

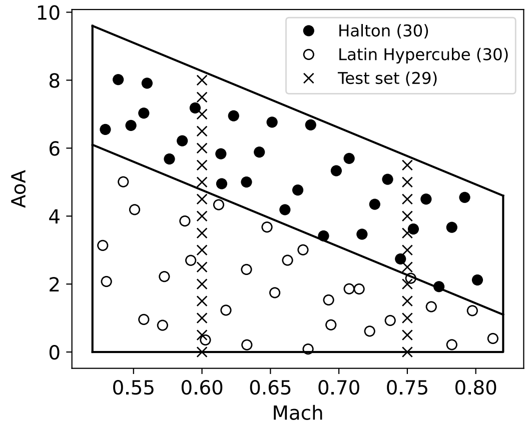


Fig. 3. Sampling points for train and test sets, RAE2822.

3. 2D application case

3.1. RAE2822 simulation

This section presents results for the turbulent flow over the two-dimensional transonic airfoil RAE2822. The DLR-CFD simulation software TAU [35] is used, which employs the FVM and a vertex-centered scheme. The fine grid and coarse grid solutions are obtained on structured C-type grids with $1,280 \times 256 = 327,680$ and $80 \times 16 = 1,280$ cells, respectively (see Figs. 2(a) and 2(b)). Hence, the resolution factor between both grids is 256. The two grids are nested grids, as can be seen in Fig. 2(c). A detailed study on this grid sequence was conducted by Langer [36].

Steady-state simulations are performed for Mach numbers between 0.52 and 0.82 and angles of attack between 0.0° and 9.0°. These simulation points are chosen by a design of experiment with a Latin Hypercube and a Halton sampling. In total, 60 simulations were conducted for the test and validation set, as given in Fig. 3, with half of them located in the linear and the other half in the non-linear regime. Additional 17 and 12 simulations are conducted at Mach numbers 0.6 and 0.75 for the test set.

To retrieve the complete data base, for each operational point a coarse and fine grid simulation is computed. For all the coarse and the fine grid computations, the convergence criterion is reached, meaning that a reduction of the density residual of 12 orders of magnitude

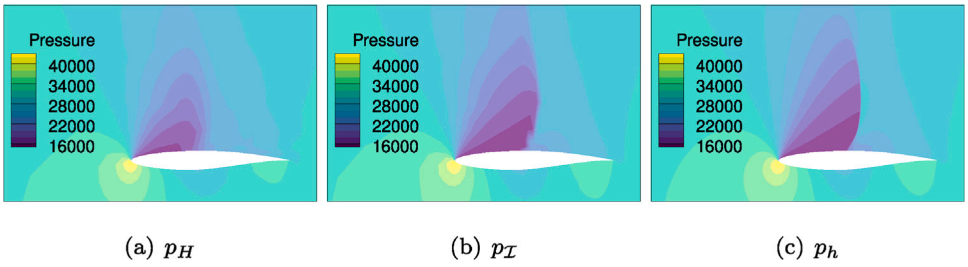


Fig. 4. Pressure field ($M = 0.75$, $AoA = 5.5°$), RAE2822.

Table 2
Wall clock time, RAE2822.

		Coarse	Fine
M [–]	AoA [°]	Time	Time
0.6	4.0	2 min 11 s	3 h 53 min
0.6	8.0	1 min 15 s	4 h 31 min
0.75	3.0	1 min 42 s	8 h 54 min
0.75	5.5	1 min 13 s	10 h 54 min

is obtained. The fine grid simulations are all conducted on a cluster, using one node with eight processes. The coarse grid equivalents are feasible on a desktop computer. The respective computational time for a selection of simulations of the test set is given in Table 2. These times indicate that using only the coarse grid with subsequent correction would greatly lower the required computational time, since the coarse grid simulations converge very quickly while the fine grid equivalents need several hours.

3.2. Mapping operator and correction term

As previously mentioned, the vertices of the coarse grid H coincide with their corresponding fine grid vertices, which makes the nearest neighbor interpolation method a natural choice as a mapping operator $\mathcal{I}_h^H$ to find the coinciding nodes and transfer the values of the fine grid solution to the coarse grid. To assess this mapping operation, different comparisons are presented in the following.

Fig. 4 depicts the field solution of the pressure variable at Mach number 0.75 and angle of attack 5.5°. Looking at the coarse grid simulation p_H in Fig. 4(a) and comparing it with the fine grid simulation p_h in Fig. 4(c), it is visible that the former does not compute the shock accurately, which is much greater in the fine grid solution. The second comparison is done by looking at the interpolation $p_{\mathcal{I}}$ of the fine grid solution onto the coarse grid given in Fig. 4(b). The low amount of degrees of freedom deteriorates the fine grid solution p_h after the mapping and the contour plot is not as smooth as on the fine grid. It has to be noted that the post-processing software has to interpolate between the coarse grid data points for Figs. 4(a) and 4(b) to visualize a contour plot, which results in a field appearing not to be smooth. Nevertheless, an enlarged shock strength and low pressure field is now present compared to p_H and an improvement is clearly visible.

Figs. 5 and 6 show different comparisons of coefficients retrieved from ϕ_H, ϕ_h as well as $\phi_{\mathcal{I}}$. It has to be emphasized that only the field variables of velocity, pressure and density ($\phi = \{u, w, p, \rho\}$) are mapped. Any other integral coefficients presented hereafter are retrieved from the field solutions. This allows to not only judge the resulting field solutions qualitatively with contour plots but also quantitatively by comparing for example the lift coefficient.

Looking at the friction coefficient in Fig. 5(a) and the viscous drag coefficient in Fig. 6(b) resulting from the mapping $\mathcal{I}_h^H$, it can be observed that the coarse boundary layer cells are not resolved enough to capture the same velocity gradient as the corresponding fine grid solution. Hence, the friction coefficient and the viscous drag coefficient are in general lower on the coarse grid H after the mapping operation. Nonetheless, it is assumed for this study that the coarse grid has enough degrees of freedom to capture the relevant physical phenomena of the fine grid solution after the mapping. Therefore, the mapped solution $\phi_{\mathcal{I}}$ is taken as ground truth for the ML model training and evaluation.

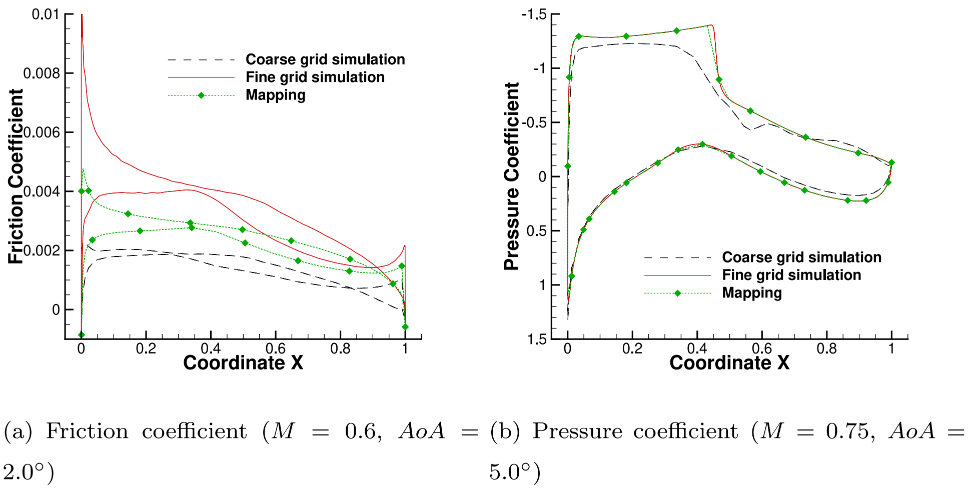


Fig. 5. Comparison of surface coefficients on H and h, RAE2822.

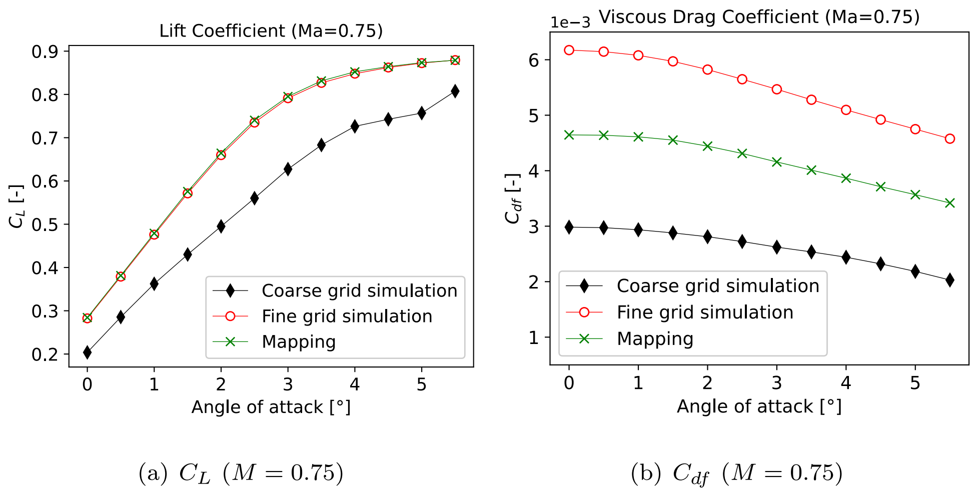


Fig. 6. Integral coefficients on H and h, RAE2822.

3.3. Training and validation strategy

For all three ML models, a four-fold cross validation is performed. Therefore, during each fold, the models are trained on 45 simulations and validated on the remaining 15. The 39 simulations at Mach numbers 0.6 and 0.75 are then tested only on the best performing models resulting from this cross validation procedure. For the implementation of the RF the Scikit-learn library [37] and for the NN model the SMARTy library [38] with the PyTorch backend [39] are used. For the GNN model with the GCN layers, again SMARTy is used which works as a wrapper for the pytorch geometric library [40]. For hyperparameter tuning, the optuna toolbox is used [41] with a TPE sampler and a median pruning strategy. As data pre-processing, standardization is applied to each feature as well as to each output variable independently (i.e. shift to zero-mean and unit variance).

The final hyperparameters for the RF are 236 estimators with 12 features to consider for a split. For the NN, the number of layers is set to 7 with 317 neurons each and the initial learning rate is 3.24×10^{-4}. The batch size shows no significant influence in the model performance and is finally set to 1,024. For both the NN and GNN, the learning rate decays to a fifth of its initial value after 1,000 epochs and a stopping criterion based on the validation data is employed. During the tuning of the NN, the Adam optimizer performs better than Stochastic Gradient Descent and also the hyperbolic tangent outperforms the rectified linear unit function. Finally, for the GNN, the tuning converges to nine layers with 285 feature dimensions for each layer and an initial learning rate of 1.62×10^{-3}. More details on the hyperparameter search space are given in the Appendix.

All described ML related methods are performed either directly on CPU (RF) or on a GPU (NN, GNN). The GPU used is a NVIDIA Quadro P2200 with 5 GB memory. The training times on the two dimensional case revolve around 20 min (RF), 1.7 h (NN) and 2.2 h (GNN). The deployment of the models during the prediction phase, e.g. retrieving the corrections, takes only seconds.

3.4. Test case results

This section presents results for the test set of the RAE2822 at Mach numbers 0.6 and 0.75. Since the correction of the flow field is the main focus of the proposed method, firstly a qualitative comparison of the pressure field is given in Fig. 7 for Mach number 0.6 at angle of attack 7.5°. Looking at the ground truth (e.g. the mapped pressure $p_{\mathcal{I}} = \mathcal{I}_h^H p_h$) in Fig. 7(b), it is visible that the flow field is not smoothly captured by the coarse grid H after the mapping, but it rather shows hard edges along the contours which is due to the very low amount of degrees of freedom. Nevertheless, the mapping $\mathcal{I}_h^H$ to the coarse grid is able to capture the shock resolved on the fine grid, whereas the stand-alone simulation on the coarse grid does not feature such an extensive shock (see Fig. 7(a)). This is reflected in the percentage error given in Fig. 7(d), which is defined as the difference between coarse grid simulation given in Fig. 7(a) and its mapping equivalent in Fig. 7(b). The figure shows that the error is mainly present around and upstream of the shock location. Figs. 7(e)–7(g) show the reduced error after the correction of the ML models. All three models are able to reduce the error between the leading edge and the shock location, thus predicting a shock where the coarse grid simulation fails to do so. All models succeed in reducing the error, with the performance of the GNN being the best. For the RF model the maximum error is found at 85.6% within the boundary layer, for the NN it is found outside the boundary layer but with an amount of 103.1% . The GNN model decreases the error to a maximum of 48.1% and the resulting pressure field shows the smoothest distribution as presented in Fig. 7(c), this might be explained by the GNN taking into account neighboring information of each corrected vertex value. Similar field correction trends are observed for the other variables and flow conditions.

In the following, different integral coefficients are presented and compared to quantitatively assess the results. Again it is emphasized that all coefficients are derived from either the simulated, mapped or ML corrected field solutions. In Fig. 8, the friction and pressure coefficient along the airfoil surface are visible for the coarse grid simulation to be corrected, the ground truth and the ML corrections for Mach number 0.75 at angle of attack 0.0° and Mach number 0.6 at angle of attack 7.5°, respectively. Fig. 8(a) presents the case with low angle of attack where all models performed well, but it is visible that the RF shows outliers (e.g. on the upper surface around $x = 0.8$). Fig. 8(b) shows the case at high angle of attack. Again, the GNN model predicts the smoothest distribution, while the NN shows outliers or oscillating behavior especially in close proximity to the shock location comparable to the RF. This smooth behavior of the GNN along the surface can be explained by the additional regularization through incorporating information from neighboring nodes. Lift, pressure and viscous drag coefficients are given in Fig. 9. All models improve the accuracy significantly. At higher angles of attack, a reduction of the prediction accuracy occurs due to physical non-linearities with the RF model showing the noisiest behavior. Still, all models improve the coefficients with the GNN being the best performing one.

A final comparison on the test set RAE2822 between the models is done by computing the coefficient of determination R^2, given

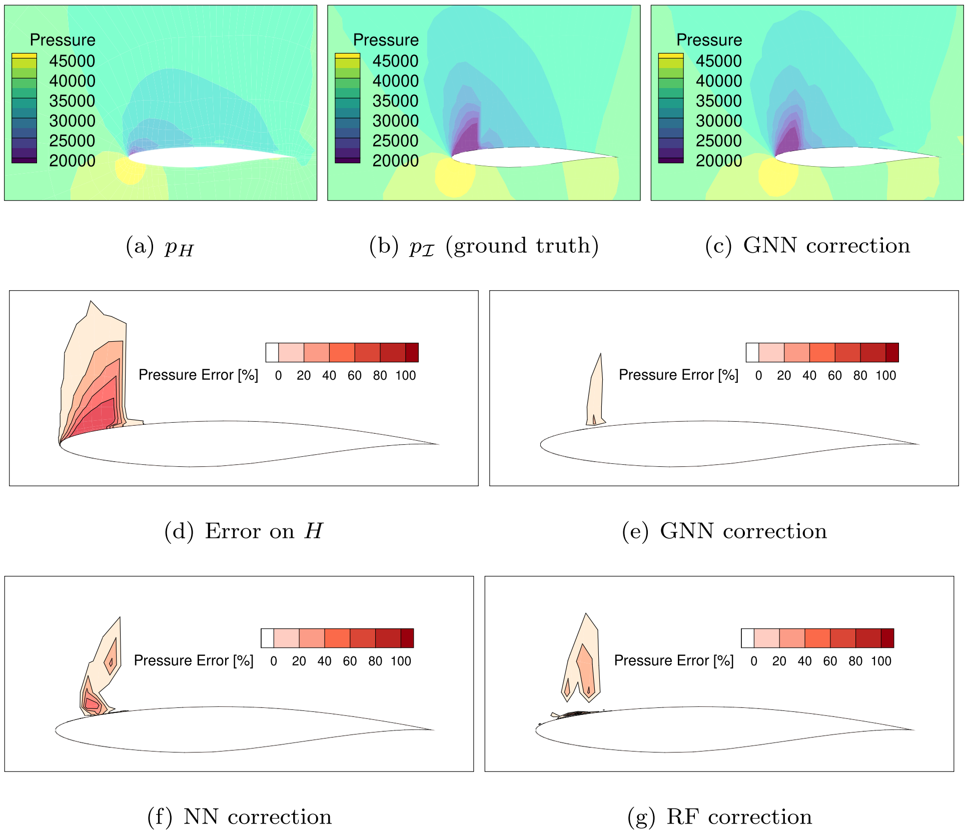


Fig. 7. Pressure field and error ($M = 0.6$, $AoA = 7.5°$), RAE2822.

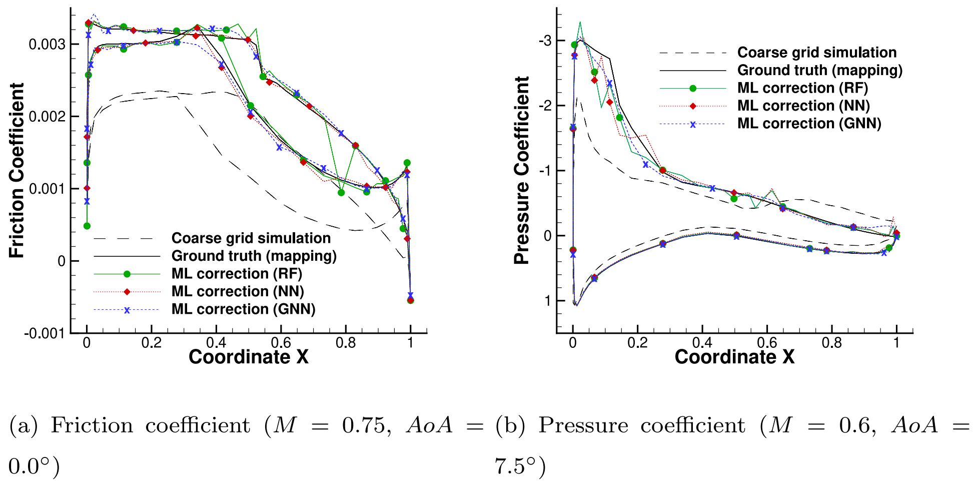


Fig. 8. Corrected surface coefficients, RAE2822.

in Eq. (14), which allows to assess quantitatively the prediction of the discretization error $\tilde{\varepsilon}$. Fig. 10 contains the scores averaged over all simulations within each test set and indicates that the GNN is the best performing model. Considering the test set at lower Mach number 0.6 on the left hand side of Fig. 10: the NN and the GNN show a consistent performance with an R^2 value between 0.9 and 1 across all corrected field variables, whereas the RF shows a drop of performance for the pressure and density error prediction. On the right hand side of Fig. 10, the results for the test set at Mach number 0.75 are depicted. All models show a drop in performance for the error prediction of the pressure and density field. Again, the RF model scores lowest whereas the GNN shows for all predictions a relatively high score above 0.84. The test set results of the NN and GNN were observed to be in similar ranges as the R^2 values of the cross validation. Only the RF showed a significant drop in performance for the high Mach number test set, indicating an overfit.

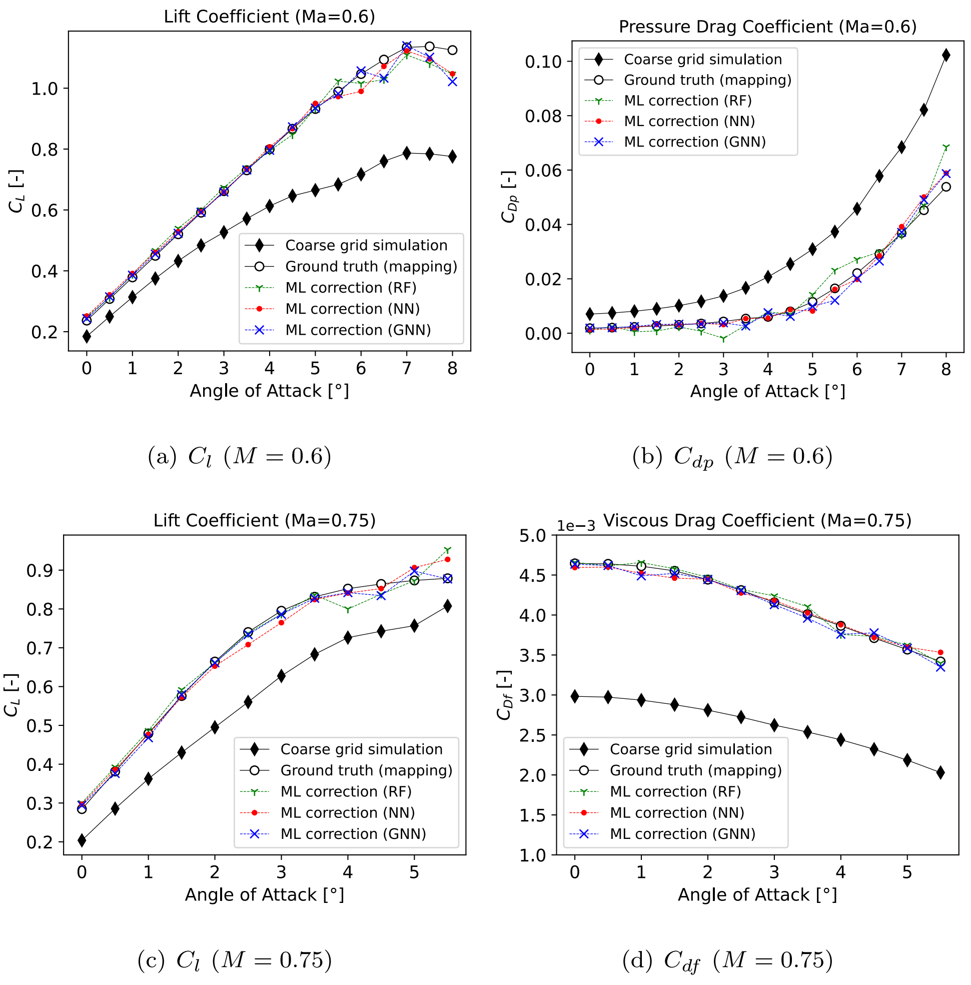


Fig. 9. Corrected integral coefficients, RAE2822.

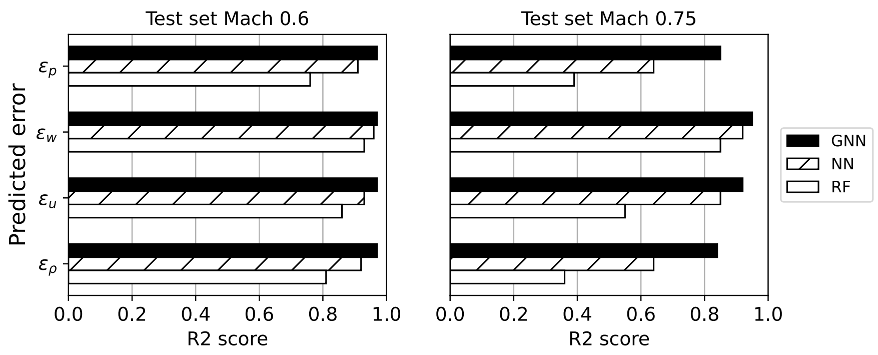


Fig. 10. Coefficient of determination R^2 ($M = 0.6$ and $M = 0.75$), RAE2822.

3.5. Generalization test case

As a test case to assess the generalization capabilities of the trained models, additional simulations are carried out on a different transonic airfoil geometry. For this, mesh deformation is employed based on the fine grid h of the RAE2822 with a subsequent agglomeration to obtain the coarse grid H. The similar airfoil shape RAE5212 is chosen, as given in Fig. 11. Solutions are computed for Mach number 0.6.

The GNN model trained on the RAE2822 geometry as described above is directly employed for the prediction on the RAE5212 data. It is clear that training on a variety of geometries will lead to improved predictions. Nevertheless, the intention is to show the capabilities and limitations of the proposed method on new settings, this time slightly

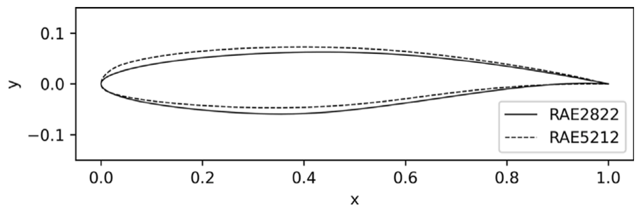


Fig. 11. RAE2822 and RAE5212 airfoils for this generalization study.

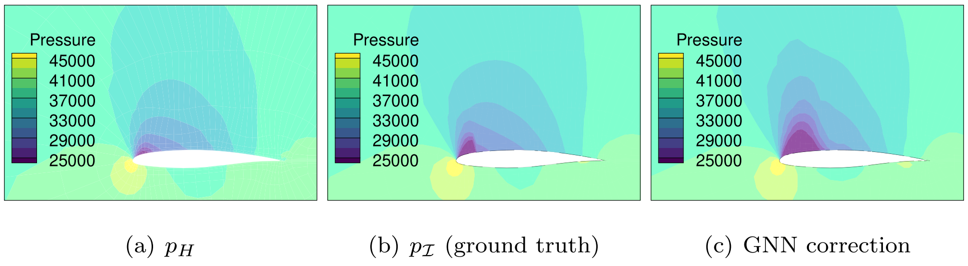


Fig. 12. Pressure field ($M = 0.6$, $AoA = 4.0°$), RAE5212.

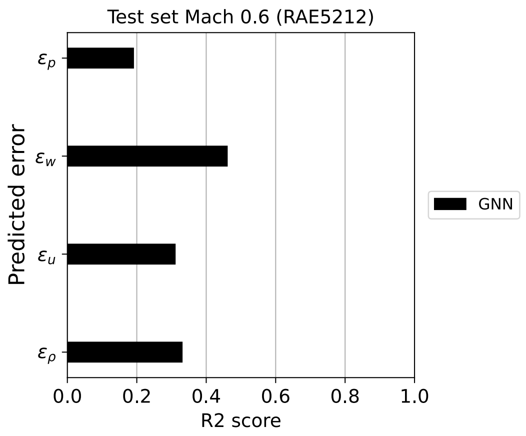


Fig. 13. R^2 score ($M = 0.6$), RAE5212.

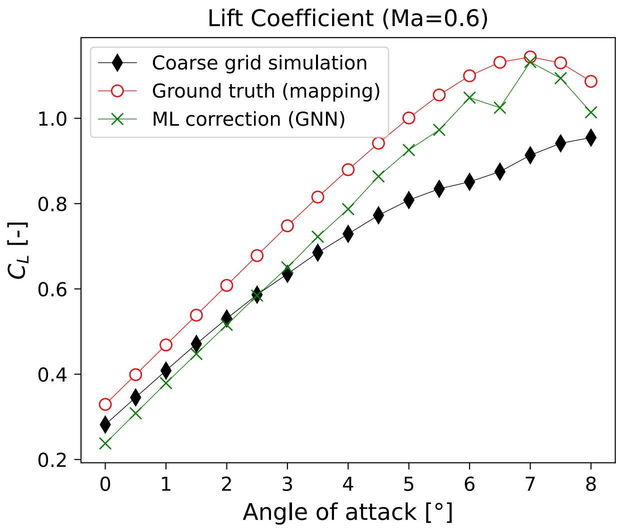


Fig. 14. C_l ($M = 0.6$), RAE5212..

varying the geometry instead of only the boundary conditions with Mach number and angle of attack. Having a look at the corrected pressure field for angle of attack 4.0° in Fig. 12(c) shows that the model tends to over-correct the pressure field, resulting in an enlarged shock on the upper surface as well as a higher pressure area on the lower surface shortly before the trailing edge. This trend is observed for all tested angles of attack. In general, the model shows low scores for the R^2 value across all angles of attack presented in Fig. 13. The lift coefficients derived from the corrected fields are shown in Fig. 14, which indicates improved values along the surface and boundary layer vertices for higher angles of attack. Nevertheless, the results are not comparable to the test set of the RAE2822, especially at lower angles of attack where the coefficients are worsened.

Table 3
Wall clock time, LANN.

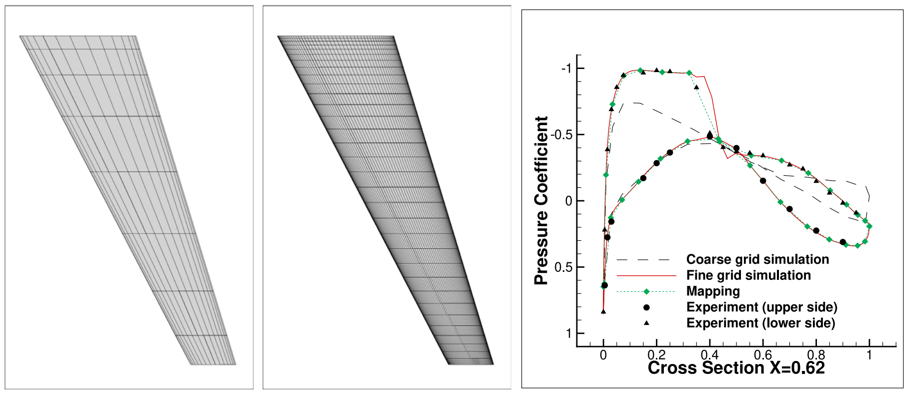
		Coarse	Fine
M [–]	AoA [°]	Time	Time
0.6	0.0	6 min 29 s	4 h 34 min
0.6	6.5	8 min 6 s	6 h 5 min
0.75	0.0	6 min 32 s	5 h 9 min
0.75	5.0	9 min 37 s	5 h 7 min

4. 3D application case - LANN wing

As a final application test, the proposed method is applied to a three dimensional geometry, namely the LANN wing. Again, coarse and fine grids are retrieved from a structured and nested grid series [42]. The grids contain 7,040 and 450,560 elements, respectively, resulting in a resolution factor of 64. The grids are shown in Figs. 15(a) and 15(b). Simulations on both grids are conducted at operational point with Mach number 0.82 and angle of attack 0.6°. Fig. 15(c) compares the results of the pressure coefficient along the cross section at $X = 0.62$ to measurement data from [43] to confirm the simulation settings. Additionally, Fig. 15(c) shows the C_p of the mapped solution $\phi_{\mathcal{I}} = \mathcal{I}_h^H \phi_h$. As for the test case presented before, the nearest neighbor interpolation is employed as mapping operator. Comparing for example the fine grid pressure coefficients with the mapped ones in Fig. 15(c) implies that the coarse grid has sufficient degrees of freedom along the wing surface and the boundary layer to capture the coefficients accurately after the mapping.

In total 40 simulations are computed between Mach number 0.52 and 0.82 and angle of attack 0.0° and 7.5° to obtain training data. As before, two design of experiments are carried out and combined: one for 20 data points within the linear regime and one for 20 data points within the non-linear regime. This is done with the aim to obtain a denser distribution within the more challenging flow regime, as can be seen in Fig. 16. For testing 14 points at Mach number 0.6 and 11 points at Mach number 0.75 are selected.

As for the two dimensional data set, for each operational point a coarse and fine grid simulation is computed with convergence criterion being set to reduce the density residual by 12 orders of magnitude. The fine grid simulations are all conducted on a cluster, using one node with eight processes. The coarse grid equivalents are decomposed into four domains and computed on a desktop computer. The respective computational time for a selection of simulations of the test set is reported in Table 3. As before, these wall clock times indicate the enormous acceleration which can be achieved when relying on corrected coarse grid simulations.



(a) Coarse grid (b) Fine grid (c) C_p ($M = 0.82$, $AoA = 0.6°$)

Fig. 15. Coarse and fine surface grid and measurement data comparison, LANN.

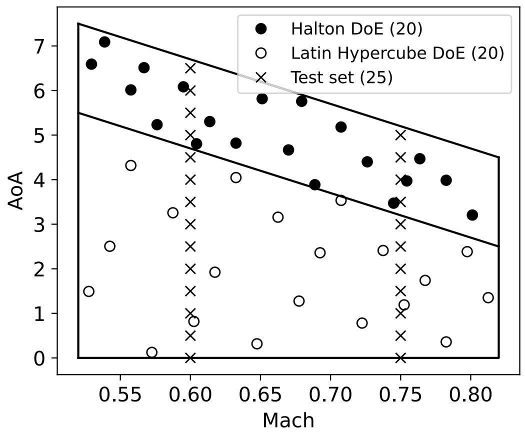


Fig. 16. Sampling points for train and test sets, LANN.

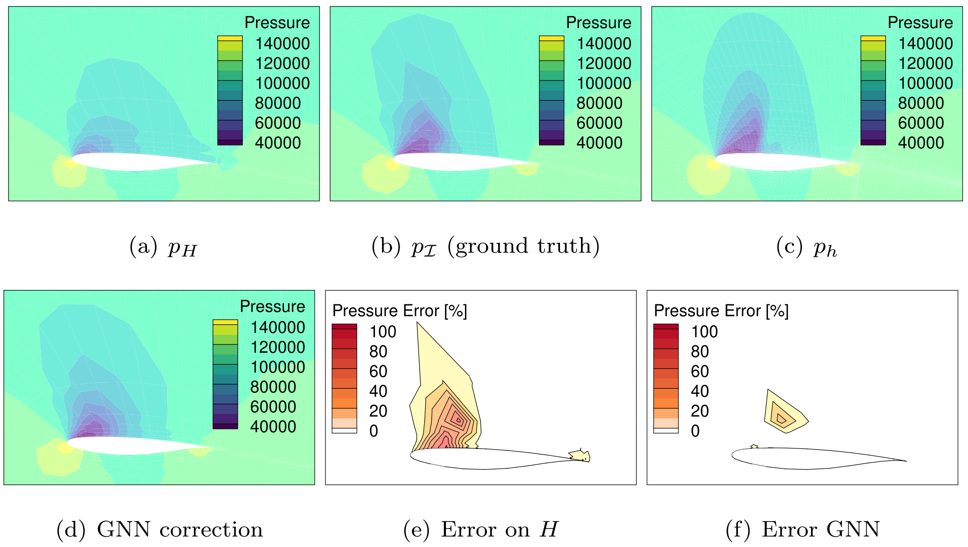
4.1. Training strategy

For the regression task, only the previously best performing model on the RAE2822 is used, i.e. the GNN. The same features are used as inputs, including the gradients in the third dimension and again, standardization is applied to features and the output variables individually. Instead of conducting a rigorous cross validation, the best hyperparameters from the previous training on the RAE2822 airfoil are directly employed, e.g. nine layers with 285 feature dimensions each and an initial learning rate of 1.62×10^{-3} (see also Appendix for more details on the hyperparameters). Only the number of epochs is different, since a stopping criterion based on evaluating the current training loss is employed and leads to higher number of epochs until convergence.

Compared to the 2.2 h training time on the two dimensional RAE2822, the training for the LANN data set took 9.6 h on the same hardware. This can be put into relation with the amount of training instances: 60 RAE2822 simulations in total with each 1,280 data points sums up to 76,800 training instances for the two dimensional training case, whereas the LANN data base consists of 40 simulations with 7,040 data points, resulting in 281,600 training instances.

4.2. Test case results

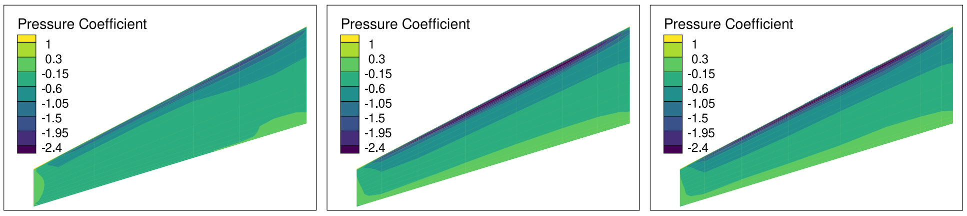
The first results presented are the contour plots of the pressure field at Mach number 0.75 and angle of attack 4.5° in Fig. 17, showing a cross section at $X = 0.25$. Comparing the coarse and fine solution in Figs. 17(a) and 17(c), it is visible that the coarse grid is not capable of resolving the shock intensity properly. As for the ground truth



(a) p_H (b) $p_{\mathcal{I}}$ (ground truth) (c) p_h

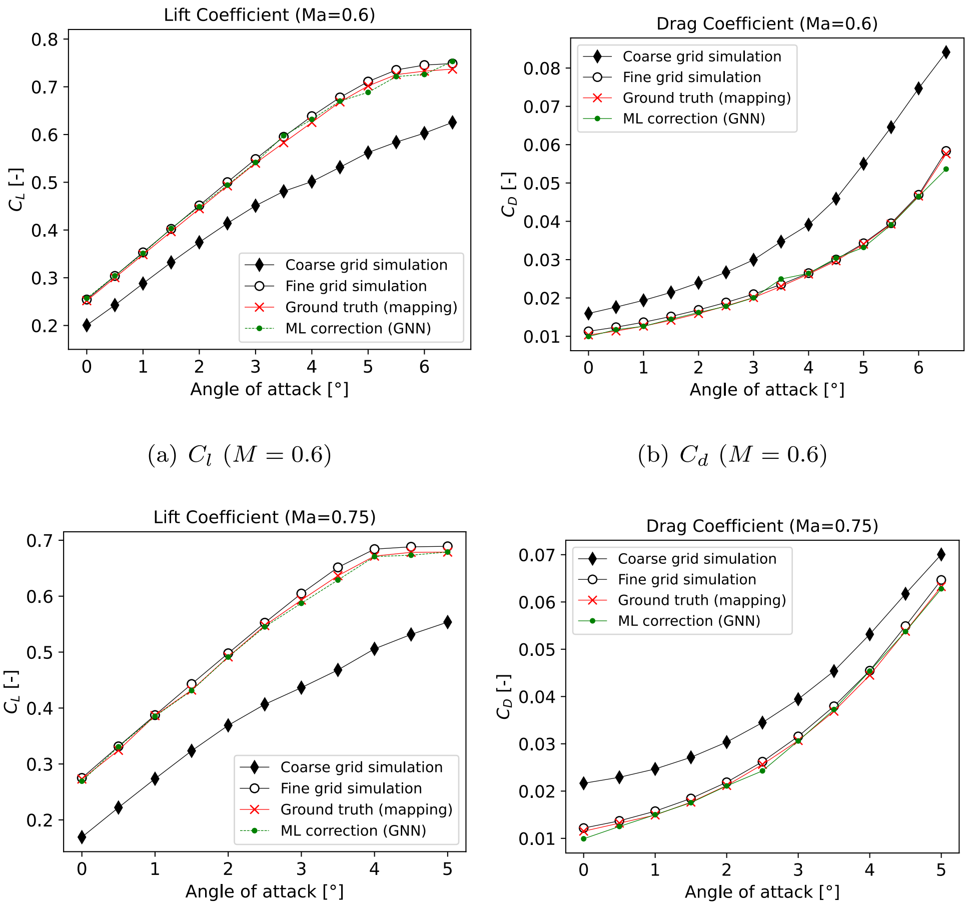
(d) GNN correction (e) Error on H (f) Error GNN

Fig. 17. Pressure field and error at slice $X = 0.25$ ($M = 0.75$, $AoA = 4.5°$), LANN.



(a) $C_{p,H}$ (b) $C_{p,\mathcal{I}}$ (ground truth) (c) ML correction

Fig. 18. Cp upper side (M = 0.6, AoA = 5.0°), LANN.



(a) C_l $(M = 0.6)$ (b) C_d $(M = 0.6)$ (c) C_l $(M = 0.75)$ (d) C_d $(M = 0.75)$

Fig. 19. C_l and C_d, LANN.

used to train the model, its quality can be assessed by comparing the pressure field of the mapping operation $p_{\mathcal{I}} = \mathcal{I}_h^H p_h$ given in Fig. 17(b) with the corresponding coarse and fine grid pressure field p_H and p_h. The mapped pressure field is not capable of smoothly capturing the fine solution p_h due to the low amount of degrees of freedom on the coarse grid, which can be seen by comparing Fig. 17(b) with Fig. 17(c). Nevertheless, the mapped solution captures the pressure on the upper wing surface shortly after the leading edge and on the lower surface right before the trailing edge better than the simulation on the coarse grid in Fig. 17(a). Fig. 17(e) depicts the percentage of error of the coarse grid solution compared to the mapped solution. The error is mainly located around the shock area and additionally around the trailing edge. Finally, Fig. 17(f) shows the percentage error after the GNN correction is applied. The result of the correction is given in Fig. 17(d), proving that the GNN can approximate the ground truth.

The performance of the GNN is also reflected in the accuracy of the corrected pressure coefficient, given in Fig. 18 for the upper wing surface. The ML correction on the wing surface given in Fig. 18(c) almost perfectly matches the distribution of the ground truth in Fig. 18(b). The same trend is observed for both test sets at Mach numbers 0.6 and 0.75 if the integrated coefficients of lift and drag are evaluated, given in Fig. 19. These plots also show the coefficients of the corresponding fine grid solution.

Finally, Fig. 20 presents the coefficient of determination averaged over all angles of attack for the predicted errors. The GNN scores a value of $R^2 = 0.89$ or higher for both test sets.

Furthermore, it can be noted that since the training is performed on the coarse grid, no additional arrangements have to be taken into account to fit the three dimensional data into memory. Nevertheless, the training time of the GNN revolved around 10 h. But as before, the time needed for a prediction takes only a few seconds.

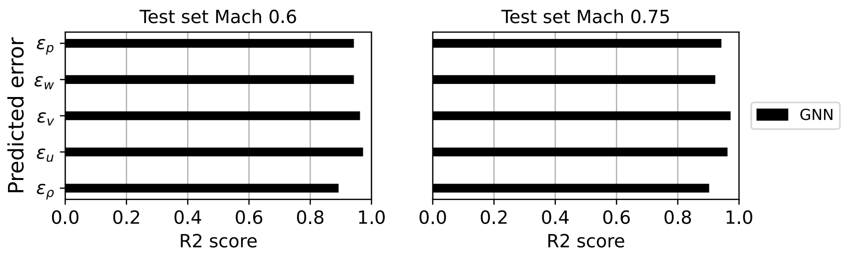


Fig. 20. Coefficient of determination R^2 ($M = 0.6$ and $M = 0.75$), LANN.

5. Conclusion

Conducting a tremendous amount of CFD simulations on fine grids can be a daunting task due to their high computational costs. Thus, it is of interest to find methods which yield accurate results while keeping computational costs low. This work presents a procedure to quantify the discretization error between solutions of two nested grids. Subsequently to training regression models on this error, they can be employed to predict the error if only the coarse grid solution is available. This error is then used to correct the coarse grid field variables to approximate the corresponding fine grid simulation mapped onto the coarse grid.

This work presents two different test cases, one in two and one in three dimensions. First, a data base of corresponding coarse and fine grid solutions is computed for the RAE2822 airfoil. After using a mapping function to transfer the fine grid field variables to the coarse grid, the discretization error can be quantified at each coarse grid vertex for each variable to be corrected. With a supervised learning approach using a Random Forest (RF), a Neural Network (NN) and a Graph Neural Network (GNN), this error can be reproduced vertex-wise by learning a function which takes as input several features of the coarse grid. Secondly, the same procedure is repeated with the best model on the LANN wing data set.

Good results are achieved on the test set for the RAE2822 airfoil covering varying Mach numbers and angles of attack. Both the RF and NN models show noisy behavior in their corrections, whereas the GNN is capable of returning a smooth flow field. The overall better performance of the GNN model underlines its capability of grasping more information by taking into account the connectivity to neighboring nodes and their features. Qualitatively, the field corrections are analyzed and a quantitative analysis is done by deriving various surface and integral coefficients. Difficulties are encountered in the non-linear flow regime at high angles of attack where shocks are dominant.

Testing the models trained on the RAE2822 on the different airfoil geometry RAE5212 obtained by mesh deformation shows generalization limits, since the trained models return excessive corrections, such as enlarged shocks. It is expected that extending the training data base with more diverse cases (e.g. varying geometries) enlarges the space in which the models return valid results.

As for the three dimensional case of the LANN wing, the results show significant improvements of the coarse grid accuracy by employing the GNN. Again, flow fields, surface and integral coefficients are compared to assess the models quality. The achieved results without retuning the hyperparameters found for the RAE2822 airfoil and reusing them for the LANN wing proof that GNNs are a robust choice.

Often, current work found in literature aiming to reduce the error of coarse grid CFD simulations resort to the use of convolutional neural networks (CNN), which are commonly used for images. The results of this work have shown that GNNs are a natural choice for CFD data and are thus a promising alternative to CNNs. Additionally, the findings indicate that the discretization error with respect to a fine grid simulation can be quantified, learned and finally corrected, even for complex 2D and 3D external flows with transonic conditions. This indicates that the proposed method to improve coarse grid simulations is a viable alternative to accelerating already accurate simulations.

Generally, a limitation of the method is the chosen mapping function, which naturally decreases the data quality of the fine grid. Nevertheless, this work shows that ML models can find patterns in coarse grid CFD simulations and therefore improve them. The requirements met in this work to achieve these results are vertex-centered data structures as well as coarse grids which still capture the physics of interest. Further studies could focus on investigating mapping functions for cases in which these requirements are not met, e.g. for cell-centered data structures and unstructured grids. For the future, it is of interest to explore an extension of the proposed method to tackle the super-resolution problem, which closely resembles the so-called multigrid technique already known within the CFD community. In this context, it might be worthwhile to learn certain building blocks of the multigrid method, such as the two interpolation operators itself, namely the restriction and the prolongation. This would allow to couple traditional approaches with data-driven methods instead of completely substituting one with the other. For the immediate future, it is planned to develop a more integrated methodology by closely coupling an ML model with the CFD solver to correct coarse grid CFD simulations. This should enable improvements for correcting the discretization error beyond a pure post-processing approach, since a continuous exchange between solver and ML model allows the former to regularize the influence of the data-driven corrections.

CRediT authorship contribution statement

A. Kiener: Investigation, Methodology, Validation, Writing – original draft, Visualization, Software. **S. Langer:** Conceptualization, Writing – review & editing, Resources, Supervision, Project administration, Funding acquisition. **P. Bekemeyer:** Conceptualization, Writing – review & editing, Resources, Supervision, Software, Funding acquisition.

Declaration of competing interest

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

Data availability

The authors do not have permission to share data.

Appendix. Hyperparameter tuning

The following sections summarize for each ML model the search space used during hyperparameter tuning and reports the final hyperparameters.

A.1. Random forest

The tuned hyperparameters for the RF model are the number of decision tree estimators N and the maximum features for a split x_{max} (see Table A.4). For the RF implementation the Scikit-learn library [37] was used.

Table A.4
Hyperparameter search space for RF.

Hyperparameter	Search space	Final hyperparameter
N	[50–350]	236
x_{max}	[5–25*]	12

* Corresponding to the max. number of available features in the 2D case.

A.2. Neural network

For the NN architecture the number of hidden layers l, number of neurons in each hidden layer n_l, the batch size b, activation function for the input and hidden layers g, optimizer o and the learning rate l_r with and without exponential decrease are chosen for the tuning process (see Table A.5). For the implementation of the NN the SMARTy library [38] was used with the PyTorch backend [39].

Table A.5
Hyperparameter search space for NN.

Hyperparameter	Search space	Final hyperparameter
l	[4–12]	7
n_l	[50–550]	317
b	[32, 128, 256, 1,024, 1,280*, 2,048]	1024
g	[tanh, relu]	tanh
o	[Adam, SGD]	Adam
l_r	[0.00001–0.001]	0.000324 with exp. decrease

*Corresponds to the number of data points in one 2D simulation case.

A.3. Graph neural network

For the GNN architecture the same ADAM optimizer and tanh activation function were chosen for the NN training. The learning rate l_r and the number of stacked convolutional layers l_c (which corresponds to an aggregation of "hops" in the neighborhood around each node) and the feature dimensionality H_l of each such layer (excluding in- and output) was optimized for (see Table A.6). The GNN was implemented using the pytorch geometric library [40] and specifically the implementation of the graph convolutional operator "GCNConv" proposed by Kipf et al. [33] was used.

Table A.6
Hyperparameter search space for GNN.

Hyperparameter	Search space	Final hyperparameter
l_r	[0.00001–0.01]	0.00162
l_c	[4–12]	9
H_l	[50–550]	285

References

[1] Brunton SL, Noack BR, Koumoutsakos P. Machine learning for fluid mechanics. Annu Rev Fluid Mech 2020;52(1):477–508. http://dx.doi.org/10.1146/annurev-fluid-010719-060214.

[2] Vinuesa R, Brunton SL. The potential of machine learning to enhance computational fluid dynamics. 2021, http://dx.doi.org/10.48550/ARXIV.2110.02085, URL https://arxiv.org/abs/2110.02085.

[3] Duraisamy K. Perspectives on machine learning-augmented Reynolds-averaged and large eddy simulation models of turbulence. Phys Rev Fluids 2021;6:050504. http://dx.doi.org/10.1103/PhysRevFluids.6.050504, URL https://link.aps.org/doi/10.1103/PhysRevFluids.6.050504.

[4] Fukami K, Fukagata K, Taira K. Assessment of supervised machine learning methods for fluid flows. Theor Comput Fluid Dyn 2020;34(4):497–519. http://dx.doi.org/10.1007/s00162-020-00518-y.

[5] Franz T, Zimmermann R, Goertz S, Karcher N. Interpolation-based reduced-order modelling for steady transonic flows via manifold learning. Int J Comput Fluid Dyn 2014;28(3–4):106–21.

[6] Raveh DE. CFD-based models of aerodynamic gust response. J Aircr 2007;44(3):888–97.

[7] Schmid PJ. Dynamic mode decomposition of numerical and experimental data. J Fluid Mech 2010;656:5–28.

[8] Hines D, Bekemeyer P. Graph neural networks for the prediction of aircraft surface pressure distributions. Aerosp Sci Technol 2023;137:108268. http://dx.doi.org/10.1016/j.ast.2023.108268, URL https://www.sciencedirect.com/science/article/pii/S1270963823001657.

[9] Sabater C, Stürmer P, Bekemeyer P. Fast predictions of aircraft aerodynamics using deep-learning techniques. AIAA J 2022;60(9):5249–61. http://dx.doi.org/10.2514/1.J061234.

[10] Fukami K, Fukagata K, Taira K. Super-resolution reconstruction of turbulent flows with machine learning. J Fluid Mech 2019;870:106–20. http://dx.doi.org/10.1017/jfm.2019.238, URL http://arxiv.org/pdf/1811.11328v2.

[11] Jiang CM, Esmaeilzadeh S, Azizzadenesheli K, Kashinath K, Mustafa M, Tchelepi HA, et al. MeshfreeFlowNet: A physics-constrained deep continuous space-time super-resolution framework. In: Proceedings of the international conference for high performance computing, networking, storage and analysis. IEEE Press; 2020.

[12] Gao H, Sun L, Wang J-X. Super-resolution and denoising of fluid flow using physics-informed convolutional neural networks without high-resolution labels. Phys Fluids 2021;33(7):073603. http://dx.doi.org/10.1063/5.0054312.

[13] Park SC, Park MK, Kang MG. Super-resolution image reconstruction: a technical overview. IEEE Signal Process Mag 2003;20(3):21–36. http://dx.doi.org/10.1109/MSP.2003.1203207.

[14] Dong C, Loy CC, He K, Tang X. Image super-resolution using deep convolutional networks. IEEE Trans Pattern Anal Mach Intell 2016;38(2):295–307. http://dx.doi.org/10.1109/TPAMI.2015.2439281.

[15] Gao H, Sun L, Wang J-X. PhyGeoNet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state PDEs on irregular domain. J Comput Phys 2021;428(5):110079. http://dx.doi.org/10.1016/j.jcp.2020.110079, URL http://arxiv.org/pdf/2004.13145v2.

[16] Morimoto M, Fukami K, Zhang K, Nair AG, Fukagata K. Convolutional neural networks for fluid flow analysis: toward effective metamodeling and low dimensionalization. Theor Comput Fluid Dyn 2021;35(5):633–58. http://dx.doi.org/10.1007/s00162-021-00580-0.

[17] Ogoke F, Meidani K, Hashemi A, Farimani AB. Graph convolutional networks applied to unstructured flow field data. Mach Learn: Sci Technol 2021;2(4):045020. http://dx.doi.org/10.1088/2632-2153/ac1fc9.

[18] Chen J, Hachem E, Viquerat J. Graph neural networks for laminar flow prediction around random two-dimensional shapes. Phys Fluids 2021;33(12):123607. http://dx.doi.org/10.1063/5.0064108.

[19] Belbute-Peres FdA, Economon TD, Kolter JZ. Combining differentiable PDE solvers and graph neural networks for fluid flow prediction. 2020, http://dx.doi.org/10.48550/ARXIV.2007.04439, URL https://arxiv.org/abs/2007.04439.

[20] Bode M, Gauding M, Lian Z, Denker D, Davidovic M, Kleinheinz K, et al. Using physics-informed enhanced super-resolution generative adversarial networks for subfilter modeling in turbulent reactive flows. In: Proceedings of the combustion institute, vol. 38, no. 2. 2021, p. 2617–25. http://dx.doi.org/10.1016/j.proci.2020.06.022, URL https://www.sciencedirect.com/science/article/pii/S1540748920300481.

[21] Obiols-Sales O, Vishnu A, Malaya NP, Chandramowlishwaran A. SURFNet: Super-resolution of turbulent flows with transfer learning using small datasets. In: 2021 30th international conference on parallel architectures and compilation techniques. 2021, p. 331–44. http://dx.doi.org/10.1109/PACT52795.2021.00031.

[22] Wang L, Luo Z, Xu J, Luo W, Yuan J. A novel framework for cost-effectively reconstructing the global flow field by super-resolution. Phys Fluids 2021;33(9):095105. http://dx.doi.org/10.1063/5.0062775.

[23] Kochkov D, Smith J, Alieva A, Wang Q, Brenner M, Hoyer S. Machine learning–accelerated computational fluid dynamics. Proc Natl Acad Sci 2021;118. http://dx.doi.org/10.1073/pnas.2101784118.

[24] Um K, Brand R, Fei YR, Holl P, Thuerey N. Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers. In: Larochelle H, Ranzato M, Hadsell R, Balcan M, Lin H, editors. Advances in neural information processing systems, vol. 33. Curran Associates, Inc.; 2020, p. 6111–22, URL https://proceedings.neurips.cc/paper/2020/file/43e4e6a6f341e00671e123714de019a8-Paper.pdf.

[25] Hanna BN, Dinh NT, Youngblood RW, Bolotnov IA. Machine-learning based error prediction approach for coarse-grid Computational Fluid Dynamics (CG-CFD). Prog Nucl Energy 2020;118:103140. http://dx.doi.org/10.1016/j.pnucene.2019.103140.

[26] Mavriplis DJ. Multigrid techniques for unstructured meshes. Tech. rep., Institute for Computer Applications in Science and Engineering (ICASE); 1995.

[27] Brandt A, Livne OE. Multigrid techniques. Society for Industrial and Applied Mathematics; 2011, http://dx.doi.org/10.1137/1.9781611970753, URL https://epubs.siam.org/doi/abs/10.1137/1.9781611970753.

[28] Trottenberg U, Oosterlee C, Schuller A, Brandt A. Multigrid. Elsevier Science; 2001, URL https://books.google.de/books?id=-og1wD-Nx_wC.

[29] Allmaras S, Johnson F, Spalart P. Modifications and clarifications for the implementation of the spalart-allmaras turbulence model. In: Seventh international conference on computational fluid dynamics. 2012, p. 1–11.

[30] White JA, Nishikawa H, Baurle RA. Weighted least-squares cell-average gradient construction methods for the VULCAN-CFD second-order accurate unstructured grid cell-centered finite-volume solver. In: AIAA scitech 2019 forum. 2019, http://dx.doi.org/10.2514/6.2019-0127, URL https://arc.aiaa.org/doi/abs/10.2514/6.2019-0127.

[31] Breiman L. Random forests. Mach Learn 2001;45:5–32. http://dx.doi.org/10.1023/A:1010933404324.

[32] Scarselli F, Gori M, Tsoi AC, Hagenbuchner M, Monfardini G. The graph neural network model. IEEE Trans Neural Netw 2009;20(1):61–80. http://dx.doi.org/10.1109/TNN.2008.2005605.

[33] Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. In: 5th international conference on learning representations. 2016, http://dx.doi.org/10.48550/ARXIV.1609.02907, URL https://arxiv.org/abs/1609.02907.

[34] Bergstra J, Bardenet R, Bengio Y, Kégl B. Algorithms for hyper-parameter optimization. In: Shawe-Taylor J, Zemel R, Bartlett P, Pereira F, Weinberger K, editors. Advances in neural information processing systems, vol. 24. Curran Associates, Inc.; 2011, URL https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[35] Kroll N, Langer S, Schwöppe A. The DLR flow solver TAU - Status and recent algorithmic developments. In: 52nd aerospace sciences meeting. 2014, http://dx.doi.org/10.2514/6.2014-0080, URL https://arc.aiaa.org/doi/abs/10.2514/6.2014-0080.

[36] Langer S. An initial investigation of solving RANS equations in combination with two-equation turbulence models. Tech. rep., Institut für Aerodynamik und Strömungstechnik; 2019, URL https://elib.dlr.de/129170/.

[37] Buitinck L, Louppe G, Blondel M, Pedregosa F, Mueller A, Grisel O, et al. API design for machine learning software: experiences from the scikit-learn project. In: ECML PKDD workshop: Languages for data mining and machine learning. 2013, p. 108–22.

[38] Bekemeyer P, Bertram A, Chaves DAH, Ribeiro MD, Garbo A, Kiener A, et al. Data-driven aerodynamic modeling using the DLR SMARTy toolbox. In: AIAA AVIATION 2022 forum. 2022, http://dx.doi.org/10.2514/6.2022-3899, AIAA 2022-3899. URL https://arc.aiaa.org/doi/abs/10.2514/6.2022-3899.

[39] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An imperative style, high-performance deep learning library. In: Advances in neural information processing systems 32. Curran Associates, Inc.; 2019, p. 8024–35, URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[40] Fey M, Lenssen JE. Fast graph representation learning with PyTorch geometric. In: ICLR workshop on representation learning on graphs and manifolds. 2019.

[41] Akiba T, Sano S, Yanase T, Ohta T, Koyama M. Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25rd ACM SIGKDD international conference on knowledge discovery and data mining. 2019.

[42] Ceresola N, Djayapertapa L, Heinrich R, Leichner S, Berglind TE, Caballero Rubiato N, et al. Task 5 oscillating LANN wing. 2006, GARTEUR AD(AG38) - Time Accurate Methods, Chapter 7.

[43] Zwaan IRJ. LANN wing pitching oscillations. 1982, Compendium of Unsteady Aerodynamic Measurements, AGARD-R-702 Addendum No. 1.