TOWARDS FAST AERODYNAMIC SIMULATIONS WITH MACHINE LEARNING CORRECTIONS FOR DISCRETIZATION ERRORS

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

* 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

Abstract

Computational fluid dynamics has become an important method for aerodynamic analysis, supplementing or partially even substituting wind tunnel experiments or flight tests. A spatial discretization on so-called grids is used to approximate a solution of the boundary-value problem of interest. In general, increasing the number of degrees of freedom, the accuracy of the approximate solution improves, typically going hand-in-hand with additional computational complexity. Consequently, conducting a rigorous amount of accurate simulations, as for example for a complete flight envelope, is possibly infeasible. Thus, there is a need for novel approaches to reduce the overall numerical cost to allow for faster and inexpensive design process iterations. This work shows how machine learning methods can be employed as a post-processing tool to improve the accuracy of comparably inexpensive low-fidelity results, including coarse grid finite volume and low-order discontinuous Galerkin simulations. It is shown that using two different regression models, a random forest and a graph neural network, inaccurate simulations can be corrected to approximate the high-fidelity simulation projected to the low-fidelity discretization. Improved flow fields are obtained as well as improvements in pressure and lift coefficient. A main limitation of the method involves the loss of accuracy due to projection, resulting in less significant corrections of velocity gradient dependent values, such as friction and drag coefficient. Independent of the chosen discretization, be it finite volume or discontinuous Galerkin, results show possibilities and drawbacks of applying data-driven methods to correct low-fidelity simulations. It is anticipated that the proposed method can be used to quickly iterate over simulations conducted within a chosen design space, which could entail a given flight envelope. Furthermore, this work is a promising baseline for further research, including the correction of unsteady simulations and the use of a machine learning based error indicator for refinement strategies.

Keywords

Discretization Error; Correction; Finite Volume; Discontinuous Galerkin; Machine Learning

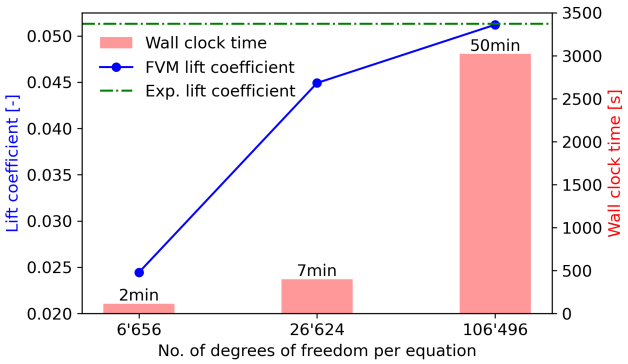
NOMENCLATURE

Abbreviations

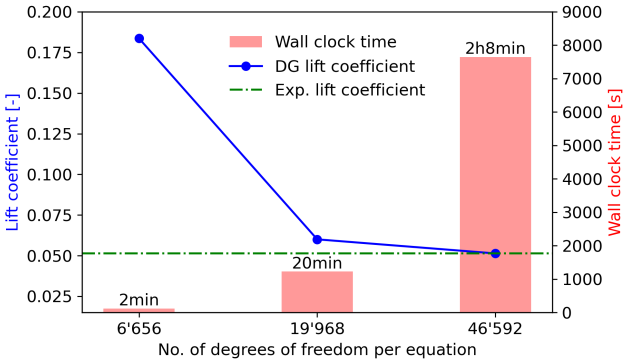
CFD	Computational Fluid Dynamics
DG	Discontinuous Galerkin
DoFs	Degrees of Freedom
FV	Finite Volume
GNN	Graph Neural Network
ML	Machine Learning
RF	Random Forest

1. INTRODUCTION

Computational fluid dynamics (CFD) has become an important method for aerodynamic analysis in the aerospace industry, allowing insights into complex fluid phenomena for analysis or optimization tasks. Even if CFD supplements or partially even substitutes wind tunnel experiments or flight tests, conducting full scale simulations is still computationally demanding. Currently, second order Finite Volume (FV) is the industrial standard in terms of spatial discretization to solve for the equations of interest for fluid problems [1]. Another method which receives more attention from academia than from industry is the so-called Discontinuous Galerkin (DG) method [2]. Both methods require the design of a grid on which the equations are discretized. Accurate results with FV are achieved by refinement of these grids. Thus, it is common to have grids with several hundred million elements and Degrees of Freedom (DoFs). With DG, a solution is approximated as a piece-wise polynomial per grid element. Thus, different levels of accuracy can be achieved on the same grid by increasing the polynomial degree. Nevertheless, for both FV and DG, the computational cost increases significantly with a higher number of DoFs and improved accuracy. For a laminar flow around a NACA0012 airfoil this effect is



(a) FV: number of DoFs increases with elements in the grid



(b) DG: number of DoFs increases with polynomial degree

FIG 1. Cost and accuracy versus DoFs using FV and DG

presented in figure 1 for FV and DG: increasing the number of elements for FV or the polynomial degree for DG leads to a more accurate lift coefficient approximating the reference value [3]. Obviously, the computational time does not scale linearly with the number of DoFs. Though only shown exemplary, this effect can be observed for other test cases [4–6]. Thus, it is necessary to investigate novel approaches to improve these methods to enable faster and computationally less expensive design process iterations. Many methods aim to decrease the computational cost of high-fidelity and accurate simulations, as obtained by FV on fine grids or by DG with a high polynomial degree. Such approaches include multi-grid methods [7] and adaptive refinement of the grid and of the polynomial degree [8,9]. Still, for 3-D industrial test cases, so far none of the developed methods show in general a linear scaling of the computational complexity with respect to the number of DoFs. As a consequence, in addition to improve algorithms as the ones previously mentioned, this work explores the possible use of Machine Learning techniques to improve the accuracy of low-fidelity discretizations, such as coarse grid FV or low-order DG simulations, while preserving their low simulation costs. This has recently been the aim of many data-driven approaches, where the core idea is to inject high-fidelity information into the low-fidelity solution by a Machine Learning (ML) trained model. For instance, a neural network has been trained to predict corrected drag forces for coarse grid fluid-particle interactions in [10]. The authors achieved improved results across varying metrics, including the run-off distance of particles. [11] directly corrects the error resulting from the coarse grid discretization, e.g. the discretization error, of FV simulations. Two different ML models, a random forest and a neural network, were trained to predict the element-wise correction based on local flow features for a 3-D flow in a lid-driven cavity. This approach was adopted by [12], where a random forest was used to correct the flow around a bluff-body in an enclosed duct, computed on coarse grids. Similarly, [13] employ a neural network to correct coarse grid induced errors of mixing flows inside spinner flask bioreactors. While limitations were encountered regarding the generalization capabilities of the trained models, all three studies report reduced errors while increasing computational efficiency compared to the fine grid counterpart simulation. For DG simulations, similar work has been conducted to increase the efficiency of unsteady simulations. DoFs of low-order DG simulations for the 1-D Burgers' equation [14] and the 3-D Navier-Stokes equations for the Taylor Green Vortex problem [15] were corrected by employing a neural network, aiming to approximate the filtered high-order simulation.

This work extends the approach proposed in [11] to the turbulent flow around an RAE2822 airfoil for coarse grid FV, as initially explored in [16], and low-order DG simulations. For all CFD computations, the Reynolds Averaged Navier-Stokes equations with Spalart-Allmaras turbulence model are employed and data is generated by varying Mach number and angle of attack. Two ML models are explored: a random forest and a graph neural network, with latter being a natural choice for graph structured data, which is the case for simulations obtained on grids.

This paper is structured as follows: section 2 describes the methodology to correct coarse grid FV simulations, followed by the correction approach for low-order DG simulations. Then, random forest, graph neural network, and the selected model input features are presented. Section 3 discusses the choice of design space for sample generation, the ML training and the ML corrected solutions, before closing with a conclusion in section 4.

2. METHODOLOGY

Let $\boldsymbol{u}$ be the unknown function and $\tilde{\boldsymbol{u}}$ an approximate solution of the discretized problem. This work considers the residual $R(\tilde{\boldsymbol{u}}) = 0$, where R corresponds to a boundary value problem of the Reynolds Averaged Navier-Stokes equations [4, 17].

2.1. Correction of Coarse Grid FV Simulations

Denoting the coarse and fine grids with subscript c and f, respectively, the variables of interest on each grid become $\tilde{\boldsymbol{u}}_c$ and $\tilde{\boldsymbol{u}}_f$. For the FV study of this work, the goal is to quantify and learn the discretization error on the coarse grid. Thus, a fine-to-coarse mapping operator I_f^c is introduced, such that the fine grid solution

$\tilde{\boldsymbol{u}}_f$ is projected to the coarse grid discretization:

$$\tilde{\boldsymbol{u}}_{I,c} = I_f^c(\tilde{\boldsymbol{u}}_f) \tag{1}$$

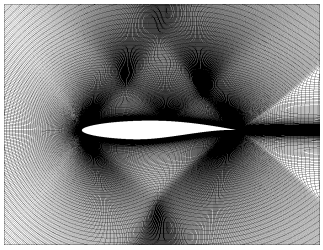
Subsequently, the error $\Delta\tilde{\boldsymbol{u}}$ can be formulated as:

$$\Delta\tilde{\boldsymbol{u}} = \tilde{\boldsymbol{u}}_{I,c} - \tilde{\boldsymbol{u}}_c \tag{2}$$

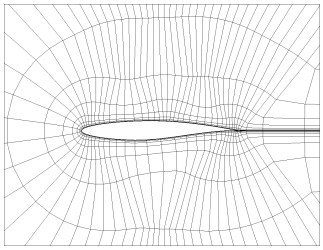
After the ML training, the model returns a prediction $\Delta\hat{\boldsymbol{u}}$ of the true error $\Delta\tilde{\boldsymbol{u}}$. This prediction is used to correct coarse grid simulations unseen during the ML training to approximate the projected fine grid solution on the coarse grid discretization:

$$\tilde{\boldsymbol{u}}_c + \Delta\hat{\boldsymbol{u}} \approx \tilde{\boldsymbol{u}}_{I,c} \tag{3}$$

For the FV simulations, the DLR CFD solver TAU is employed [4], which uses a vertex-based data structure. The coarse and fine C-type grids for this study consist of 1'280 and 327'680 elements, respectively, and are shown in figure 2, while the number of DoFs per equation are reported in table 1. A detailed study for the complete grid sequence can be found in [18]. Since these grids are nested, injection can be used as mapping operator I_f^c, which takes for each coarse grid node the value of the corresponding fine grid node. The variables of interest to be corrected include density, velocity, and pressure, such that $\boldsymbol{u} = [\rho, U_x, U_z, p]^T$.



(a) Fine grid for the FV study



(b) Coarse grid for the FV study

FIG 2. Fine and coarse grid used for the FV study

Coarse	Fine
1'368	329'088

TAB 1. Degrees of freedom per equation for FV study

2.2. Correction of Low-Order DG Simulations

Employing a DG discretization with orthonomal basis function, the numerical solution $\tilde{\boldsymbol{u}}$ is expressed in each element k as piecewise polynomial function:

$$\tilde{\boldsymbol{u}} = \sum_{i=1}^{N} a_i \boldsymbol{\psi}_i \tag{4}$$

with $\boldsymbol{a} = [a_1, \ldots, a_N]$ being the unknowns of $\tilde{\boldsymbol{u}}$ at each DoF in element k, $\boldsymbol{\psi} = [\psi_1, \ldots, \psi_N]$ the basis functions defined by a modified Gram-Schmidt procedure [19], and N the number of DoFs per equation per element. Denoting the low-order and high-order solutions with subscript LO and HO, respectively, the variables of interest are denoted as $\tilde{\boldsymbol{u}}_{LO}$ and $\tilde{\boldsymbol{u}}_{HO}$ with their unknowns $\boldsymbol{a}_{LO}$ and $\boldsymbol{a}_{HO}$ and it is assumed that $N_{LO} < N_{HO}$. To define the discretization error on the low-order discretization, which is later to be learned by the ML model, the high-order DoFs are truncated to the low-order discretization, resulting in the truncated solution $\tilde{\boldsymbol{u}}_{T,LO}$:

$$\tilde{\boldsymbol{u}}_{T,LO} = \sum_{i=1}^{N_{LO}} a_{i,HO} \boldsymbol{\psi}_i \tag{5}$$

This results in an L_2 projection, since the basis is hierarchical and orthonormal. With the truncated DoFs, the error is computed as:

$$\Delta a_i = a_{i,HO} - a_{i,LO} \tag{6}$$

for $1 \leq i \leq N_{LO}$. Thus, with the ML predictions $\Delta\hat{\boldsymbol{a}}$ approximating the true error $\Delta\boldsymbol{a}$, the LO solution can be corrected to approximate the truncated solution $\tilde{\boldsymbol{u}}_{T,LO}$:

$$\sum_{i=1}^{N_{LO}} (a_{i,LO} + \Delta\hat{a}_i)\boldsymbol{\psi}_i \approx \tilde{\boldsymbol{u}}_{T,LO}. \tag{7}$$

For the DG simulations, the CFD software developed by ONERA, DLR, Airbus (CODA) [17] is used, employing a cell-centered data structure. The variables of interest include density, momentum, and total energy, such that $\boldsymbol{u} = [\rho, M_x, M_z, E]^T$. For both high- and low-order simulations, the same C-type grid is employed with 2'464 quadratic elements, as given in figure 3. The accurate high-order solution is achieved using a polynomial degree of 2, whereas inaccurate simulation results are obtained using polynomial degrees of 0 and 1. Here, only the results correcting low-order of polynomial degree 0 are discussed. The number of DoFs per equation are reported in table 2.

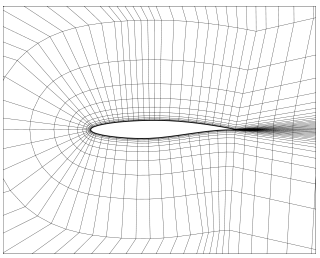


FIG 3. Grid used for the DG study

Degree 0	Degree 2
2'464	17'248

TAB 2. Degrees of freedom per equation for the DG study

2.3. ML Model 1: Random Forest

A Random Forest (RF) model is used as a baseline model for this study. Compared to graph neural networks, RFs are easy to train with few hyperparameters needing adjustment to quickly obtain a well fitted model. RF was first introduced by Breiman [20] and is classified as an ensemble method, combining the results of a multitude of trained weak learners. For RF, the weak learners are decision trees and the final prediction result $\hat{y}$ is the averaged output of all N_{DT} decision trees, where a single decision tree prediction is based on the feature vector input $\boldsymbol{x}_k$, such that

$$\hat{y} = \frac{1}{N_{DT}} \sum_{i=1}^{N_{DT}} \hat{y}_i(\boldsymbol{x}_k) \tag{8}$$

2.4. ML Model 2: Graph Neural Network

The core principle of Graph Neural Networks (GNN) is to update the initial vertex feature vector of the currently considered vertex v_k given as $\boldsymbol{h}_k^{t=0} = \boldsymbol{x}_k$ to a new node feature vector representation $\boldsymbol{h}_k^{t+1}$. This update is dependent on the feature vector of the current vertex at previous step $\boldsymbol{h}_k^t$, the vertex features of all neighbouring vertices $\boldsymbol{h}_j^t$ for all $v_j \in N(v_k)$, and if applicable any edge feature vectors $\boldsymbol{e}_{j,k}$ for all $v_j \in N(v_k)$. This is done within three steps: the message passing, the aggregation, and the final update of the feature vector representation. During the message passing, a function M_t is defined with parameters to be optimized for during training. This function passes neighbourhood information as a message $\boldsymbol{m}_{j,k}^{t+1}$ from each neighbour vertex v_j to vertex v_k, as given in equation 9. During the aggregation step, all messages received from neighbouring vertices need to be combined into the message $\boldsymbol{m}_k^{t+1}$. This aggregation function $\bigoplus$ is permutation-invariant and often consists of a sum or mean, such that the final message is independent of the number of neighbours or their indices. The general aggregation function is given in equation 10. In the final step, the feature vector representation of current vertex v_k is updated based on a function U_t, which similarly to the message passing function consists of parameters to be optimized for during training. The update function U_t takes into account the previous feature vector $\boldsymbol{h}_k^t$ and the aggregated message $\boldsymbol{m}_k^{t+1}$, as given in equation 11. Combining these three steps of message passing, aggregation and update, yields equation 12.

$$\boldsymbol{m}_{j,k}^{t+1} = M_t(\boldsymbol{h}_j^t, \boldsymbol{h}_k^t, \boldsymbol{e}_{j,k}) \tag{9}$$

$$\boldsymbol{m}_k^{t+1} = \bigoplus_{v_j \in N v_k} \boldsymbol{m}_{j,k}^{t+1} \tag{10}$$

$$\boldsymbol{h}_k^{t+1} = U_t(\boldsymbol{h}_k^t, \boldsymbol{m}_k^{t+1}) \tag{11}$$

$$\boldsymbol{h}_k^{t+1} = U_t\left(\boldsymbol{h}_k^t, \bigoplus_{v_j \in N v_k} M_t(\boldsymbol{h}_j^t, \boldsymbol{h}_k^t, \boldsymbol{e}_{j,k})\right) \tag{12}$$

For CFD simulations, the data is already represented in a graph structure using the computational grid, thus GNNs are easily applicable even with unstructured and three dimensional grids. The advantage of using GNNs compared to the RF approach is that its vertex-wise predictions are not only based on the current vertex features, but also takes into account neighbourhood information, and optionally also edge information.

2.5. ML Input Features

For both FV and DG, each ML model predicts 4 outputs and receives 25 inputs per vertex or element. For FV, the ML models predict for each vertex separately one discretization error per variable of interest, whereas for the DG study this is done for each element. The input features are defined as the first and second derivatives of the variables of interest, computed by a weighted least squares approach, and a local cell Reynolds number which is formulated using the local velocity, wall distance d_w and molecular viscosity ν, as given in equation 13, where k denotes either vertex or element.

$$Re_k = \frac{|U_k| d_{w,k}}{\nu_k} \tag{13}$$

3. RESULTS

This section reports the sample points used for the generation of the data set and the training of the ML models, which is identical for FV and DG if not stated differently. Then, results on the test set samples are presented firstly for the FV study, followed by results for the DG study. The section closes with a discussion of possible limitations of the proposed methodology.

3.1. Design of Experiment and ML Training

For both FV and DG studies, steady-state simulations are performed across Mach numbers $0.52 \leq M \leq 0.82$ and angles of attack $0.0 \leq \alpha \leq 9.0$, as depicted in figure 4. For higher angles of attack, 30 points are chosen using a Halton sampling, for lower angles of attack additional 30 points are added using a Latin Hypercube sampling. These 60 points make up the train and validation set, whereas additional 29 samples are added at Mach numbers 0.6 and 0.75 for testing of the trained ML models, a strategy previously deployed

in [16]. The average simulation wall clock times are reported in table 3 and 4.
For the ML training, the training data is divided into five folds. Only for the FV study, a 5-fold cross validation is used to find optimal hyperparameters, as reported in [16]. For the DG study and both ML models, the same hyperparameters found during the FV study are employed. The hyperparameters used for the RF are: 236 decision tree estimators, 12 features considered for each split, and no bootstrapping applied. For the GNN, the found hyperparameters are 0.00162 for the the initial learning rate, a gamma value of 0.995 for the exponential decay of the learning rate, 285 hidden channels with 9 graph convolutional layers [21], Adam optimizer, and hyperbolic tangent as non-linear layer. For the hyperparameter optimization, the Optuna library [22] was employed, for the RF models the scikit-learn library [23], and for GNN the SMARTy library [24] with pytorch-geometric backend [25].

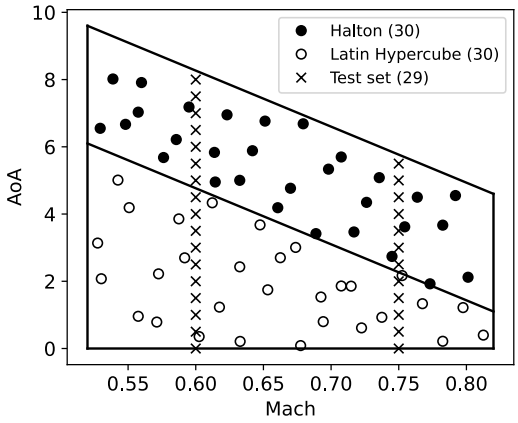


FIG 4. Design of Experiment for FV and DG

Coarse	Fine
1 min 30 s	7 h

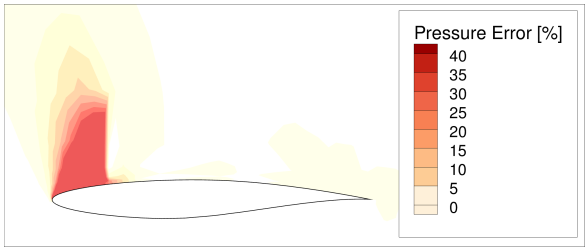
TAB 3. Average wall clock time for FV simulations

Degree 0	Degree 2
30 s	50 min

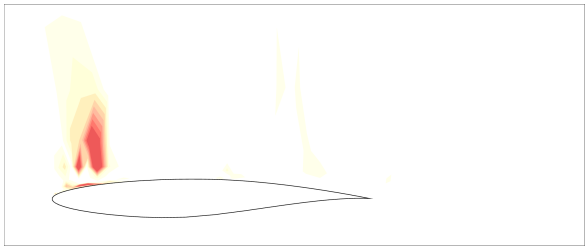
TAB 4. Average wall clock time for DG simulations

3.2. ML Corrected FV Solutions

Since the main aim of this study is to correct flow fields, the first qualitative result presented is the percentage error of the pressure field before and after the ML correction with respect to the fine grid solution mapped onto the coarse grid discretization. Figure 5 depicts this error for test set at Mach 0.6 and angle of attack 7.5°. At this high angle of attack, the error is mainly located at the developing shock location around the leading edge on the suction side of the airfoil. The RF correction reduces the error, whereas remaining larger errors are observed close to the airfoil boundary. Using the GNN, the error is additionally decreased.
It is of further interest to investigate how well quantities derived from the corrected flow fields perform, since quantitative values such as the pressure coefficient C_p and lift coefficient C_L are of interest. Figure 6 shows C_p along the airfoil for Mach 0.75 and angle of attack 5.0° for the coarse and fine grid simulations, as well as the derived C_p from the ML corrected flow fields. The coarse grid simulation does not capture the shock appropriately. On the other hand, the RF corrected coarse grid solution better captures the shock, but shows noisy behaviour along the suction side, specifically towards the trailing edge of the airfoil. With the GNN, the error is reduced significantly, resulting in a smoother correction. Nevertheless, discrepancies between corrected and fine grid solution remain close to the shock location.
Figure 7 depicts the lift coefficient for both test sets at Mach 0.6 and Mach 0.75. Again, the coarse grid simulation results deviate from the fine grid simulation, more dominantly at high angles of attack, where non-linearities, such as shocks, are encountered which are not appropriately captured by the low-fidelity discretization. Both corrections, be it RF or GNN, in general improve the lift coefficient, very well matching the fine grid results at low angles of attack. At higher angles of attack, the results slightly deviate, more so for the RF correction.



(a) Coarse grid simulation



(b) Coarse grid simulation + RF correction

(c) Coarse grid simulation + GNN correction

FIG 5. Pressure error [%] for M=0.6 and α=7.5, FV

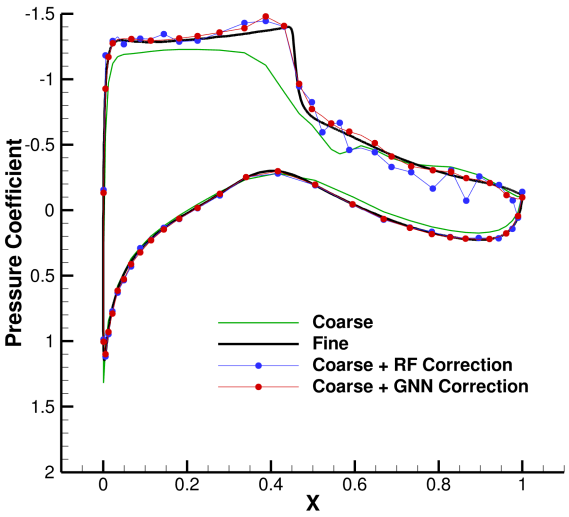
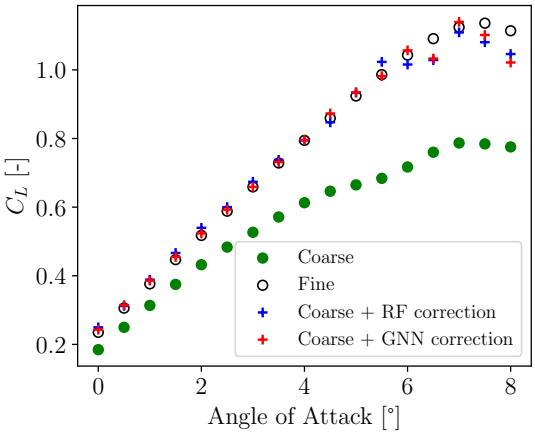
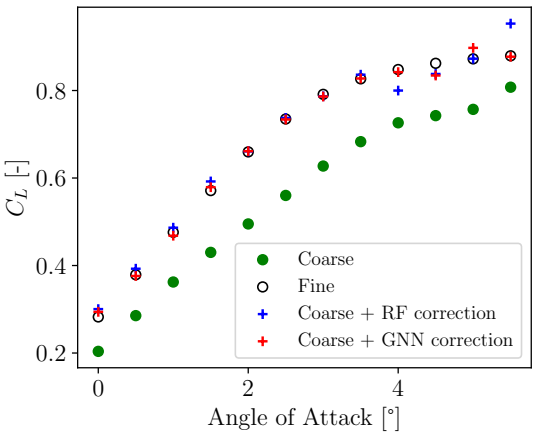


FIG 6. C_p for M=0.75 and α=5.0, FV



(a) Lift coefficient, test set M=0.6, FV



(b) Lift coefficient, test set M=0.75, FV

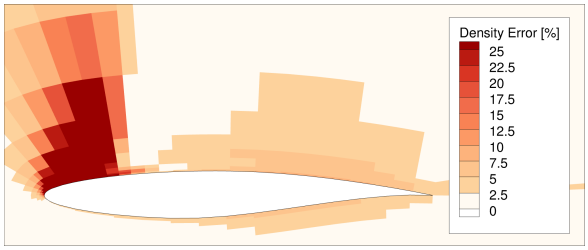
FIG 7. Lift coefficient for both test sets, FV

3.3. ML Corrected DG Solutions

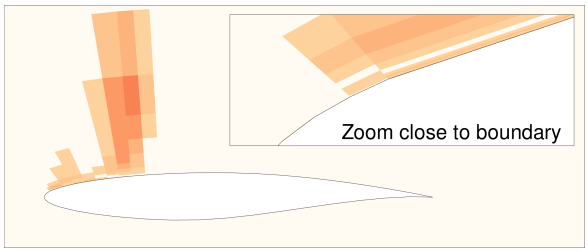
As for the FV study, the first qualitative results of interest are the corrected fields. Figure 8 depicts the density error of the low-order simulation compared to the truncated high-order solution at Mach 0.6 and angle of attack 7.5°. Again, the error is mainly located at the developing shock location and both ML model corrections reduce it, while the GNN corrects the error more significantly. Differently to the FV correction, the GNN corrected field exhibits greater errors close to the boundary than the RF correction. This can be seen in figure 8.

This behaviour is reflected in the evaluation of surface related values, such as the pressure coefficient in figure 9. Although the GNN approximates the pressure coefficient obtained by the high-order method well at the developing shock location, it shows overall greater discrepancies than the RF corrected solution, due to the correction being less significant along the airfoil boundary.

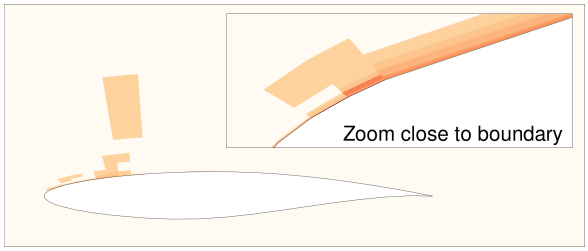
Looking at the resulting lift coefficients for both test sets in figure 10, both ML models achieve good performance for lower angles of attack. As for the previous FV study, the performance drops for high angles of attack, where non-linear phenomena are dominant. For the DG correction it can be concluded that the RF correction results in overall better surface related values, since the correction in the elements along the boundary layer is better compared to the GNN model.



(a) Low-order simulation



(b) Low-order simulation + RF correction



(c) Low-order simulation + GNN correction

FIG 8. Density error [%] for M=0.6 and α=7.5, DG

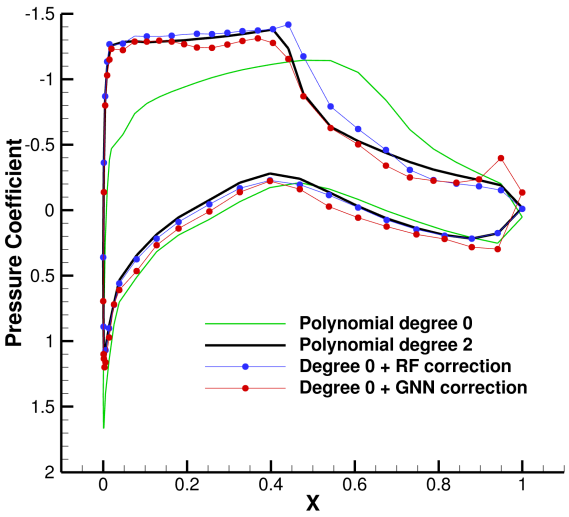
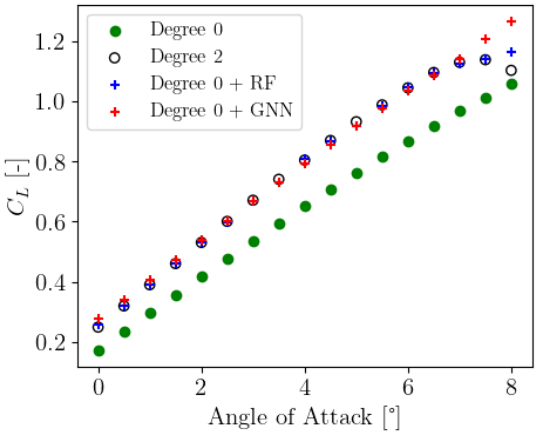
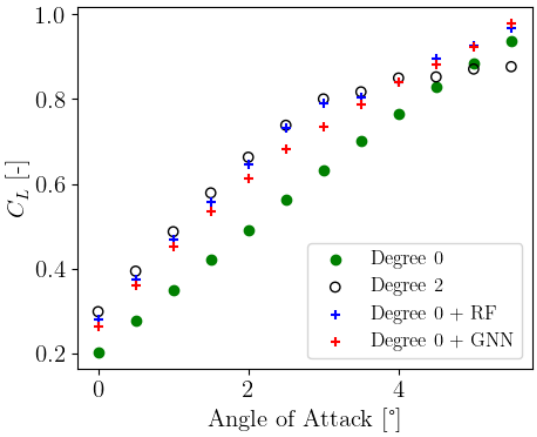


FIG 9. C_p for M=0.75 and α=5.0, DG



(a) Lift coefficient, test set M=0.6, DG



(b) Lift coefficient, test set M=0.75, DG

FIG 10. Lift coefficient for both test sets, DG

3.4. Limitations

For both discretization strategies, FV and DG, the main limitation is the loss of accuracy after projection of the high-fidelity solution to the low-fidelity discretization space. For variables such as pressure and lift coefficient, this is not relevant as shown in the previous result sections. This limitation is of relevance for variables such as friction and drag coefficients, which depend on velocity gradients. A possible explanation is that these gradients might not be well resolved by the respective low-fidelity discretization. This is shown in figures 11 and 12, with the friction coefficient along the airfoil surface for the FV study and the drag coefficient for the DG study. It can be seen that with the projection from fine to coarse grid and the truncation of the higher polynomial degree only slight improvements are achieved. Figures 11 and 12 furthermore depict that the ML correction can at best only approximate these mapped or truncated solutions and not the respective fine grid or high polynomial degree solution. This, since the correction term is defined by the projected solution or truncated degrees of freedom, as given in equations 2 and 6, respectively.

Further drawbacks of the method, which are akin to any data-driven method, are the need to generate high-fidelity data for the training stage and the invested time required to find optimal hyperparameters of the ML models.

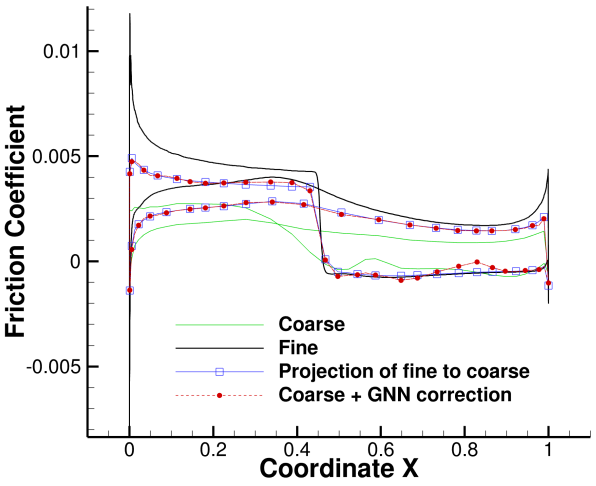


FIG 11. C_f for M=0.75 and α=5.0, FV

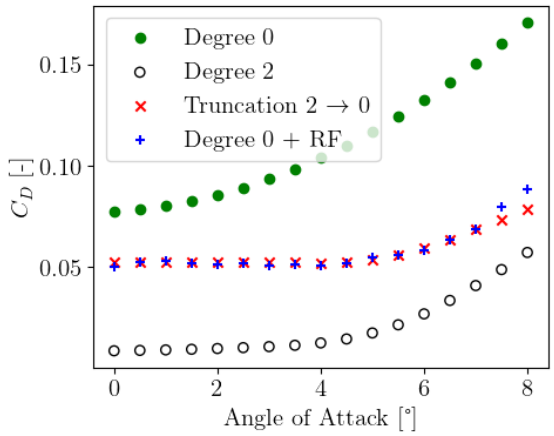


FIG 12. Drag coefficient, test set M=0.6, DG

4. CONCLUSION

Because of computational costs, conducting a vast amount of high-fidelity CFD simulations can be a daunting task. Therefore, it is of interest to reduce computational costs while maintaining a sufficient amount of accuracy. This study presents a procedure to quantify and learn discretization errors of low-fidelity simulations, specifically between coarse and fine grids for a FV discretization and between different polynomial degrees for a DG discretization.
To demonstrate the proposed method, the 2-D turbulent flow around an RAE2822 airfoil is employed. It is shown that using a RF and a GNN model, the low-fidelity flow fields can be corrected element- or vertex-wise, such that the resulting lift and pressure coefficient closely approximate the high-fidelity solution results. Since the GNN model takes neighbourhood information into account, it has shown better field corrections than the RF model for both FV and DG study. For the DG study though, the corrections close to the boundary layer were less significant, resulting in better lift and pressure coefficient using the RF model. Limitations of the method are found at high angles of attack, were the ML correction is less significant due to non-linear phenomena. Additionally, defining the error and thus the correction in terms of the mapped or truncated high-fidelity simulation, leads to a loss of accuracy which results in less significant corrections of friction and drag coefficient.
Future work will include the extension of the method to unsteady simulations and three dimensional DG cases, since promising results have been found for three dimensional FV problems in [16]. Additionally, for DG, it is of interest to investigate the corrective capabilities for polynomial degree greater than 0. Using the hyperparameters found during the FV study for the RF in the DG study showed satisfying results, whereas it can be expected that conducting a separate hyperparameter optimization study for the GNN could improve the results along the airfoil boundary. Furthermore, an interesting avenue would be to use the ML predictions not as correction, but as an error indicator for either h- or p-refinement. A direct comparison of the method between FV and DG is not conducted in this work, since the employed solvers, variables of interest, grids and number of degrees of freedom do not match. Such a comparative study matching these criteria would be of interest for further investigations.

ACKNOWLEDGEMENT

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

Contact address:

anna.kiener@dlr.de

References

[1] Freddie D. Witherden and Antony Jameson. Future directions in computational fluid dynamics. In 23rd AIAA Computational Fluid Dynamics Conference, 2017. DOI: 10.2514/6.2017-3791.

[2] Sebastian Boblest, Fabian Hempert, Malte Hoffmann, Philipp Offenhäuser, Matthias Sonntag, Filip Sadlo, Colin W. Glass, Claus-Dieter Munz, Thomas Ertl, and Uwe Iben. Toward a discontinuous galerkin fluid dynamics framework for industrial applications. In High Performance Computing in Science and Engineering '15, pages 531–545, Cham, 2016. Springer International Publishing.

[3] Robert Swanson and Stefan Langer. Steady-state laminar flow solutions for naca 0012 airfoil. Computers & Fluids, 126:102–128, 2016. DOI: https://doi.org/10.1016/j.compfluid.2015.11.009.

[4] Norbert Kroll, Stefan Langer, and Axel Schwöppe. The dlr flow solver tau - status and recent algorithmic developments. In 52nd Aerospace Sciences Meeting, 2014. DOI: 10.2514/6.2014-0080.

[5] Stefan Langer, Axel Schwöppe, and Norbert Kroll. Investigation and comparison of implicit smoothers applied in agglomeration multigrid. AIAA Journal, 53(8):2080–2096, 2015. DOI: 10.2514/1.J053367.

[6] Stefan Langer. Agglomeration multigrid methods with implicit runge–kutta smoothers applied to aerodynamic simulations on unstructured grids. Journal of Computational Physics, 277:72–100, 2014. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2014.07.050.

[7] Achi Brandt and Oren E. Livne. Multigrid Techniques. Society for Industrial and Applied Mathematics, 2011. DOI: 10.1137/1.9781611970753.

[8] Ralf Deiterding. Adaptive mesh refinement - theory and applications. Construction and Application of An AMR Algorithm for Distributed Memory Computers, 41:361–372, 01 2005.

[9] Fabio Naddei, Marta de la Llave Plata, and Vincent Couaillier. A comparison of refinement indicators for p-adaptive discontinuous Galerkin methods for the Euler and Navier-Stokes equations. 2018. DOI: 10.2514/6.2018-0368.

[10] Tsimur Davydzenka and Pejman Tahmasebi. High-resolution fluid–particle interactions: a machine learning approach. Journal of Fluid Mechanics, 938:A20, 2022. DOI: 10.1017/jfm.2022.174.

[11] Botros N. Hanna, Nam T. Dinh, Robert W. Youngblood, and Igor A. Bolotnov. Machine-learning based error prediction approach

for coarse-grid computational fluid dynamics (cg-cfd). Progress in Nuclear Energy, 118:103140, 2020. ISSN: 0149-1970. DOI: https://doi.org/10.1016/j.pnucene.2019.103140.

[12] Chin Yik Lee and Stewart Cant. A grid-induced and physics-informed machine learning cfd framework for turbulent flows. Flow, Turbulence and Combustion, 112, 12 2023. DOI: 10.1007/s10494-023-00506-2.

[13] Fernando José Cantarero-Rivera, Ran Yang, Haochen Li, Hairong Qi, and Jiajia Chen. An artificial neural network-based machine learning approach to correct coarse-mesh-induced error in computational fluid dynamics modeling of cell culture bioreactor. Food and Bioproducts Processing, 143:128–142, 2024. ISSN: 0960-3085. DOI: https://doi.org/10.1016/j.fbp.2023.11.004.

[14] Fernando Manrique de Lara and Esteban Ferrer. Accelerating high order discontinuous galerkin solvers using neural networks: 1d burgers' equation. Computers & Fluids, 235:105274, 2022. ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2021.105274.

[15] Fernando Manrique de Lara and Esteban Ferrer. Accelerating high order discontinuous galerkin solvers using neural networks: 3d compressible navier-stokes equations. Journal of Computational Physics, 489:112253, 2023. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2023.112253.

[16] A. Kiener, S. Langer, and P. Bekemeyer. Data-driven correction of coarse grid cfd simulations. Computers Fluids, 264:105971, 2023. ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2023.105971.

[17] Pedro Stefanin Volpiani, Jean-Baptiste Chapelier, Axel Schwöppe, Jens Jägersküpper, and Steeve Champagneux. Aircraft simulations using the new cfd software from onera, dlr, and airbus. Journal of Aircraft, 61(3):857–869, 2024. DOI: 10.2514/1.C037506.

[18] Stefan Langer. An initial investigation of solving rans equations in combination with two-equation turbulence models. Technical report, Institut für Aerodynamik und Strömungstechnik, 09 2019.

[19] Francesco Bassi, Lorenzo Botti, Alessandro Colombo, and Stefano Rebay. Agglomeration based discontinuous galerkin discretization of the euler and navier–stokes equations. Computers & Fluids, 61:77–85, 2012. ISSN: 0045-7930. "High Fidelity Flow Simulations" Onera Scientific Day. DOI: https://doi.org/10.1016/j.compfluid.2011.11.002.

[20] Leo Breiman. Random forests. Machine Learning, 45:5–32, 10 2001. DOI: https://doi.org/10.1023/A:1010933404324.

[21] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations, 2017. DOI: https://doi.org/10.48550/arXiv.1609.02907.

[22] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019. DOI: https://doi.org/10.48550/arXiv.1907.10902.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011. DOI: https://doi.org/10.48550/arXiv.1201.0490.

[24] Philipp Bekemeyer, Anna Bertram, Derrick A. Hines Chaves, Mateus Dias Ribeiro, Andrea Garbo, Anna Kiener, Christian Sabater, Mario Stradtner, Simon Wassing, Markus Widhalm, Stefan Goertz, Florian Jaeckel, Robert Hoppe, and Nils Hoffmann. Data-Driven Aerodynamic Modeling Using the DLR SMARTy Toolbox. 2022. AIAA 2022-3899. DOI: 10.2514/6.2022-3899.

[25] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019. DOI: https://doi.org/10.48550/arXiv.1903.02428.