Université Paris Cité

Sorbonne Université

École doctorale [Nom et numéro identifiant]
Laboratoire

Insérer le titre de la thèse ici *(dans la langue de rédaction)*

Par Prénom NOM

Thèse de doctorat de spécialité

Dirigée par Prénom NOM
Et par Prénom NOM (en cas de co-direction et non de co-encadrement)
Présentée et soutenue publiquement/à huis clos le JJ/MM/AAAA

Devant un jury composé de :

Prénom NOM, Prof. HDR	Université de Namur, Belgique	Rapporteur
Prénom NOM, DR	Université de Ville	Rapportrice
Prénom NOM, CR-HDR	Université Paris Cité	Examinatrice
Prénom NOM, DR	Université de Y	Examinateur
Prénom NOM, MCF	Université Paris Cité	Examinatrice
Prénom NOM, PU-PH	Université Paris Cité	Membre invité
Prénom NOM, PhD	Entreprise X	Membre invité
Prénom NOM, MCF-HDR	Université Paris Cité	Directeur de thèse

Exemple de citation :
Comme lorsque la brume se dissipe,
le regard peu à peu trouve une figure
dans la vapeur dont les airs sont épaissis,
ainsi, perçant l'élément lourd et obscur
à mesure que nous approchions du bord,
l'erreur me quittait ...

L'Enfer – Chant XXXI, La Divine Comédie, Dante Alighieri

Contents

List of Figures

List of Tables

Acronyms

CFD Computational Fluid Dynamics. 4–6, 10, 15, 19, 20, 27, 32

CNN Convolutional Neural Networks. 5

DL Deep Learning. 4, 5, 10, 20

GDL Geometric Deep Learning. 4, 12, 13, 19, 21

GNN Graph Neural Networks. 5, 19

ML Machine Learning. 3, 4, 9, 16, 19, 20, 27, 30

MLP Multi-Layer Perceptron. 12–14, 19, 40

MSE Mean Squared Error. vii, 9, 11–15

NACA National Committee for Aeronautics. v, 6–8, 16, 19, 25–31, 35–39

NASA National Aeronautics and Space Administration. 5, 7, 29, 30, 35, 37

PDE Partial Differential Equations. 3–5

PINN Physics-Informed Neural Networks. 4, 5

RANS Reynolds-Averaged Navier–Stokes. 4–6, 8, 9, 11, 16, 24, 25, 29

RAS Reynolds-Averaged Simulation. 5, 6, 29

TMR Turbulence Modeling Resource. 5, 24, 35, 37

Part I

Datasets

Chapter 1

AirfRANS Dataset

Surrogate models are necessary to optimize meaningful quantities in physical dynamics as their recursive numerical resolutions are often prohibitively expensive. It is mainly the case for fluid dynamics and the resolution of Navier–Stokes equations. However, despite the fast-growing field of data-driven models for physical systems, reference datasets representing real-world phenomena are lacking. In this work, we develop AirfRANS, a dataset for studying the two-dimensional incompressible steady-state Reynolds-Averaged Navier–Stokes equations over airfoils at a subsonic regime and for different angles of attacks. We also introduce metrics on the stress forces at the surface of geometries and visualization of boundary layers to assess the capabilities of models to accurately predict the meaningful information of the problem. Finally, we propose deep learning baselines on four Machine Learning tasks to study AirfRANS under different constraints for generalization considerations: big and scarce data regime, Reynolds number, and angle of attack extrapolation.

1.1 Introduction

Numerical simulations of physical dynamics are a consequent part of scientific research as it allows us to quantitatively study natural phenomena without requiring often complex and expensive experiments. Those dynamics are mainly governed by Partial Differential Equations (PDE) and are numerically solved with the help of discretization methods such as finite differences, finite elements, or finite volumes methods. Such techniques are accurate when used over sufficiently fine meshes but are often expensive in time and resources. Thus, the optimization of meaningful quantities with respect to the parameters of the studied dynamics is, most of the time, out of scope. In particular, the numerical resolution of Navier–Stokes equations for fluid dynamics analysis leads to computations that can last for thousands of CPU hours. Hence, the design of accurate surrogate models is at the core of engineering as they allow us to tackle the task of optimization via data-driven approaches. However, to be able to compare and validate such surrogate models we need datasets of reference and evaluation protocols. For physical systems, some efforts have already been

done in this direction [1, 2] and our work is another contribution to those efforts. In [2], we developed the first version of this dataset to study Reynolds-Averaged Navier–Stokes (RANS) equations with Machine Learning (ML) models along with an appropriate evaluation protocol. In this paper, we propose an extension of this work by introducing a new high-fidelity version of the dataset. This high-fidelity version is built over finer meshes than the previous one which helps to fight numerical diffusion and allows us to recover more accurate fields and the trail of airfoils. Moreover, it allows us to accurately compute the force coefficients acting over geometries.

We focus on the classical aerodynamics task of predicting the steady-state two-dimensional fields and the force acting over airfoils in a subsonic regime. The ultimate goal of this task is to be able to find the best airfoil in terms of lift over drag ratio (see chapter 1 of [3]) in addition to the associated velocity and pressure fields. It is already a non-trivial problem in Computational Fluid Dynamics (CFD) as turbulence is involved and mesh engineering is required to find accurate force coefficients. To accelerate the resolution process, different ML frameworks can be used to build surrogate models [4, 5]. Deep Learning (DL) is among the successful candidates and has recently gained popularity for fluid simulation [6]. Moreover, the emerging field of Geometric Deep Learning (GDL) [7] models allows us to achieve learning directly on unstructured data [8] which, in this particular case, allows us to compute accurately meaningful quantities at the surface of geometries.

In this work, we present a high fidelity aerodynamics dataset of RANS solutions around airfoils. In Section 1.3 we present the RANS equations, the chosen design space for the airfoils generation, the meshes construction, and the dataset generation procedure. We also present the two force coefficients of interest, namely the drag and the lift coefficients. In Section 1.4 we introduce the different sub-tasks of the problem in addition to the evaluation protocol and the setup for our GDL baselines. In particular, the evaluation protocol contains metrics and visualizations for the force coefficient ranks and the accuracy of the surrogate models over boundary layers. We finally present, in Section 1.5 the results of our baselines on the different tasks. All the values of the constant used in this work and the definition of dimensionless quantities are given in Appendix 1.C.

1.2 Related Work

Although several research directions are established to come up with efficient surrogate models to tackle physics problems, from physically guided methods [9–13] to neural operators [14–17] and Physics-Informed Neural Networks (PINN) [18], the lack of standard benchmarking datasets, and common evaluation protocols impede making rigorous comparisons between the different families of methods for a given task. Benchmarking datasets and common evaluation protocols are shown to be the key components for making progress as it is observed in neighboring fields such as, for example, computer vision [19, 20] and speech recognition [21]. Though, few physics-based datasets have been proposed such as: 1D Burger's equation and 2D Darcy flow PDE [14], structural mechanics [8], incompressible fluid in vorticity form [16], reaction-diffusion, wave-equations and damped pendulum [22], heat transfer equation [23], Lorenz system [24]. More recently, few standard benchmarks

datasets on complex chemical and physical systems [25–29] have been proposed. More interestingly, [1] suggests a framework to study a set of representative physics problems with appropriate evaluation protocols, namely a single oscillating spring, a one-dimensional linear wave equation, a Navier–Stokes flow problem, as well as a mesh of damped springs. We follow those efforts by proposing a dataset on a steady-state aerodynamics task with dynamics that can be found in realistic flight scenarios. We also focus the validation of models on meaningful parts of the dynamic instead of only regarding the mean square loss of regressed fields.

Most of the works proposed in the literature to tackle tasks represented by Navier–Stokes equations are grid-based approaches [6, 30–35] which rely on Convolutional Neural Networks (CNN). Other architectures such as Fourier Neural Operator [16] act in the frequency domain and require a regular grid to perform a Fast Fourier Transform of the input data. Those models are not designed to directly operate on unstructured data like CFD meshes, resulting in inaccurate predictions of the physical fields at the surface of geometries. However, recent progress in learning on unstructured data [7] has enabled learning on graphs and manifolds by designing geometrical inductive bias in DL [36–40]. This framework is particularly useful to achieve learning on arbitrary shapes and frees us from the constraint of data voxelization as required by CNN. One successful attempt at learning Navier–Stokes (or RANS) equations with Graph Neural Networks (GNN) can be found in [8]. Finally, let us emphasize that PINN, as defined in [18] can act on unstructured data but are not suited for surrogate modeling as they are designed to solve one and only one PDE.

1.3 Dataset Presentation

Design-oriented dataset. This dataset is mainly motivated by a realistic shape optimization problem. We choose a classical aerodynamics problem for this purpose: airfoil design optimization. The goal is to accurately predict force coefficients in addition to the different fields of the fluid in a subsonic flight regime with a reduced quantity of data as is often the case in practice. The design space is chosen from the National Aeronautics and Space Administration (NASA) early works on airfoils via the 4 and 5 digits series [41] as they are easy to handle and already rich families of shapes.

We aim to resolve the air dynamic around a two-dimensional (2D) airfoil in a steady-state subsonic regime at sea level and $298.15\,\mathrm{K}$. More precisely, we study airflows at a Reynolds number between 2 and 6 million, which leads to turbulent behavior of the fluid. It corresponds to a Mach number smaller than 0.3 which allows us to assume incompressible flow behavior (see chapter 8 of [3]), and a velocity greater than $30\,\mathrm{m\,s^{-1}}$ which is a reasonable lower bound in subsonic flight conditions. Moreover, as the flow is turbulent in certain areas, we use Reynolds-Averaged Simulation (RAS) with a sufficiently high number of cells in our meshes to accurately compute the force acting over airfoils. This method solves the RANS equations widely used in CFD to control the numerical complexity of the resolutions.

We rely on the Turbulence Modeling Resource (TMR) of the Langley Research Center of the NASA [42–45] to generate our dataset and to check the accuracy of our simulation

setup with respect to experimental results (see Appendix 1.I). In what follows, we present the different steps to build the dataset, and we define the relevant physical quantities of the problem.

Reynolds-Averaged Navier–Stokes equations. At a high Reynolds number, untidy patterns emerge in fluid flows; we call this phenomenon turbulence. In CFD, turbulence resolution is a crucial problem as it implies transient simulations on prohibitively fine meshes most of the time. Different strategies have been developed to tackle this problem, one of them being RAS. In RAS, we solve mean-field equations similar to Navier–Stokes equations but with an effective viscosity representing the diffusion added through turbulent processes. Those equations, called incompressible RANS equations, are given by:

$$\partial_i \bar{u}_i = 0 \tag{1.1}$$

$$\partial_j(\bar{u}_i\bar{u}_j) = -\partial_i\left(\frac{\bar{p}}{\rho}\right) + (\nu + \nu_t)\partial^2_{jj}\bar{u}_i, \quad i \in \{1, 2\} \tag{1.2}$$

where $\bar{\cdot}$ denotes an ensemble-averaged quantity, ∂_i the partial derivative with respect to the i^{th} spatial components, u the fluid velocity, p an effective pressure, ρ the fluid specific mass, ν the fluid kinematic viscosity, ν_t the fluid kinematic turbulent viscosity and where we used the Einstein summation convention over repeated indices. Often, in the incompressible case, the effective pressure is replaced by the reduced pressure, abusively denoted by the same symbol via the transformation $p \to p/\rho$, which allows us to write RANS equations without explicit dependence on ρ. From now on, we will only discuss in terms of the reduced pressure. Finally, the dynamics of the turbulent viscosity is driven by a set of supplementary equations. In this work, we choose to use the well-known $k - \omega$ SST turbulence model [46] which is well suited for aerodynamics problems. Details on the RANS equations, the definition of the different quantities, the ensemble average, and the choice of the turbulent model are given in Appendix 1.D.

Airfoil design space. In the first half of the twentieth century, teams of the National Committee for Aeronautics (NACA) worked on several airfoil families. Two of them called the 4 and 5 digits series, are entirely parameterized and allow us to generate a broad spectrum of airfoils quickly. Both series define a camber line and an envelope around this camber line. An airfoil of the 4 digits one is defined by a sequence MPXX where M is the maximum ordinate of the camber line in hundredth of chords[1], P is the position of this maximum from the leading edge in tenth of chords and XX the maximum thickness in hundredth of chords. For the 5 digits series, each airfoil is defined by a sequence LPQXX. Digits L and P define in a more sophisticated manner than the 4 digits sequence the maximum camber of the camber line, Q is a boolean that switches between a single-cambered airfoil and a double-cambered one which allows in the latter case to achieve a theoretical pitching moment of 0. The last two digits XX have the same definition as in the 4 digits case.

Each simulation is first defined by an airfoil drawn in the 4 and 5 digits series families. The sampling strategy in those two series is given in Table 1.1. In our previous work [2],

[1] One chord is the characteristic length of the airfoil, in our case $1\,\mathrm{m}$

Table 1.1: Sampling strategy for generating airfoils from the 4 and 5 digits series. An interval or a discrete set means that the sampling is uniform over this set. In the 4 digits case, the sampling for P has been a uniform sampling on the interval $[0, 7]$, and all the samples smaller than 1.5 have been set to 0 to get rid of geometries that have their maximum camber too close from the leading edge.

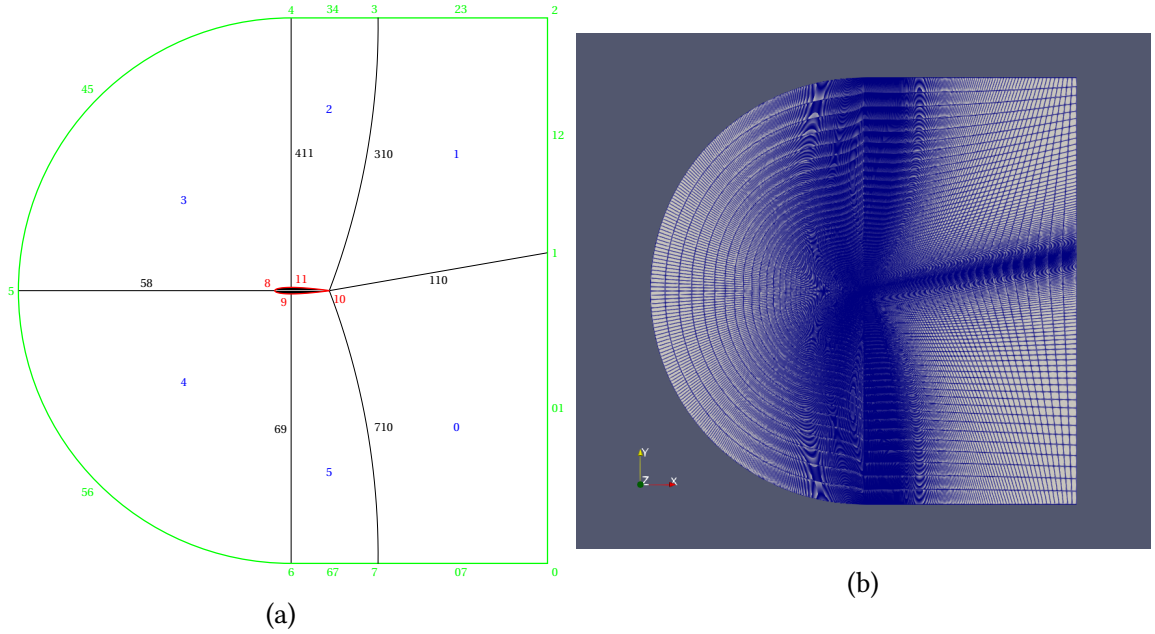
4-digits			5-digits			
M	P	XX	L	P	Q	XX
$[0, 7]$	$\{0\} \bigcup [1.5, 7]$	$[5, 20]$	$[0, 4]$	$[3, 8]$	$\{0, 1\}$	$[5, 20]$

we chose to use the UIUC Airfoil Database [47] to build the dataset but we decide here to restrict our airfoil design space to the NACA 4 and 5 digits series. Those series are already rich families of airfoils that have been widely used historically and they are easier to handle for the automation of the mesh generation due to their explicit parametrization. Moreover, in the 4 digits series, we choose to sample the parameter P between 0 and 7 and to set the drawn parameters in the interval $(0, 1.5]$ to 0. We motivate this choice as airfoils with P in the range $(0, 1.5]$ have their maximum camber close to the trailing edge which can lead to unusable airfoils.

Examples of different airfoils, details on the generation of such airfoils, and empirical statistics of the drawn parameters are given in Appendix 1.F.

Mesh generation. As airfoils are pretty simple geometries, we use the multi-block hexahedral mesh generator *blockMesh* from OpenFOAM v2112 [48] to mesh our shapes. We build a C-Grid mesh for each airfoil, mimicking the mesh developed by NASA for the NACA 0012 and 4412 cases [42]. Boundaries are at 200 chords of the airfoil to reduce the impact of boundary conditions on simulations. In Figure 1.1, we show the different block definitions with an example of a ready-to-use mesh and a final mesh on a classical airfoil. As we aim for accuracy in the computation of the global forces over the airfoil surface, such as the wall shear stresses, we mesh the boundary layer such that the first cells of the surface are of height $2\,\mu\mathrm{m}$ leading to a y^+ of around 1 in the worst case of our design space. This leads to meshes from $250\,000$ to $300\,000$ cells. All the technical details and definitions of the meshing procedure are given in Appendix 1.G.

Dataset generation. For the generation of the dataset, we run 1000 simulations, each defined by an airfoil, a Reynolds number, and an angle of attack. We choose to only run 1000 simulations as one of the goals of this dataset is to be close to real-world settings, *i.e.* limited quantity of data. The airfoil is sampled from the distribution given in Table 1.1. We motivate the design space of the initial conditions to reproduce the panel of flight conditions encountered in subsonic flights. We stop at a Mach number of 0.3 (Reynolds number of roughly 6 million) to keep the incompressible assumption valid and we start at a Reynolds number of 2 million as it is a correct lower bound of flight velocity (around 60 knots). The lower bound for angles of attack, $-5°$, is chosen such as cambered airfoils have a lift



(a) (b)

Figure 1.1: Example of mesh for the NACA 0012 at an angle of attack of $10°$. (a) Scheme of the multi-block mesh. Point number 1 moves following the angle of attack; all other points are fixed. The contour in red (airfoil) and green (freestream) are the domain's boundaries. (b) The entire domain of a ready-to-use mesh.

coefficient of roughly 0 and the upper bound, $15°$, is chosen to prevent stall and unsteady patterns in the trail of airfoils. Those ranges are tighter than the one chosen in our previous work [2] but better represent the classical ranges of velocity and angle of attack encountered in subsonic flight conditions. We then run the simulations with the help of the steady-state RANS solver *simpleFOAM* via the SIMPLEC algorithm [49, 50] and with the $k - \omega$ SST turbulence model [46] until convergence of drag and lift coefficients. Simulations are done on 16 CPU cores of an AMD Ryzen™ Threadripper™ 3960X. Figure 1.2 shows a near view of the pressure and x-component of the velocity fields. Boundary conditions types and values for each simulation are given in Appendix 1.H.

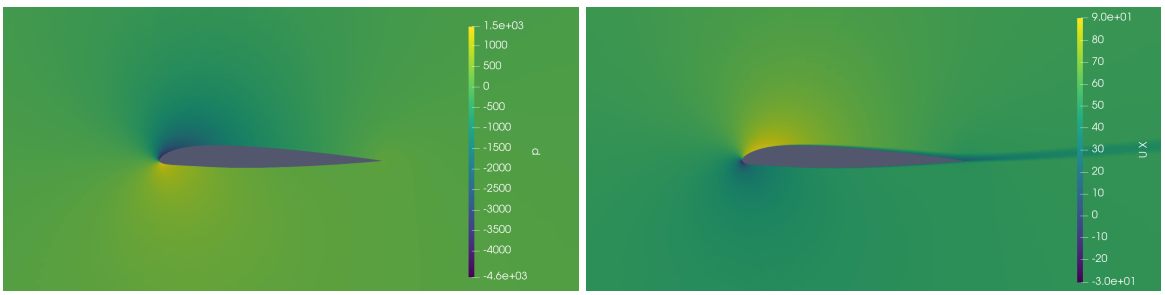


Figure 1.2: Example of a pressure (left) and x-component of the velocity (right) fields around a NACA (2.123, 3.832, 1, 9.902) at a velocity of $54.238\,\mathrm{m\,s^{-1}}$ and at an angle of attack of $7.911°$.

Force coefficients. One of the important quantities when simulating the fluid flow around a geometry is the force acting on it (see chapter 4 of [3]). This force is made by the contribution of the pressure and the viscous stresses at the surface of the geometry (called wall shear stress). The force component collinear to the free-stream velocity is called the drag D, and the one orthogonal to the free-stream velocity is the lift L. If divided by $q_\infty := \rho U_\infty^2 A/2$, where ρ is the fluid specific mass, U_∞ the inlet velocity and A the characteristic area of the geometry (in our case we take the chord as the 1D surface characteristic "area", *i.e.* $A = 1\,\mathrm{m}^2$), those components give the dimensionless drag coefficient $C_D := D/q_\infty$ and lift coefficient $C_L := L/q_\infty$. Other quantities such as the pitching moment can also be computed with force acting on the body, but we will only focus on the drag and lift coefficients in this work. To compute the wall shear stress, we need to compute the velocity gradient at the surface of the airfoil. We compute it discretely with the help of the *gradient filter* in ParaView [51] over the mesh.

1.4 Benchmarking Setup

The ML task consists in predicting the different spatial fields such as the mean-field velocity and reduced pressure $\bar{u}$ and $\bar{p}$ and the force acting over airfoils. The turbulent viscosity ν_t is not mandatory in the regression task as we do not need it to compute the force over an airfoil[2]. Still, it can give insights into the local intensity of turbulence in the volume. Regarding the force coefficients, and more precisely, the drag and lift coefficients, from a design process standpoint, we are more interested in the rank correlation with the true values than with the Mean Squared Error (MSE). Indeed, if the rank of the coefficients is accurately approximated, the optimization process will lead to the same airfoil. We can also add linear correlation plots for qualitative prediction information. Moreover, we are dealing in this dataset with hexahedral meshes with more than 250.000 cells which represent roughly the same number of nodes. These are already big meshes for 2D simulations but it is nothing compared to 3D cases where the number of cells can be of tens of million. To train a model on such simulations, we need to find a way to reduce the numerical complexity of the problem. Cropping close to the geometry is one way to do so, but as most cells lie in the vicinity of the geometry, it is not sufficient. Also, as we want to infer the force acting over the airfoil accurately, we cannot treat the cropped simulation as an image and work on a sub-sampled regular grid.

In this work, we choose the strategy which consists of regressing the four fields $\bar{u}_x$, $\bar{u}_y$, $\bar{p}$ and ν_t and compute the wall shear stress and the associated forces as post-processing to follow the form of RANS equations. We present a method to take into account the remarks mentioned above. Finally, we use the 1000 simulations to build four different setups:

- *Full data regime:* 800 samples for the training set and 200 for the test set
- *Scarce data regime:* 200 samples for the training set and the same test set as in the full data regime

[2] the boundary condition of the turbulent viscosity on the airfoil is 0 in our case

- *Reynolds extrapolation:* the training set is composed of the samples with a Reynolds of three to five million, and the test set is formed by the samples with a Reynolds of two to three and five to six million
- *Angle of attack extrapolation:* the training set is composed by the samples with an angle of attack between $-2.5°$ and $12.5°$ and the test set is composed by the samples with an angle of attack from $-5°$ to $-2.5°$ and $12.5°$ to $15°$.

Preprocessing. As we do not need the far-field to get rid of the boundary conditions impact on the simulation as in CFD, we crop all the simulations to a rectangle of size $[-2, 4]\times[-1.5, 1.5]$ meters. It allows us to limit our point clouds' size and make the network focus on the interesting part of the simulations. Moreover, data normalization is important in DL to make the optimization process easier or feasible. We use normalization with the means and the standard deviations of the training set field components.

Loss, sampling, and graph construction. At the end of the cropping procedure, we still have to deal with roughly 150.000 cells in each simulation giving about the same number of nodes. To train a model on such simulations, we need to find a way to reduce the numerical complexity even more.

To handle this numerical complexity, at each epoch, we choose to sample uniformly on the cropped mesh 32.000 nodes and, when necessary, reconstruct a radius graph of radii $5\,\mathrm{cm}$ with a maximum number of neighbors of 64[3]. This approach has several advantages, it allows us to directly control the numerical complexity with the number of sub-sampled nodes, the radii of the graph, and the maximum number of neighbors inside it. Moreover, during the inference, it is straightforward to infer fields on each node of the initial mesh by making multiple forward passes with different sub-sampling until every node has been seen and averaging the outputs on the nodes that have been seen multiple times. It allows us to compute the velocity gradient on the airfoil with the help of the initial mesh and ParaView's pythonic interface PyVista [52] to be able to compare with the surface force targets. However, one downside effect of the method is that the mesh densities bias the learning procedure, and the learned models may not generalize well to different types of meshes. In this dataset, we are not facing this problem as all the meshes are generated via the same procedure. Additionally, we can not sample independently on the airfoil and on the volume to bias models to be more accurate on the airfoil as the force highly depends on the pressure field at the surface. Finally, The loss $\mathcal{L}$ used in this work is the sum of two terms, a loss over the volume $\mathcal{L}_\mathcal{V}$ and a loss over the surface $\mathcal{L}_\mathcal{S}$:

$$\mathcal{L} := \underbrace{\frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} \|f_\theta(x_i) - y_i\|_2^2}_{\mathcal{L}_\mathcal{V}} + \lambda \underbrace{\frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}} \|f_\theta(x_i) - y_i\|_2^2}_{\mathcal{L}_\mathcal{S}} \tag{1.3}$$

where $\mathcal{V}$, $\mathcal{S}$ are respectively the set of the indices of the nodes that lie in the volume and on the airfoil, $x_i \in \mathbb{R}^7$ is the input at node i containing the spatial coordinates, the inlet

[3] which means that we connect nodes only very locally compared to the characteristic length of $1\,\mathrm{m}$ of airfoils

velocity, the Euclidean distance function between the node and the airfoil, and the unit surface outward-pointing normal for points on the airfoil (filled with zeroes otherwise). The targets $y_i \in \mathbb{R}^4$ at node i contain the velocity, the pressure, and the turbulent kinematic viscosity at this node. And f_θ represents the model used. The coefficient λ is used to balance the weight of the error at the surface of the geometry and over the volume[4]. We have to emphasize that this loss is not necessarily a good proxy when it comes, for instance, to infer the wall shear stress accurately or ensuring that the inferred fields satisfy the RANS equations.

Metrics and visualizations. One of the challenges of this dataset is to build models that manage to predict the form of the boundary layer accurately. To evaluate the performance of the models, we define qualitative and quantitative metrics.

To qualitatively check this accuracy, we propose to plot the components of the velocity and the turbulent viscosity (if regressed) in the boundary layer of airfoils at different chord lengths. Also, we propose to check the accuracy of the prediction for the pressure and skin friction coefficients on the airfoil as they carry important information on the wing's behavior in flight conditions. Finally, we propose to plot the predicted force coefficients with respect to the true coefficients which gives us qualitative information about the correlation between both variables.

In terms of quantitative metrics, we use the MSE for each field on the volume and over the airfoil to measure the accuracy of our models. Moreover, we compute the mean and standard deviation of the relative error on the drag and lift coefficients. Finally, we compute Spearman's rank correlation coefficient between the true and predicted force coefficients. From a design process point of view, the last coefficient is the most crucial quantity to maximize as it quantifies the monotonic correlation between the true and predicted force coefficients. If this coefficient is close to 1, we can expect our model to be able to find the best airfoil maximizing or minimizing the chosen force coefficient even if the inferred value is not close to the true value.

Metrics hierarchy. From a design-oriented perspective we may set a hierarchy for the proposed metrics. The Spearman's correlation for the force coefficients is the main metrics to maximize the recovery of the best airfoils in terms of lift-over-drag ratio as it quantifies the ability of models to preserve the force coefficient ranks. In addition to this metric, the plots of the predicted with respect to true force coefficients give qualitative information on the accuracy of the model for each simulation of the test set. Then, the relative errors for the force coefficients are important but not crucial from a design perspective and their minimization is secondary. We may say that a model is effective if it has Spearman's correlations close to one and accurate if it has low relative errors and low MSE on the fields over the volume and the airfoil. The goal here is an effective and accurate model. For relative errors, the bound of $5\,\%$ is often used to state that a model is accurate enough.

[4]In this work λ is set to 1.

1.5 Benchmarking Results

To propose baselines for the problem, we train three standard GDL models and a Multi-Layer Perceptron (MLP) for each data regime. Each model is preceded by an encoder and followed by a decoder, both defined by a MLP and trained together with the model. We follow the setup defined in the previous section for the training and testing procedures. Each model is trained 5 times to compute a mean and a standard deviation for the different metrics. For each metric, we bold the best-performing method. We choose as baselines a GraphSAGE [53], a PointNet [54], a Graph U-Net [55] and a MLP. Those baselines have been chosen as they access different types of information. The MLP only has access to the features of the nodes, the GraphSAGE has in addition access to local neighborhood information, the PointNet conditioned a deep MLP with global features, and the Graph U-Net access from local to global neighborhood information via its multi-scale architecture. Models are trained in the same conditions and the details of architectures and hyperparameters can be found in Appendix 1.J.

In Table 1.2, we give the MSE over the volume and at the surface of airfoils for the different regressed fields. In Table 1.3 we give the mean relative errors on the force coefficient and the Spearman's rank correlation coefficient. In Table 1.4 we compare the computational time to run a simulation, train a model and infer on a new example. In Figure 1.3 we plot the predicted force coefficients with respect to the true coefficients in the full data regime. Plots of the velocity and turbulent viscosity profiles in the boundary layer and surface coefficients for randomly chosen test geometries are given in Figure 1.4 also in the full data regime. We do not give any plot for the other regimes as those information are more qualitative and serve here as example more than real evaluation of the difficulty of a task or the performance of a model.

Tasks complexity. Each of the scores in Table 1.2 and Table 1.3 are given on the respective test set of the task. This makes difficult the direct comparison as the test set for the Reynolds and AoA extrapolation regime are both different from the test set used for the full and scarce data scenario.

Comparing the MSE scores of the different models on the interpolation tasks, we immediately see that every models better perform when more data is available, as expected. However, for the scores on the force coefficients and their rank correlation, we interestingly note that, even though the MSE scores in the scarce data regime are significantly lower on the surface pressure, the relative error and the Spearman's correlation on the lift coefficient are quasi identical or even better in the scarce data scenario. This may be understood as the computations of the force coefficients involve an integration over the surface of certain fields (see Appendix 1.E) and can lead to the accumulation or compensation of local errors. Concerning the drag coefficient, little can be said as the scores are particularly bad in all of the scenarios, models do not manage to give a decent prediction for this quantity.

For the extrapolation task, it is harder to compare the complexity of the Reynolds extrapolation regime with the AoA one but we can still highlight specificity in each one. We note that for the x-component of the velocity, where models are mainly asked to extrap-

Table 1.2: Comparison of the MSE on the normalized fields for an MLP and standard GDL baselines on the different task for the associated test set. Only the reduced pressure is given on the surface as the other quantities are null via the boundary conditions. Those quantities are directly regressed by the models. The field denoted by $\bar{p}_s$ is the mean field reduced pressure at the surface of airfoils.

Field	Model	Task			
		Full	Scarce	Reynolds	AoA
$\bar{u}_x$ ($\times 10^{-2}$)	MLP	1.63 ± 0.19	2.32 ± 0.15	12.4 ± 2.1	8.67 ± 2.35
	GraphSAGE	$\mathbf{1.58 \pm 0.16}$	$\mathbf{1.87 \pm 0.14}$	11.9 ± 2.6	$\mathbf{5.64 \pm 1.00}$
	PointNet	6.63 ± 1.10	7.52 ± 1.55	13.2 ± 2.8	16.4 ± 4.94
	GUNet	2.81 ± 0.34	2.65 ± 0.19	$\mathbf{10.2 \pm 1.0}$	7.13 ± 1.29
$\bar{u}_y$ ($\times 10^{-2}$)	MLP	$\mathbf{1.09 \pm 0.38}$	$\mathbf{1.81 \pm 0.17}$	6.73 ± 0.31	9.82 ± 3.86
	GraphSAGE	1.41 ± 0.24	1.87 ± 0.19	6.68 ± 1.97	$\mathbf{8.88 \pm 2.51}$
	PointNet	6.01 ± 1.13	6.29 ± 0.80	11.3 ± 6.0	23.3 ± 4.9
	GUNet	2.95 ± 0.40	2.67 ± 0.21	$\mathbf{5.72 \pm 0.51}$	9.68 ± 2.37
$\bar{p}$ ($\times 10^{-2}$)	MLP	0.81 ± 0.06	4.25 ± 0.43	6.69 ± 1.48	13.1 ± 3.0
	GraphSAGE	0.87 ± 0.12	4.85 ± 0.25	7.98 ± 5.23	11.9 ± 5.1
	PointNet	2.53 ± 0.50	7.51 ± 3.13	8.38 ± 4.93	20.2 ± 4.0
	GUNet	$\mathbf{0.76 \pm 0.06}$	$\mathbf{2.88 \pm 0.46}$	$\mathbf{5.38 \pm 1.28}$	$\mathbf{9.79 \pm 1.79}$
ν_t ($\times 10^{-2}$)	MLP	2.59 ± 0.19	6.20 ± 0.95	13.8 ± 3.5	48.4 ± 4.1
	GraphSAGE	2.11 ± 0.10	$\mathbf{5.18 \pm 0.78}$	15.1 ± 4.8	$\mathbf{47.1 \pm 4.5}$
	PointNet	5.26 ± 1.89	7.16 ± 2.69	13.8 ± 9.8	51.6 ± 3.4
	GUNet	$\mathbf{1.78 \pm 0.09}$	5.38 ± 1.26	$\mathbf{10.2 \pm 3.4}$	50.2 ± 1.5
$\bar{p}_s$ ($\times 10^{-2}$)	MLP	2.00 ± 0.41	12.1 ± 1.0	$\mathbf{11.5 \pm 2.2}$	33.1 ± 10.0
	GraphSAGE	1.84 ± 0.37	14.0 ± 0.7	12.5 ± 4.2	$\mathbf{23.6 \pm 9.7}$
	PointNet	9.96 ± 5.24	18.6 ± 4.4	14.0 ± 3.2	46.5 ± 9.0
	GUNet	$\mathbf{1.44 \pm 0.19}$	$\mathbf{8.41 \pm 1.04}$	11.7 ± 2.5	23.9 ± 6.1

Table 1.3: Comparison of the relative errors (Spearman's rank correlation) for the predicted drag coefficient C_D (ρ_D) and lift coefficient C_L (ρ_L) on the four different task for the associated test set. We want the Spearman's correlation to be close to one. Those quantities are computed as a post processing from the unnormalized regressed fields.

Field	Model	Task			
		Full	Scarce	Reynolds	AoA
C_D	MLP	**6.178 ± 0.900**	**4.540 ± 0.256**	**8.293 ± 3.049**	**4.355 ± 1.128**
	GraphSAGE	7.366 ± 1.212	4.587 ± 0.396	12.794 ± 3.980	6.047 ± 1.304
	PointNet	17.392 ± 1.373	16.048 ± 1.928	17.111 ± 2.684	13.846 ± 4.316
	GUNet	13.320 ± 0.924	10.726 ± 1.154	18.103 ± 0.675	9.814 ± 1.281
C_L	MLP	0.211 ± 0.028	0.199 ± 0.031	0.621 ± 0.191	0.413 ± 0.096
	GraphSAGE	**0.148 ± 0.026**	**0.150 ± 0.011**	0.433 ± 0.060	**0.254 ± 0.052**
	PointNet	0.197 ± 0.028	0.200 ± 0.047	**0.384 ± 0.040**	0.442 ± 0.107
	GUNet	0.168 ± 0.023	0.150 ± 0.013	0.466 ± 0.118	0.376 ± 0.047
ρ_D	MLP	**0.250 ± 0.094**	0.248 ± 0.064	0.157 ± 0.143	0.347 ± 0.222
	GraphSAGE	0.194 ± 0.067	**0.254 ± 0.071**	0.039 ± 0.046	0.525 ± 0.088
	PointNet	0.074 ± 0.063	0.048 ± 0.077	0.115 ± 0.148	0.089 ± 0.278
	GUNet	0.092 ± 0.052	0.074 ± 0.021	**0.192 ± 0.130**	**0.552 ± 0.056**
ρ_L	MLP	0.993 ± 0.002	0.993 ± 0.002	0.958 ± 0.022	0.957 ± 0.036
	GraphSAGE	**0.996 ± 0.001**	**0.996 ± 0.001**	0.971 ± 0.018	**0.989 ± 0.002**
	PointNet	0.992 ± 0.002	0.992 ± 0.002	**0.981 ± 0.006**	0.978 ± 0.003
	GUNet	0.995 ± 0.001	0.994 ± 0.0003	0.964 ± 0.016	0.982 ± 0.007

olate in the Reynolds regime, MSE are significantly higher underlying the particular difficulty of this scenario. The same remark can be made for the AoA extrapolation regime, the y-component of the velocity and the pressure in the volume and on the surface of airfoils where high variations are the most expected. For the turbulent viscosity, it is interesting to note that every model have difficulties of extrapolating on unseen scenario especially for new angles of attack. Even though this quantity is not mandatory to regress when only interested in the force coefficients, we still note that it looks like a less regular and predictable quantity for those models in every tasks.

Models performance. In interpolation tasks, local models (GraphSAGE and MLP) seem to better perform for the regression of the velocity whereas the multiscale Graph U-Net architecture seems to better perform for the pressure and the turbulent viscosity. The PointNet does not manage to achieve similar accuracy as the other models underlying the non necessarily helping of the global conditioning. The same conclusion can be drawn from the relative error and the Spearman's correlation scores. In total, the GraphSAGE model seems to be a good trade-off, in this setting, between complexity and performance as it achieves almost equivalent performance with the Graph U-Net, is almost twenty times faster to call,

Table 1.4: Running time for one simulation on 16 CPU cores of an AMD Ryzen™ Threadripper™ 3960X compared to training and inference time of the different models on an NVIDIA GEFORCE RTX 3090. The inference time is given for one call of a model on a batch of 32000 nodes, for one simulation we need around a hundred calls to get a result on the entire mesh as the nodes are chosen randomly on the CFD mesh. The number of parameters for each model is given as additional information.

Model	Running time		# Parameters
	Training	Inference (µs)	
MLP	$\sim$2 h20 min	13.3 $\pm$ 0.2	19988
GraphSAGE	$\sim$4 h20 min	20.9 $\pm$ 2.3	29204
PointNet	$\sim$2 h40 min	33.9 $\pm$ 3.5	75244
Graph U-Net	$\sim$6 h50 min	357.8 $\pm$ 36.9	65820
Simulation	$\sim$25 min		
Dataset	$\sim$20 days		

and has half of the number of parameters of the Graph U-Net.

For the extrapolation tasks, the Graph U-Net seems to be the choice of reference reaching the lowest MSE or close to the lowest MSE for every regressed fields. However, those scores are still too high to correlate with good performance on the integrated quantities.

From the plots of the different examples of boundary layers and skin friction coefficients given in Figure 1.4 and Figure 1.5 in the full data scenario, we conclude that models have difficulties to predict the wall shear stresses as the velocity values at the closest nodes from the geometry are often largely overestimated. This particularly affects the accuracy of the drag coefficient as we can note with the Spearman's correlation ρ_D, the relative error C_D, and the plot of the predicted drag with respect to the true drag coefficients (see Figure 1.3 left). However, the wall shear stress has a small impact on the lift coefficient compared to the pressure at the surface of airfoils. Hence, as the pressure is more accurately inferred compared to the wall shear stress, as we can see by looking at the plots of the pressure coefficient at the surface of airfoils in Figure 1.5 (left), the lift coefficient is also more accurately inferred and the rank is better predicted (see Figure 1.3 right) leading to a Spearman's correlation close to one.

Finally, in Table 1.4, we confirm that even for a two-dimensional case, the training cost of models is rapidly amortized (after, in the worst case, a dozen of simulations).

1.6 The airfrans library

As MSE, or variants of it such as mean absolute error or relative version of them, are often used as a single metrics for a given regression problem and that the manipulation of simulations under the *.vtu* extension can be new to users of the dataset, we developed a library

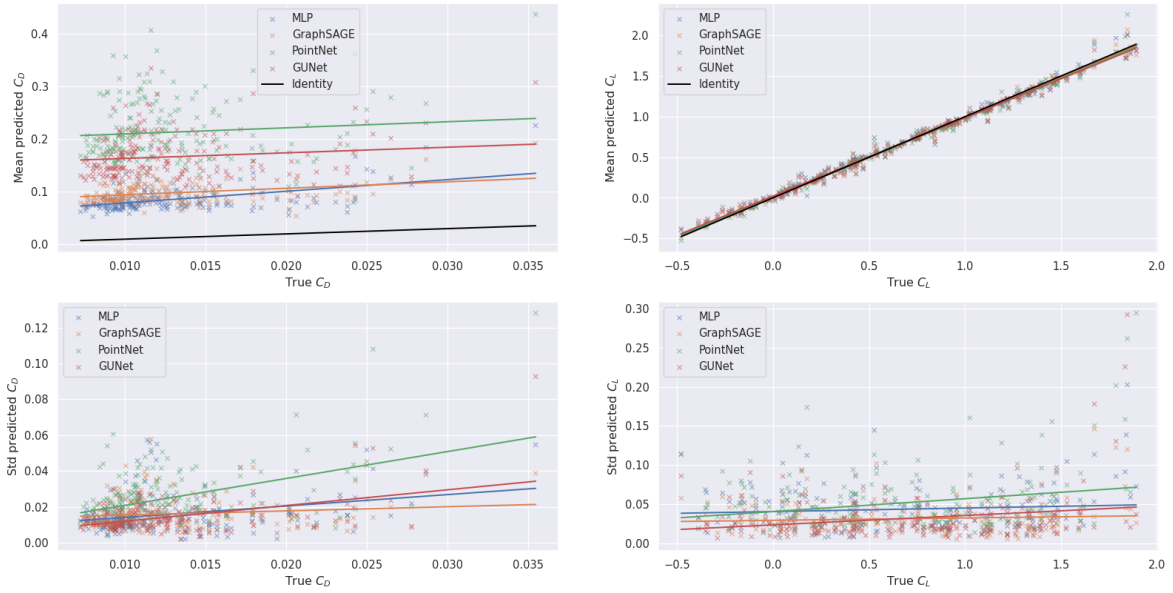


Figure 1.3: Predicted drag (left) and lift (right) coefficients with respect to the true ones in the full data regime. The mean (top) and standard deviation (bottom) of each point on the five copy of the trained models are separated for sake of readability. A linear regression is done for each point cloud in order to highlight linear trends. On the top plots, the Identity graph is given in black for comparison.

to easily compute the previously given metrics and to generate meaningful plots of regions of interest.

This library is mainly built over PyVista [52] and allows users to blindly compute force coefficients, metrics and plot boundary layers profile and fields over airfoils' surface without requiring any knowledge on how to use ParaView [51] or compute derivatives with original meshes. It also includes methods to sample with respect to the mesh or a uniform distribution the volume and the surface of each simulation. The field values attributed to sampled points are given via barycentric interpolation after triangulation of original meshes.

We refer to the user documentation for more information.

1.7 Conclusion

In this work, we presented a high-fidelity dataset of solutions of the two-dimensional RANS equations around NACA airfoils. Simulations have been done at Reynolds of the order of magnitude of what we find in subsonic flight regimes mimicking classical aerodynamics setups. We defined four ML tasks highlighting the different challenges of surrogate models, from scarce data regimes to extrapolation. We proposed different metrics focusing not only on the velocity and pressure fields but also on the force coefficients. Those metrics quantify the ability of ML models to accurately predict fields and force coefficients in addition to their ability to rank the latter, for example, for shape optimization. Different baselines have been

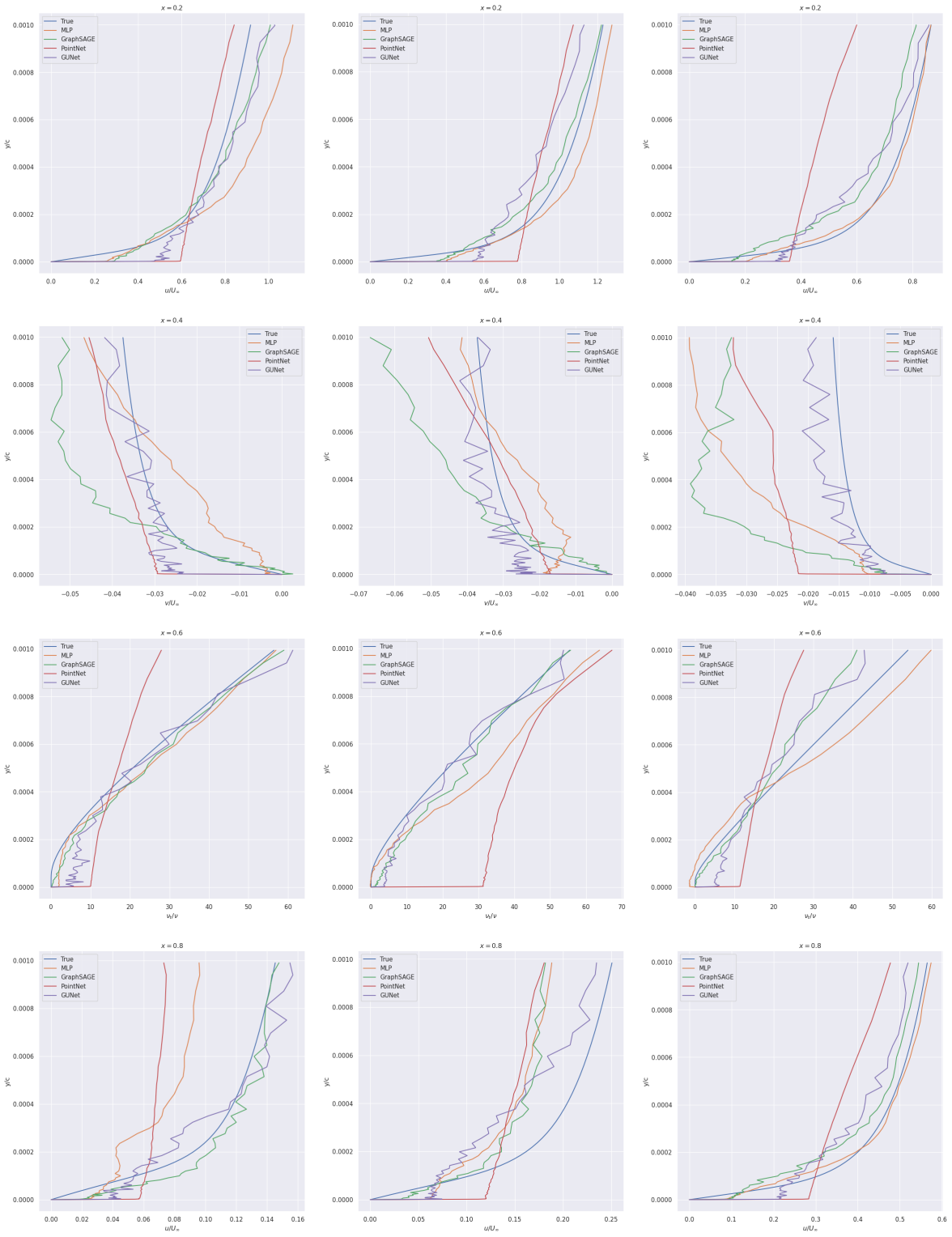


Figure 1.4: Comparison of the predicted boundary layers profiles on three random test geometries at different abscissas in the full data regime with respect to the true ones. Each column of plots represent a different airfoil and each line of plots represent a different abscissas. The x and y component of the velocity are denoted by u and v respectively and the turbulent viscosity is denoted by ν_t. Each quantity is normalized either by u_∞ the inlet velocity magnitude or ν the fluid viscosity.

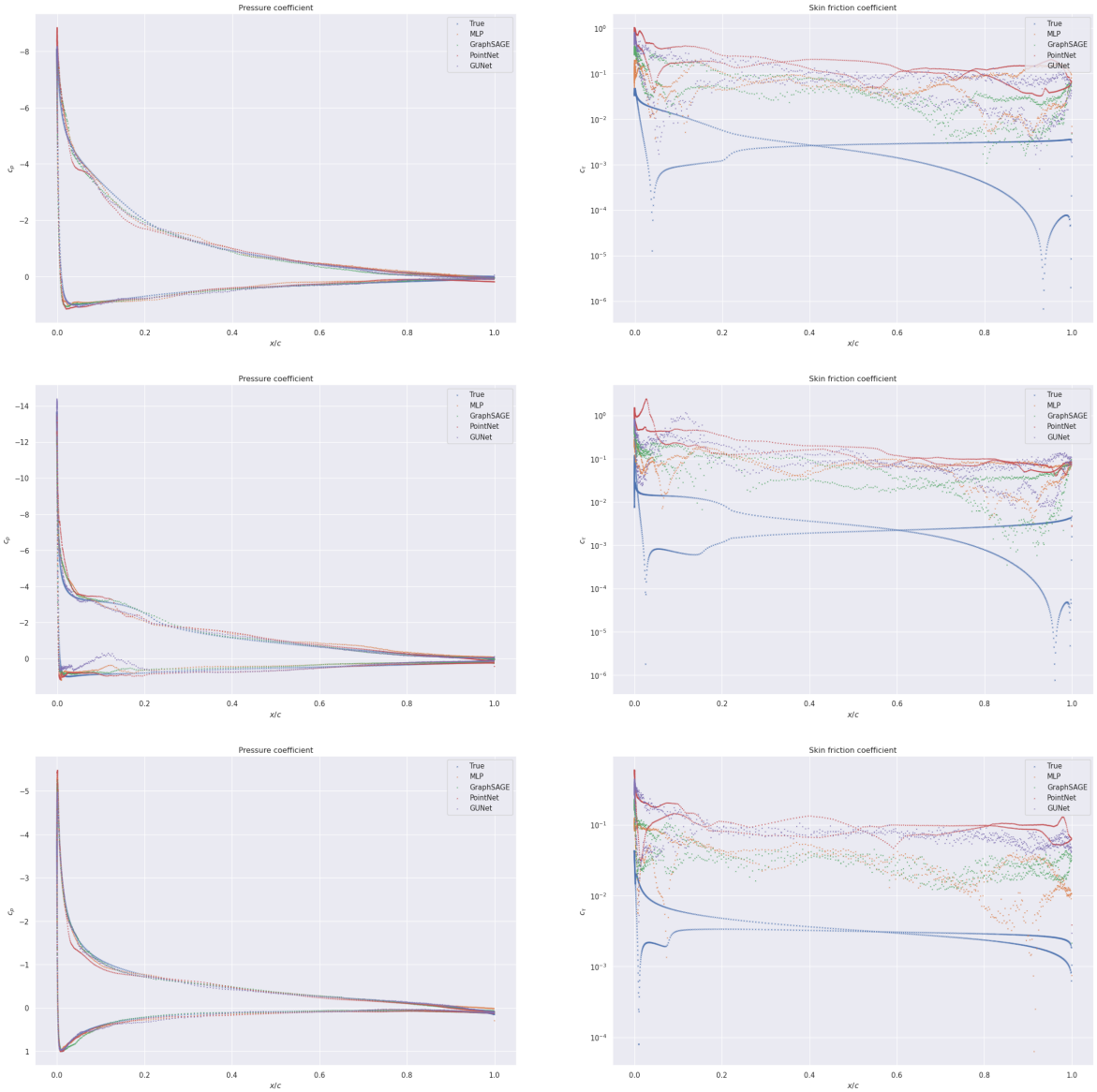


Figure 1.5: Comparison of the predicted surface coefficients profiles on three random test geometries in the full data regime with respect to the true one. (left) Surface coefficient c_p (right) Skin friction coefficient c_τ. Each line of plots represents a different airfoil. Skin friction coefficient plots are given in log scale.

introduced from the GDL framework, highlighting the need for models that can handle unstructured point clouds in order to be able to accurately predict force coefficients. Those baselines have shown in the proposed setting, as expected, that the prediction of the drag coefficient is more challenging than the prediction of the lift coefficient as the wall shear stress is derived from the velocity field and not directly regressed like the pressure.

Limitations. Concerning the dataset itself, we restricted the design space of airfoils to NACA 4 and 5 digits for the sake of simplicity and meshing automation but we can expect that models trained on this dataset will have difficulties generalizing to more exotic shapes. Following this, we did not propose an extrapolation test on out-of-distribution airfoils. Also, even though the problem proposed is a classical aerodynamics one, it is two-dimensional and does not reflect entirely the complexity of three-dimensional natural phenomena which implies that models working on this dataset could not necessarily be extended to more generic three-dimensional cases.

In terms of baselines, we proposed four architectures of different types, an MLP, a GNN, a network acting on point clouds, and a multi-scale GNN architecture. The GNN approach suffers from its heaviness when dealing with large graphs leading to the necessity of building a downsampling strategy. In addition to that, auto differentiation with respect to positions is not possible with GNN as an entire graph is given in inputs. This leads to difficulties in the inference process and the need to rely on input CFD meshes to compute derivatives using numerical schemes. On the other hand, the MLP approach allows more flexibility and does not require a subsampling strategy or the input CFD mesh to compute derivatives. Uniform sampling is also possible in this case and several models have already shown their ability to fit complex signals [56, 57]. The main downside of such approaches is the generalization capacity requiring conditioning or hypernetworks to work on multiple examples and generalize to unseen ones. Moreover, let us mention equivariant networks [58] as another promising direction to handle the lack of data often encountered in such tasks by leveraging symmetries of the problem. Finally, neural operators handling unstructured data such as DeepONet [17] are by definition well suited for such tasks and could be mixed with previous techniques to achieve high-performance surrogate models.

The loss function used in this work has been chosen to put more weight on surface fields but could be designed to better weight point close to the surface as we know that they are crucial for accurately predicting drag coefficients. This has to be designed in tandem with the sampling of volume and surface points that also induce a bias in the optimization process.

In total, we consider this work as a first step towards the generic treatment of real-world physical phenomena in ML. We hope that it will lower the potential barrier for entering the field of ML applied to physical systems and that it will encourage the construction of models that are not only good on the predicted fields but also on meaningful derived quantities.

1.A Reproducibility statement

We provide a GitHub repository to reproduce all the experiments and a link to download the preprocessed dataset and another one for the raw OpenFOAM data. The experiments have been conducted with a single NVIDIA RTX 3090 24Go. The repository include code to reproduce the Machine Learning (ML) experiments and code to generate the figures.

We also provide a GitHub repository to run new simulations and to be able to reproduce the generation settings of the dataset. The simulations have been done with 16 CPU cores of an AMD Ryzen™ Threadripper™ 3960X. The codes in the repository include (extensible) code to generate the meshes and code to run new simulations and/or build the dataset.

Finally, we provide the AirfRANS Python library with its associated GitHub repository to easily manipulate simulations from the dataset.

1.B Description Of Software

In this section, we describe the tools that we have used in this work to build the dataset[5], make the visualizations, and train the models. This work makes use of Computational Fluid Dynamics (CFD) and ML tools.

OpenFOAM [48] stands for *Open-source Field Operation And Manipulation*, a C++ software for developing custom numerical solvers to study continuum mechanics and CFD problems. In this work, we have used version v2112 of OpenFOAM to make our simulations. OpenFOAM is released as free and open-source software under the *GNU General Public Licence*.

ParaView [51] is an open-source visualization tool designed to explore and visualize efficiently large data using quantitative and qualitative metrics. ParaView runs on distributed and shared memory parallel and single processor systems. In this work, we have used it to visualize the following: point clouds, meshes, the predicted (as well as the ground truth) physical fields. We have used version 5.10.0 of ParaView in this work. ParaView is released as free and open-source software under the *Berkeley Software Distribution License*.

PyVista [52] is an open-source tool based on a handy interface for the Visualization ToolKit (VTK). It is simple to use in interaction with NumPy [59] and other Python libraries. It is mainly used for mesh analysis. In this work, we use PyVista to build the inputs of our DL models. We have used version 0.36.1 of PyVista in this work. PyVista is released as free and open-source software under the *MIT License*.

PyTorch [60] is an open-source library for DL using GPUs and CPUs. In this work, we use PyTorch to build our training protocol. In this work, we have used version 1.11.0 of Pytorch along with CUDA 11.3 and Python 3.9.12. PyTorch is released as free and open-source *Berkeley Software Distribution License*.

[5] similar to our first version of our dataset [2]

Table 1.C.1: Properties of air at $298.15\,\mathrm{K}$ ($25\,^{\circ}\mathrm{C}$) and at sea level on earth.

Name	Symbol	Value
Kinematic viscosity	ν	$1.56 \times 10^{-5}\,\mathrm{m^2\,s^{-1}}$
Specific mass*	ρ	$1.184\,\mathrm{kg\,m^{-3}}$
Thermal diffusivity*	α	$2.25 \times 10^{-5}\,\mathrm{m\,s^{-1}}$
Specific heat*	c_p	$1005\,\mathrm{J\,kg^{-1}\,K^{-1}}$
Atmospheric pressure*	p_0	$1.013 \times 10^{5}\,\mathrm{N\,m^{-2}}$
Atmospheric temperature*	T_0	$298.15\,\mathrm{K}$
Speed of sound in the fluid*	c	$346.1\,\mathrm{m\,s^{-1}}$

* Those values are important only in the compressible case, they are given for comparison with compressible simulations. Especially, the absolute pressure is set to $0\,\mathrm{N\,m^{-2}}$ for the incompressible case as it only depends on the differential pressure.

PyTorch Geometric (PyG) [61] is an open-source library for GDL built upon PyTorch which targets the training of geometric neural networks, including point clouds, graphs and meshes. We use PyG to design our message passing schemes. In this work, we have used version 2.0.4 of PyG along with CUDA 11.3. PyG is released as free and open-source software under the *MIT License*.

1.C Constant and Dimensionless Quantities

The fluid used in this study is the air at $298.15\,\mathrm{K}$ ($25\,^{\circ}\mathrm{C}$) and at sea level on earth. In Table 1.C.1 we give the different values of the constant associated with this fluid.

The only dimensionless quantity for the incompressible case is the Reynolds number Re. We add the Mach number Ma and the Prandtl number Pr in the compressible case. Those quantities are defined by:

$$Re = \frac{UL}{\nu}, \quad Ma = \frac{U}{c}, \quad Pr = \frac{\nu}{\alpha} \tag{1.4}$$

where U is the characteristic velocity of the problem, L its characteristic length, c the speed of sound in the fluid, ν its kinematic viscosity and α its thermal diffusivity. The Reynolds number compares the order of magnitude of the convective term with respect to the diffusive term in the Navier–Stokes equations, the Mach number quantifies flow compressibility and the Prandtl number compares the order of magnitude of the variation of energy via momentum with respect to the variation of energy via heat transfer in the compressible form of Navier–Stokes equations.

1.D Reynolds-Averaged Navier–Stokes Equations

Under certain assumptions, the dynamics of a fluid is governed by the Navier–Stokes equations. Those equations are composed of a continuity equation, three momentum equations and an energy equation (see §49 of [62] and §5.3 of [63]):

$$\partial_t \rho + \partial_i(\rho u_i) = 0 \tag{1.5}$$

$$\partial_t(\rho u_i) + \partial_j(\rho u_j u_i) = -\partial_i p + \partial_j \sigma_{ji}, \quad i \in \{1,2,3\} \tag{1.6}$$

$$\partial_t \left(\rho \left(\epsilon + \frac{1}{2}u^2\right)\right) + \partial_j \left(\rho u_j \left(h + \frac{1}{2}u^2\right)\right) = \partial_j(u_i \sigma_{ij}) - \partial_j q_j \tag{1.7}$$

where ∂_t denotes a partial derivative with respect to time, ∂_i a partial derivative with respect to the i^{th} space coordinate, ρ the fluid specific mass, u the fluid velocity, p the fluid pressure, σ the viscous stress tensor, ϵ the fluid specific energy, h the fluid specific enthalpy ($h := \epsilon + p/\rho$) and q the heat flux density due to thermal conduction. Moreover, we used the Einstein summation convention over repeated indices. Finally, fluid dynamics theory, thermodynamic relations, Fourier law and the perfect gas law give us:

$$\sigma_{ij} = \mu \left(\partial_i u_j + \partial_j u_i - \frac{2}{3}\partial_k u_k \delta_{ij}\right), \quad i,\, j \in \{1,2,3\} \tag{1.8}$$

$$\epsilon = c_v T \tag{1.9}$$

$$h = c_p T \tag{1.10}$$

$$q_i = -\kappa \partial_i T, \quad i \in \{1,2,3\} \tag{1.11}$$

$$p = \rho R T \tag{1.12}$$

where δ_{ij} is the kronecker tensor, T the fluid temperature, μ the fluid dynamic viscosity (function of T), κ the fluid thermal conductivity (function of T), R the fluid specific constant, c_v and c_p the fluid specific heat coefficient for constant volume and pressure respectively (taken constant here). This leads to a close set of partial differential equations with 6 unknowns (ρ, p, u, T) and 6 equations:

$$\partial_t \rho + \partial_i(\rho u_i) = 0 \tag{1.13}$$

$$\partial_t(\rho u_i) + \partial_j(\rho u_j u_i) = -\partial_i p + \partial_j \left(\mu \left(\partial_i u_j + \partial_j u_i - \frac{2}{3}\mu \partial_k u_k \delta_{ij}\right)\right),\ i \in \{1,2,3\} \tag{1.14}$$

$$\partial_t \left(\rho \left(c_v T + \frac{1}{2}u^2\right)\right) + \partial_j \left(\rho u_j \left(c_p T + \frac{1}{2}u^2\right)\right) = \partial_j \left(\mu u_i \left(\partial_i u_j + \partial_j u_i - \frac{2}{3}\partial_k u_k \delta_{ij}\right)\right) + \partial_j(\kappa \partial_j T) \tag{1.15}$$

together with the state equation 1.12. We chose the perfect gas law for the state equation as we are going to treat the case of air but this equation can be replaced by any state equation better suited for the problem.

In certain cases, we can decently assume that the fluid is incompressible with constant density ρ. We then need only 4 equations to close the problem. We get rid of the energy

and state equations and find the incompressible Navier–Stokes equations:

$$\partial_i u_i = 0 \tag{1.16}$$

$$\partial_t u_i + \partial_j (u_i u_j) = -\partial_i \left(\frac{p}{\rho}\right) + \nu \partial^2_{jj} u_i, \quad i \in \{1,2,3\} \tag{1.17}$$

where $\nu := \mu/\rho$ is the fluid kinematic viscosity, taken constant in this case. In order to explicitly write an important dimensionless quantity in fluid mechanics, we can rewrite last equations with dimensionless variables. Let us define, T, U, L and P characteristic time, velocity, length and pressure of the problem, respectively. We write:

$$t = T\hat{t}, \quad u = U\hat{u}, \quad x = L\hat{x}, \quad p = P\hat{p} \tag{1.18}$$

where x is the cartesian position, $\hat{t}$, $\hat{u}$, $\hat{x}$ and $\hat{p}$ are dimensionless quantities. If we write $P = \rho U^2$ and $T = L/U$, we find for the incompressible case:

$$\partial_{\hat{i}} \hat{u}_i = 0 \tag{1.19}$$

$$\partial_{\hat{t}} \hat{u}_i + \partial_{\hat{j}} (\hat{u}_i \hat{u}_j) = -\partial_{\hat{i}} \hat{p} + \frac{1}{Re} \partial^2_{\hat{j}\hat{j}} \hat{u}_i, \quad i \in \{1,2,3\} \tag{1.20}$$

where $Re := UL/\nu$ is the Reynolds number. This dimensionless number quantifies the importance of the convective term with respect to the diffusive term (in order of magnitude):

$$\frac{\|\partial_j (u_i u_j)\|}{\nu \|\partial^2_{jj} u_i\|} \approx \frac{UL}{\nu} = Re \tag{1.21}$$

When the Reynolds number tends to 0, diffusion term are dominant, we call it a Stokes flow. When the Reynolds number tends to ∞, the equations get closer to the Euler equations for inviscid fluid. At high Reynolds, new chaotic patterns can emerge close to walls and the different fields get untidy. This transition is the transition between what we call laminar (tidy) and turbulent (untidy) flows. Turbulence is a process that emerges at high Reynolds number and allows more dissipation than expected with laminar flows via the emergence of eddies of different length scales (see §33 of [62]). Theoretical resolution of such dynamics is an open problem and direct numerical simulations (DNS) are highly challenging because of their huge computational costs. Hence, different technologies have been developed in order to reduce the computational complexity of the task, for example, large eddy simulations (LES) try to filter in space the pressure and velocity fields and model the smallest eddies by adding dissipation. Another one, that we will use here, try to model all the scales of eddies by doing an ensemble average on the velocity and pressure fields. An ensemble average is a theoretical average over multiple equivalent experiments, this can also be equivalently replaced by a time averaging on a time scale big compared to turbulent fluctuations rate and small compared to the macroscopic evolution rate of the fluid. We write:

$$u = \bar{u} + u', \quad p = \bar{p} + p' \tag{1.22}$$

where $\bar{\cdot}$ denotes an ensemble averaged quantity and $\cdot'$ its fluctuations. If we set those expressions into the incompressible Navier–Stokes equations and take the ensemble averaging of the equations, we get:

$$\partial_i \bar{u}_i = 0 \tag{1.23}$$

$$\partial_t \bar{u}_i + \partial_j (\bar{u}_i \bar{u}_j) = -\partial_i \left(\frac{\bar{p}}{\rho}\right) + \nu \partial^2_{jj} \bar{u}_i - \frac{1}{\rho} \partial_j (\sigma_t)_{ij}, \quad i \in \{1,2,3\} \tag{1.24}$$

where $(\sigma_t)_{ij} := -\rho\overline{u_i' u_j'}$ is called the Reynolds stress tensor. We now have a new unknown in our equations and the problem is not close anymore. A common assumption known as the Boussinesq hypothesis is to write the Reynolds stress tensor in the same way as the viscous stress tensor:

$$(\sigma_t)_{ij} = \rho\nu_t \left(\partial_i \bar{u}_j + \partial_j \bar{u}_i\right) - \frac{2}{3}\rho k \tag{1.25}$$

$$k = \frac{1}{2}\overline{u_i' u_i'} \tag{1.26}$$

where ν_t is called the turbulent kinematic viscosity and k the specific kinematic energy of turbulence. The term $-2\rho k/3$ is set to ensure the null value of the trace of σ_t. By defining an effective pressure, abusively denoted by the same symbol, $\bar{p}$, $\bar{p} := \bar{p} + 2\rho k$, we find:

$$\partial_i \bar{u}_i = 0 \tag{1.27}$$

$$\partial_t \bar{u}_i + \partial_j(\bar{u}_i \bar{u}_j) = -\partial_i \left(\frac{\bar{p}}{\rho}\right) + \partial_j \left[(\nu + \nu_t)\partial_j \bar{u}_i\right], \quad i \in \{1, 2, 3\} \tag{1.28}$$

This set of equations is known as the Reynolds-Averaged Navier–Stokes (RANS) equations. In order to close our set of equations, we need a last equation for ν_t. Such equation is called a turbulence model and plenty of them have been developed in the last decades to recover experimental results in certain environments. We very briefly present two turbulence models that we are going to use in our experiments and let the details of those model in the references given. The Spalart-Allmaras model [64] is a one-equation model designed for aerodynamics problems, it involves a modified viscosity called $\tilde{\nu}$. The $k-\omega$ SST model [46] is the blending of two two-equations turbulence model, namely the $k-\varepsilon$ and the $k-\omega$ models [65, 66], and it extends the domain of application of both by switching models where it is more relevant to use one instead of another. It involves two quantities, the specific kinematic turbulent energy k and the specific turbulence dissipation rate ω.

In the compressible case, a mass-average is applied to the Navier–Stokes equations, for example in the case of the velocity, we use:

$$\tilde{u} = \frac{1}{\bar{\rho}}\overline{\rho u} \tag{1.29}$$

and we can decompose the velocity in a mass-averaged term and a fluctuation term:

$$u = \tilde{u} + u'' \tag{1.30}$$

By doing this for different quantities such as specific energy, specific enthalpy, temperature etc... we can write a new set of equations in a similar form as 1.5 - 1.7. This is called the Favre-Averaged Navier–Stokes equations, details can be found in chapter 5 of [63].

Finally, the RANS equations are the equations solved by the *simpleFoam* solver and the Favre-Averaged Navier–Stokes equations the ones solved by the *rhoSimpleFoam* solver in the OpenFOAM suite. We compare our results with compressible simulations and the results given in the TMR [42] in Appendix 1.I.

1.E Force Coefficients

The stress force df acting on a face of area dS and normal n is:

$$df = -pn + 2\mu S \cdot n \tag{1.31}$$

We can conclude that for a geometry of surface $\mathcal{S}$, the stress force F acting on it can be computed via:

$$F = \oint_{\mathcal{S}} \sigma \cdot n dS \tag{1.32}$$

$$= -\oint_{\mathcal{S}} p n dS + \oint_{\mathcal{S}} 2\mu S \cdot n dS \tag{1.33}$$

We call the term $P := -pn$ the wall pressure and the term $\tau := 2\mu S \cdot n$ the wall shear stress. Ultimately, we call drag D and lift L the component of F that are respectively parallel and orthogonal to the main direction of the flow. If $u_{\parallel}$ is the unit direction of the velocity vector and $u_{\perp}$ its orthogonal unit direction, we have:

$$D = \left(\oint_{\mathcal{S}} p dS + \oint_{\mathcal{S}} \tau dS \right) \cdot u_{\parallel} \tag{1.34}$$

$$L = \left(\oint_{\mathcal{S}} p dS + \oint_{\mathcal{S}} \tau dS \right) \cdot u_{\perp} \tag{1.35}$$

In the case of RANS equations, we add terms that take in account the effect of turbulence over the geometry. The pressure p is replaced by an effective mean-field pressure $\bar{p}$ and the wall shear stress is given by $\tau = 2(\mu + \mu_t) S \cdot n$ where μ_t is the dynamic turbulent viscosity. However, as the turbulent viscosity is null over the airfoil, we recover $\tau = 2\mu S \cdot n$.

For incompressible fluids we also often divide those quantities by ρ the density of the fluid and solvers often express the results in terms of reduced pressure $\bar{p} \to \bar{p}/\rho$ and kinematic (turbulent) viscosity $\nu := \mu/\rho$ ($\nu_t := \mu_t/\rho$). We use this convention in this work.

1.F Airfoil Generation and Statistics

In this section, we review the construction of the NACA 4 and 5 digits [41]. Both of them are built in the same manner and rely only on 3 or 4 parameters for the 4 or 5 digits respectively. Each airfoil is defined via a camber lined and an envelope, the only difference between the 4 and 5 digits is the definition of the camber line.

NACA 4 digits. Those profiles are defined by the name NACA followed by four digits MPXX where the first two digits M and P defined the camber line and the last two digits XX defined the maximum thickness of the profile in percentage of the chord (the total length of the airfoil). More precisely, M defines the maximum ordinate of the camber line in hundredth of the chord and P the position of this maximum from the leading edge in

tenth of the chord. If we denote the chord c, the camber line of the NACA 4312 profile will have a maximum ordinate of camber of $y = 0.04c$, at $x = 0.3c$ and the profile will have a maximum thickness of $0.12c$. Also, the leading edge and the trailing edge of each airfoil are always taken at the points $(0, 0)$ and $(c, 0)$ in the $x - y$ plane respectively. From this point, all the abscissas and ordinates will be given in length per chord.

For the NACA 00XX, a symmetrical profile, the camber line is a straight line from $x = 0$ to $x = 1$ and the upper surface is defined by the graph of the function:

$$y_t(x) = \frac{t}{0.2}\left(0.2969\sqrt{x} - 0.126x - 0.3516x^2 + 0.2843x^3 - 0.1015x^4\right) \tag{1.36}$$

where $t := XX/100$ is the thickness defined with the two last digits. This definition involve a trailing edge with a thickness of $0.002c$. If, for example for numerical propose, we want to have a thickness of 0 at the trailing edge, we can change the coefficient of the fourth order term from -0.1015 to -0.1036. The lower surface is defined as $-y_t$.

For a generic NACA MPXX, The camber line is defined by the first two digits and follow the graph of the function:

$$y_c(x) = \begin{cases} m\frac{x}{p^2}(2p - x), & 0 \leqslant x \leqslant p \\ m\frac{1-x}{(1-p)^2}(1 + x - 2p), & p < x \leqslant 1 \end{cases} \tag{1.37}$$

where $m := 0.01$M and $p := 0.1$P.

Finally, the upper surface of a generic NACA MPXX is given by the set of coordinates (x_u, y_u) defined as:

$$\begin{cases} x_u = x - y_t(x)\sin\theta(x) \\ y_u = y_c(x) + y_t\cos\theta(x) \qquad \text{for } x \in [0, 1] \\ \theta(x) = \arctan y_c'(x) \end{cases} \tag{1.38}$$

where y_c' is the derivative of y_c. The lower surface is given by a similar set of coordinates (x_l, y_l) defined as:

$$\begin{cases} x_l = x + y_t(x)\sin\theta(x) \\ y_u = y_c(x) - y_t\cos\theta(x) \qquad \text{for } x \in [0, 1] \\ \theta(x) = \arctan y_c'(x) \end{cases} \tag{1.39}$$

NACA 5 digits. As we already stated earlier, the only difference between the NACA 4 and 5 digits is the definition of the camber line. In the case of the 5-digits LPQXX, the 3 first parameters defining the camber line are less explicit than for the 4 digits case but allow more complex shapes. The last two are the same as in the 4-digits case (*i.e.* maximum thickness in hundredth of chord). The first digit L controls the camber implicitly via an optimal lift coefficient C_L, *i.e.* it will give you the camber of the airfoil such that $C_L = 0.15$L. The second digit P is almost the same as in the 4-digits case, *i.e.* it defines the position of the maximum of camber of the camber line in twentieth of chord. Lastly, the third digit Q is either 0 or 1 and represent a standard camber (similar to the 4-digits case) for 0 and a reflex

camber for 1. The reflex camber is a double cambered line that makes the profile more stable (by setting the pitch coefficient to 0).

For a generic NACA LPQXX, the standard camber line is defined via the graph of the function:

$$y_c(x) = \begin{cases} K_1 \left(m^2(3-m)x - 3mx^2 + x^3\right), & 0 \leqslant m \\ K_1 m^3(1-x), & m < x \leqslant 1 \end{cases} \tag{1.40}$$

where m is not the position of the maximum camber $p := 0.05\mathrm{P}$ but is related to it via the equation:

$$p = m\left(1 - \sqrt{\frac{m}{3}}\right) \tag{1.41}$$

and K_1 is related to C_L via:

$$\begin{cases} K_1 = \frac{C_L}{Q} \\ Q = \frac{3m - 7m^2 + 8m^3 - 4m^4}{\sqrt{m(1-m)}} - \frac{3}{2}(1-2m)\left(\frac{\pi}{2} - \arcsin(1-2m)\right) \end{cases} \tag{1.42}$$

The reflex camber line is defined via the graph of the function:

$$y_c(x) = \begin{cases} K_1 \left(m^3(1-x) - K_2(1-m)^3 x + (x-m)^3\right), & 0 \leqslant m \\ K_1 \left(m^3(1-x) - K_2(1-m)^3 x + K_2(x-m)^3\right), & m < x \leqslant 1 \end{cases} \tag{1.43}$$

where m and K_1 are defined as in the standard case and K_2 is defined via:

$$K_2 = \frac{3(m-p)^2 - m^3}{(1-m)^3} \tag{1.44}$$

We use a standard Newton's method to numerically solve equation 1.41 in m.

Parameter sets. In Table 1.1 we give the parameter sets for the sampling. Let us underline that the digits are not necessarily integers. Also, for the values of P in the NACA 4-digits case, we actually uniformly sample in $[0, 7]$ and set the values of P strictly inferior to 1.5 to 0. This implies that the NACA 4-digits set is slightly biased towards symmetrical profiles. We assume that this bias is not of a great importance in the ML task. This bias could actually be leveraged to produce a smaller dataset of only symmetrical profiles where we can test the performance of invariant/equivariant models with respect to the plane symmetry of axis $y = 0$. Finally, we set the chord c to $1\,\mathrm{m}$ for all of the airfoils. In Figure 1.F.1, we show statistics of the airfoil parameters.

1.G Meshing Procedure

The construction of meshes is at the core of CFD tasks. A mesh completely determine the quality of a simulation and its characteristics. For CFD problems, the local size of cells

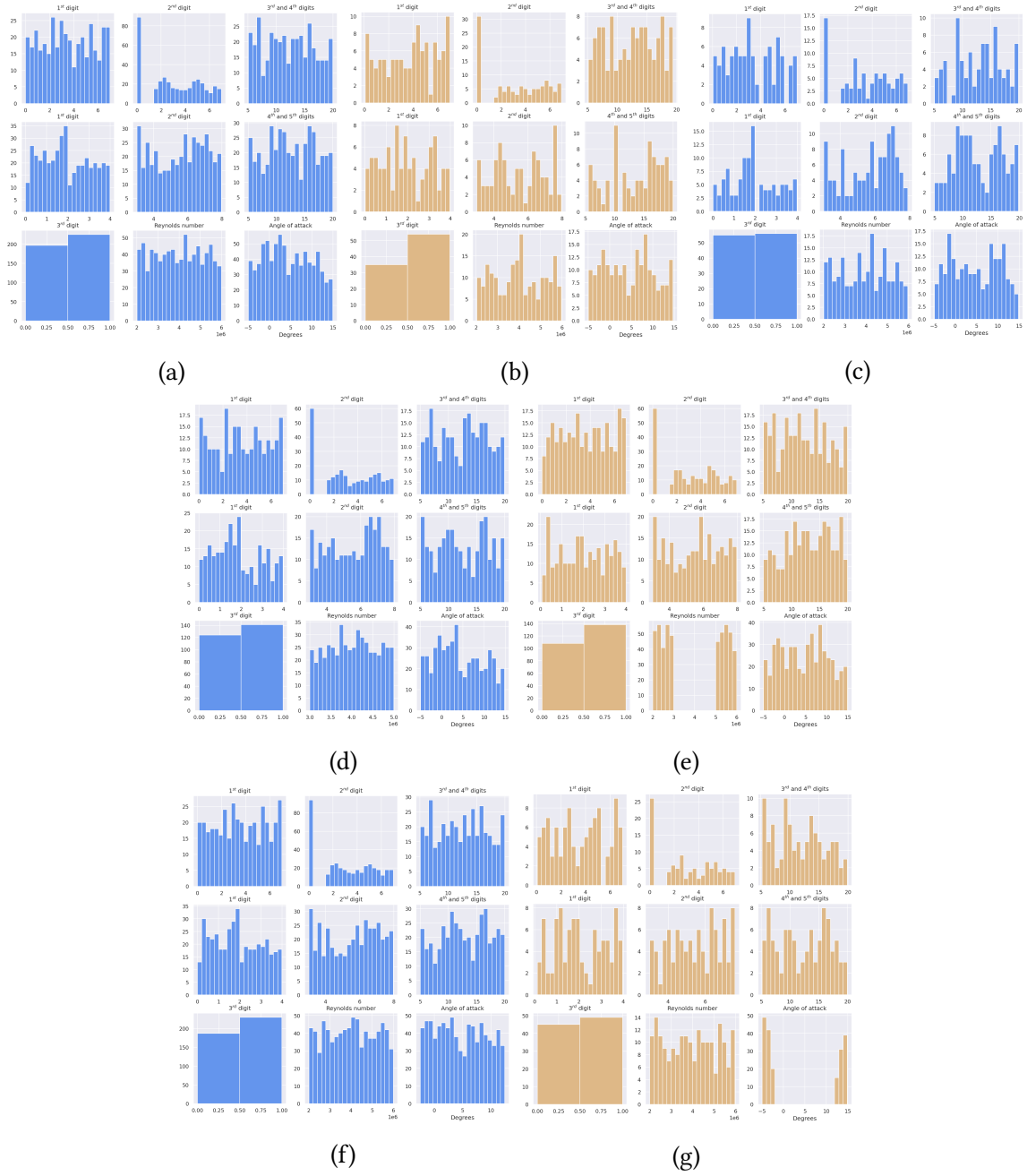


Figure 1.F.1: Histogram of the different sampled parameters for the airfoils. For each subfigure, the top line represents the parameters of NACA 4-digits, the middle line, the parameters of the NACA 5-digits along with the left plot of the bottom line. The last two plots of the bottom lines are the histograms for the Reynolds number and the angle of attack. Each subfigure represents a different regime: (a)-(b) Full data regime (c) Scarce data regime (d)-(e) Reynolds extrapolation regime (e)-(f) Angle of attack extrapolation regime. Trainsets are in blue and testsets in yellow.

gives the information of the resolved scales. For example, in turbulent regime, it would be impossible to simulate eddies of characteristic length smaller than your typical cell length scale. Unfortunately, it is often impossible to run direct numerical simulation (DNS) to correctly simulate all the length scale of a fluid dynamics problem as it implies a prohibitive quantity of cells in the mesh. Another technology, large eddy simulation (LES), tries to model the smallest length scales via a theoretical local filtering of the solution in order to reduce the characteristic length scale of the smallest cell needed to correctly simulate the phenomena studied. However, this can still lead to a prohibitive quantity of cells in the mesh. Finally, Reynolds-Averaged Simulation (RAS), use the RANS equations described in section 1.D to model all the length scales involved in turbulence via a theoretical ensemble averaging. This allows to recover a mean field solution which requires much less cells in the mesh. In this work, we chose to run steady-state RAS to recover mean steady-state fields around airfoils.

Another aspect of the construction of a mesh is the choice of a strategy to resolve the boundary layers close to an obstacle. There are two strategies but each are based on the value of the y^+. This quantity represents a local Reynolds close to the obstacle and is defined as:

$$\begin{cases} y^+ = \frac{y u_\tau}{\nu} \\ u_\tau = \sqrt{\frac{\tau_w}{\rho}} \end{cases} \tag{1.45}$$

where y is the distance from the wall, τ_w the magnitude of the wall shear stress, ρ the density of the fluid and ν the kinematic viscosity of the fluid. In order to compute the y^+, experimental values on thin plates give the order of magnitude of the wall shear stress term [67]. When we take y as the height of the first cell close to the wall, we can define two strategies:

- *Low-Reynolds simulation:* resolve entirely the boundary layer, $y^+ < 1$
- *High-Reynolds simulation:* model the boundary layer via a so-called wall function, $y^+ \sim 10^2$

As we are interested in accurate force coefficients at the surface of our airfoils, we chose the first strategy to avoid modelling close to the wall. This implies that we need the maximum height y of our first cells close to the wall to be smaller than ν/u_τ. In our case, we chose $y = 2\,\mu\text{m}$ which set the y^+ to be around 1 in the worst case of our design space.

Let us now present the mesh we use for our simulations. This is inspired by the National Aeronautics and Space Administration (NASA) mesh used in [42] to recover experimental force coefficients on the NACA 0012 and 4412. We do not pretend to have the same quality of mesh as the NASA but we still argue that our mesh is well suited for our case. We show it in the next sections by comparing our results to experimental results. Meshes have been generated with the help of *blockMesh*, a hexahedral mesh generator included in the OpenFOAM suite that works by defining blocks. With the help of a dictionnary (namely the *blockMeshDict* file), we set the number of cells and the grading we want to fully determined the meshing inside each block. A scheme of how the domain is divided in multiple blocks

and a result of the meshing procedure on the NACA 0012 with an angle of attack of 10° is given in Figure 1.G.1. More precisely, as in the NASA mesh, a C-Grid domain is defined with a radius and a length of $200\,\mathrm{m}$ (which means 200 chords here). This is smaller than the 500 chords length domain of the NASA but we found it big enough to be insensible to boundary conditions. We now use the index of the nodes, edges and blocks defined on Figure 1.G.1. For the edges 310, 411, 58, 69 and 710, the smallest cell is of height $2\,\mu\mathrm{m}$ as already said above and we set the expansion ratio (the length ratio between two consecutive cells) to 1.075. For the edges 01 and 12 the smallest cell is of height $100\,\mu\mathrm{m}$ and the expansion ratio is also set to 1.075. At the upper surface of the airfoil, at the leading edge, the smallest cell is of width $10\,\mu\mathrm{m}$ (at node 8) and we set the expansion ratio to 1.025 until roughly the maximum of camber of the airfoil (at node 11). From node 11 to node 10, an automatic expansion ratio is computed to fill the entire segment. Almost the same procedure is applied at the lower surface of the airfoil, the only difference is that, for consistency, the expansion ratio between node 9 and 10 is set such that the last cells (at node 10) is of the same width as the one at the upper surface. Edge 34 or 67 have a fix grading of 1 and edge 45 or 56 have a grading such that the width of the cell at the junction of blocks 2 and 3 or 4 and 5 are the same (this is only true at node 4 or 6 and a significant width difference can be seen at the center of edges 411 or 69). Finally edges 07, 110 and 23 have a smallest cell of the same width as for block 2 or 5 (with the same remark, this is true only at nodes 3, 7 and 10) and a grading of 1.075 is applied.

In Figure 1.G.2 we give the number of cells and nodes in simulations of the dataset. We also give those quantities for the cropped simulations used for the ML tasks.

1.H Boundary Conditions

In this section, we explicit the different boundary conditions set on the different patches of the mesh. The quantities needed for a simulation depend on whether we run incompressible or compressible physics and on the turbulence model chosen. In Table 1.H.1, 1.H.2 and 1.H.3 we give the different OpenFOAM settings used for the different fields involved in the simulation, that are:

- U : ensemble averaged velocity in $\mathrm{m\,s^{-1}}$
- p : ensemble averaged effective pressure in Pa (in the incompressible case the pressure is divided by the density ρ, $p \to p/\rho$)
- ν_t : kinematic turbulent viscosity in $\mathrm{m^2\,s^{-1}}$
- $\tilde{\nu}$[6]: Spalart-Allmaras variable in $\mathrm{m^2\,s^{-1}}$
- k[2] : turbulent kinetic energy in J
- ω[2] : specific dissipation rate via turbulence in $\mathrm{s^{-1}}$

[6]Only for the Spalart-Allmaras turbulent model.

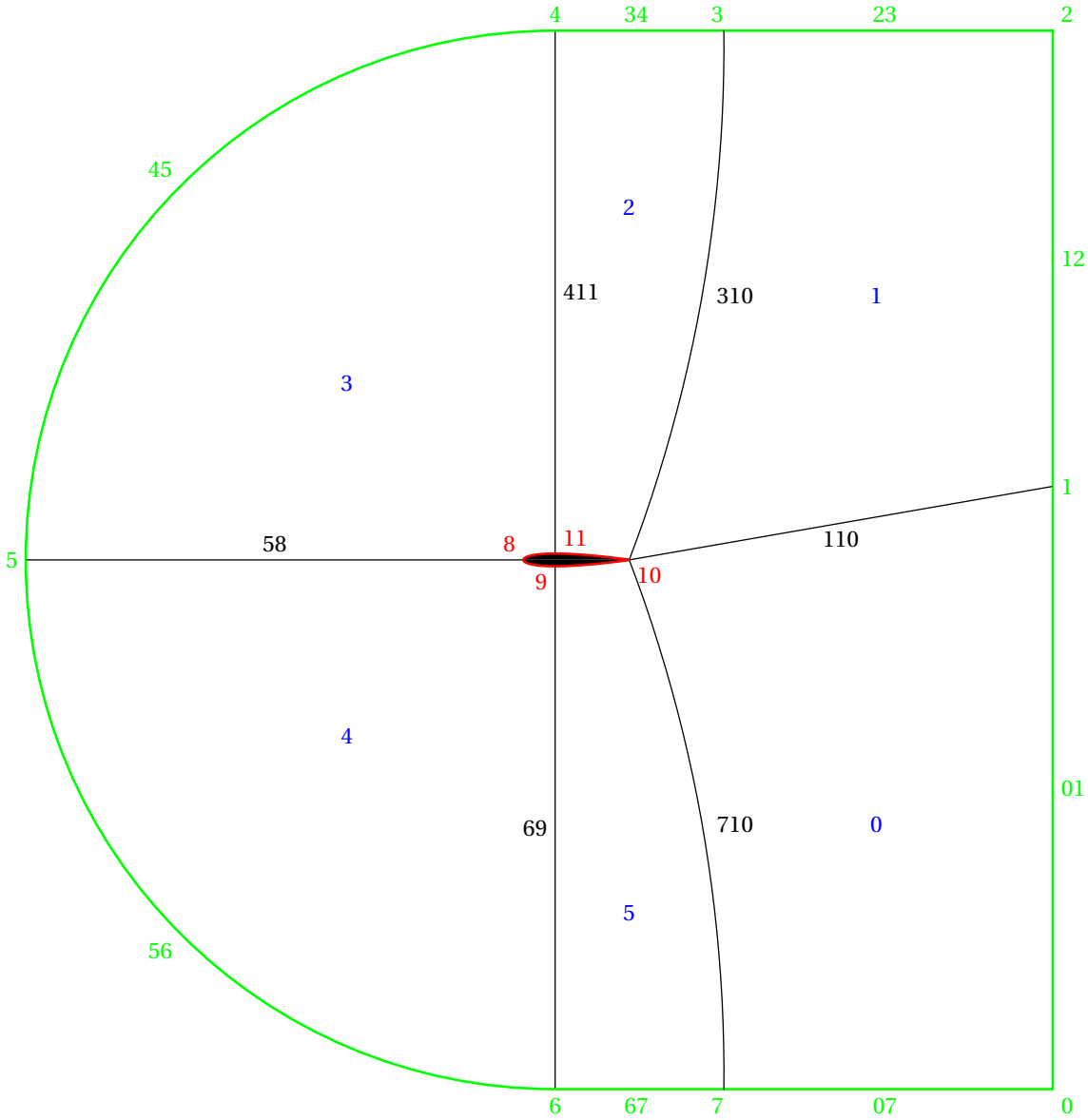


Figure 1.G.1: Scheme of the mesh template. This is a scheme for the NACA 0012 with an angle of attack of $10°$. The aerofoil patch is highlighted in red, the freestream patch is highlighted in green and the internal patch is the union of the blocks 0 to 5 highlighted in blue. The indices of nodes and blocks are the same as in the *blockMeshDict* file.

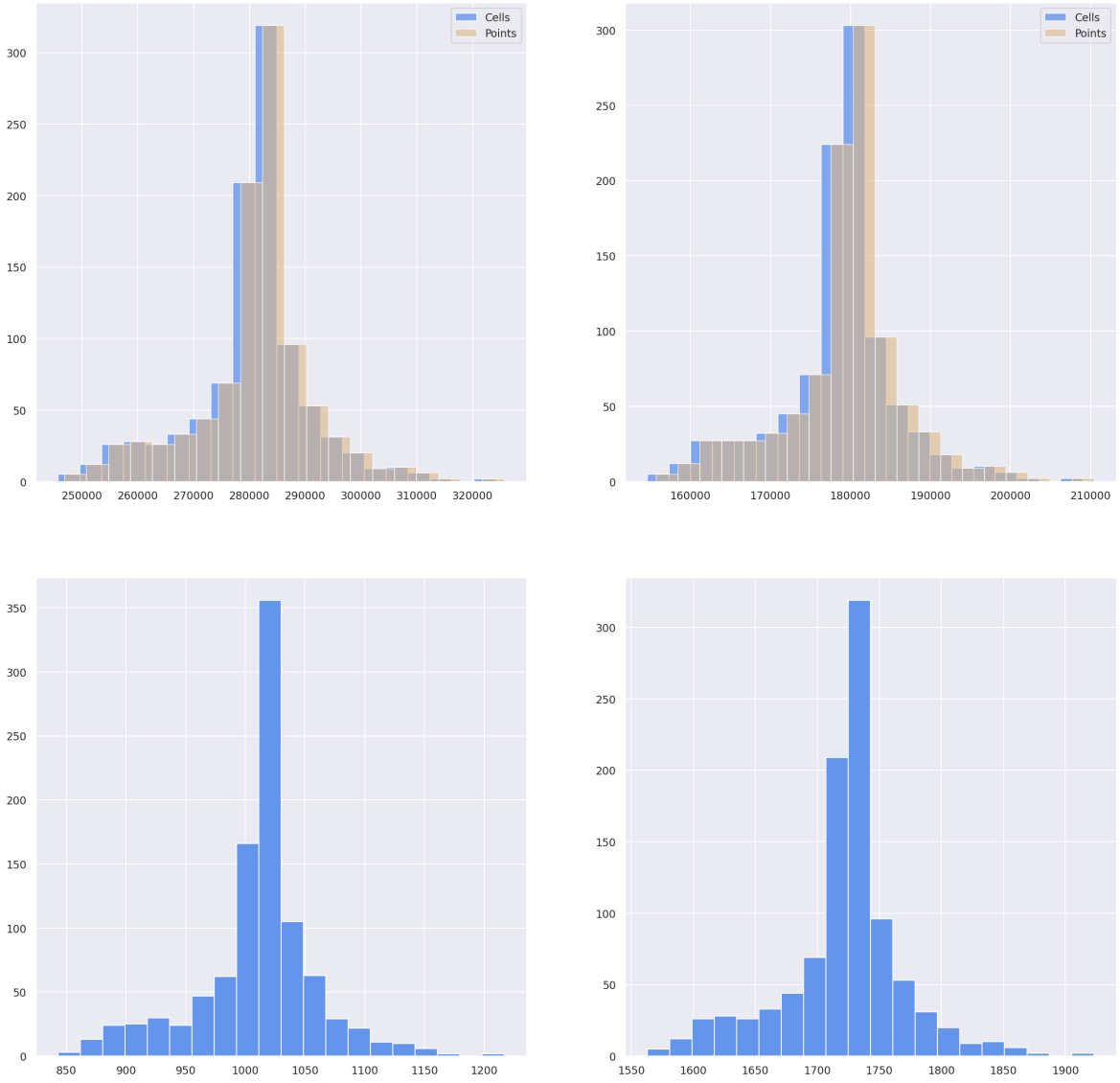


Figure 1.G.2: Histograms of the number of cells and nodes in simulations of the dataset. (top left) Number of cells and nodes in internal meshes for CFD simulations (top right) Numbers of cells and nodes in internal meshes for cropped simulations (bottom left) Number of nodes on airfoils patches (bottom right) Number of nodes on freestream patches. For the bottom plots, we only give the number of nodes as they are equal to the number of cells.

Table 1.H.1: Boundary conditions set on the different patches of the mesh for compressible and incompressible simulations. Values of the constants are given for the air at sea level and at $298.15\,\mathrm{K}$.

Fields	Internal	Aerofoil	Freestream
U	U_∞	*noSlip*	*freestreamVelocity*
p	$0^\star$	*zeroGradient*	*freestreamPressure*
ν_t	ν	*nutLowReWallFunction*	*freestream*
$\tilde{\nu}$	4ν	*fixedValue*	*freestream*
k	$0.001U_\infty^2/Re_L$	*fixedValue*	*freesteam*
ω	$5U_\infty/L$	*omegaWallFunction*	*freestream*
T	$298.15\,\mathrm{K}$	*zeroGradient*	*freestream*
α_t	ν_t/Pr_t	*compressible::alphatWallFunction*	*calculated*

* This value has to be set to an absolute pressure value, in our case $1.013 \times 10^5\,\mathrm{Pa}$, for the compressible case.

Table 1.H.2: Definition of the quantities involved in Table 1.H.1 and their values for the air at sea level and at a temperature of $298.15\,\mathrm{K}$ ($25\,^\circ\mathrm{C}$).

Quantity	Definition	Value
ρ	Density of the fluid	$1.184\,\mathrm{kg\,m^{-3}}$
ν	Kinematic viscosity of the fluid	$1.56 \times 10^{-5}\,\mathrm{m^2\,s^{-1}}$
L	Length of the domain	$400\,\mathrm{m}$
U_∞	Velocity at the inlet	-
Re_L	Reynolds number computed with L	$U_\infty L/\nu$
Pr_t	Turbulent Prandtl number (constant)	0.85

- T[3] : temperature in K

- α_t[3] : turbulent thermal diffusivity in $\mathrm{m^2\,s^{-1}}$

We do not present here the discretization schemes chosen for the simulations, nor the linear solver and hyperparameters of the SIMPLE algorithm. You can find them in the *fvSchemes* and *fvSolution* dictionnaries respectively. We just mention that we used the SIMPLEC [50] algorithm for the incompressible case and the classical SIMPLE [49] one for the compressible setup as the SIMPLEC was not stable in this case.

[2]Only for the $k-\omega$ SST turbulent model [46].
[3]Only in the case of compressible simulations.

Table 1.H.3: Definition of the boundary conditions involved in Table 1.H.1 and values we use when asked by OpenFOAM. The values given for *fixedValue* are the values of k and $\tilde{\nu}$ at the surface of the airfoil. The quantity $\beta_1 = 0.075$ is a constant of the $k-\omega$ model [66] and $\Delta y = 2\,\mu\mathrm{m}$ the height of the first cells of the boundary layer.

Boundary condition	Definition	Value
fixedValue	Set the quantity to a constant	$0\,\mathrm{J}/0\,\mathrm{m}^2\,\mathrm{s}^{-1}$
calculated	Derived from other quantities	Internal field value
noSlip	Set the velocity to 0	-
zeroGradient	Set the field at the boundary to the value of the internal field	-
freestream	Mixed boundary condition between *fixedValue* and *zeroGradient* depending on the direction of the flux	Internal field value
freestreamVelocity	Same as *freestream* but switches in accordance with *freestreamPressure*	Internal field value
freestreamPressure	Same as *freestream* but switches in accordance with *freestreamVelocity*	Internal field value
nutLowReWallFunction	Set the turbulent viscosity to 0	$0\,\mathrm{m}^2\,\mathrm{s}^{-1}$
omegaWallFunction	For low Reynolds simulation, equivalent to *fixedValue*	$\frac{6\nu}{\beta_1 \Delta y^2}$
compressible::alphatWallFunction	Equivalent to *fixedValue* with a value of ν_t/Pr_t	$0\,\mathrm{m}^2\,\mathrm{s}^{-1}$

1.I Simulation Validation

In this section, we test our mesh and boundary conditions on two different problems, with two turbulence models and in the compressible and incompressible settings, in order to validate the choice made in this work. To do so, we use the experimental data produced by the NASA and available on the Turbulence Modeling Resource (TMR) website of the Langley Research Center [42] for the NACA 0012 and 4412.

NACA 0012 airfoil. We compare our results with experimental data for the force coefficients done on the NACA 0012 [43, 44]. We restricted our study to the case of a Reynolds of 6 million for different angle of attacks (see Table XIII [44]). In our simulations, we run an incompressible solver with the properties of the air at $298.15\,\mathrm{K}$ and at sea level (see Table 1.H.2) which gives an inlet velocity U_∞ of $93.6\,\mathrm{m\,s^{-1}}$ with a characteristic length equal to the chord of the airfoil (in our case $1\,\mathrm{m}$). The celerity of sound in this medium is taken to be $346.1\,\mathrm{m\,s^{-1}}$, which gives a Mach number ($Ma := U_\infty/c$) of roughly 0.27. We often set a limit a 0.3 for the Mach number in order to run incompressible simulations, and as we are close to this limit, we run incompressible and compressible simulations for an additional restricted set of angle of attacks of $0°$ and $10°$. The pressure coefficient at the surface of the airfoil are compared to another set of experimental data (Table II of [43]) done at Mach 0.3 and Reynolds 6 million for this two angles of attack (more precisely at angle $0.0169°$ and $10.0254°$). Moreover, we tried with the Spalart-Allmaras and $k-\omega$ SST models of turbulence.

The pressure coefficient c_p at the surface of the airfoil is a dimensionless coefficient defined as:

$$c_p := \frac{\bar{p} - \bar{p}_\infty}{q_\infty}, \qquad q_\infty := \frac{1}{2} U_\infty^2 A \tag{1.46}$$

where $\bar{p}$ is the mean-field reduced pressure, $\bar{p}_\infty$ the far field pressure (set to 0 in the incompressible case), U_∞ is the magnitude of the inlet velocity and A is the characteristic area of the problem, we take here $A = 1\,\mathrm{m}^2$.

In Figure 1.I.1 and 1.I.2, the surface pressure coefficient is given and we see no significant difference between the two models nor between the compressible and incompressible cases. All the simulations are in good agreement with the experiments.

In Figure 1.I.3 are displayed the drag and lift coefficients with respect to angle of attacks and the drag coefficient with respect to the lift coefficient for the two models and the experiments. In the compressible case, only $0°$ and $10°$ have been simulated for time and stability reasons, no significant differences are present with the incompressible simulations. In the incompressible case, a missing point in the plot means that the simulation was unstable and we did not manage to make it converge correctly. We can see that both compressible and incompressible solver gives a slightly over estimated drag with respect to experiments, this is in agreement with the TMR. We also see that the $k-\omega$ SST model is more stable than the Spalart-Allmaras model, we noticed a faster convergence for the first too. Finally, the $k-\omega$ SST model fits better the experiments than the Spalart-Allamaras model. In total,

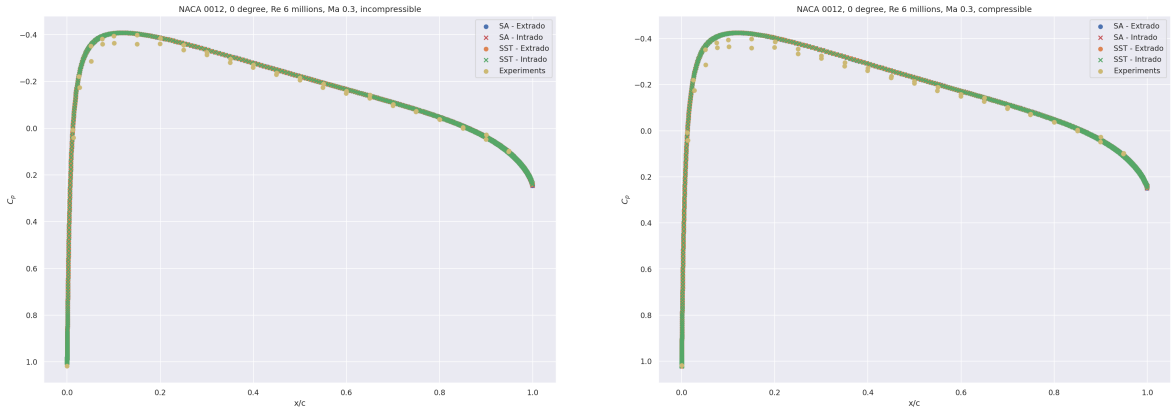


Figure 1.I.1: Pressure coefficient at the surface of the airfoil for the NACA 0012 at an angle of attack of $0°$ in the incompressible (left) and compressible (right) cases for the Spalart-Allamaras, $k-\omega$ SST models and the experiments with respect to the abscissas in chord length. The points on the upper and lower surfaces are given in different colors for the simulations.

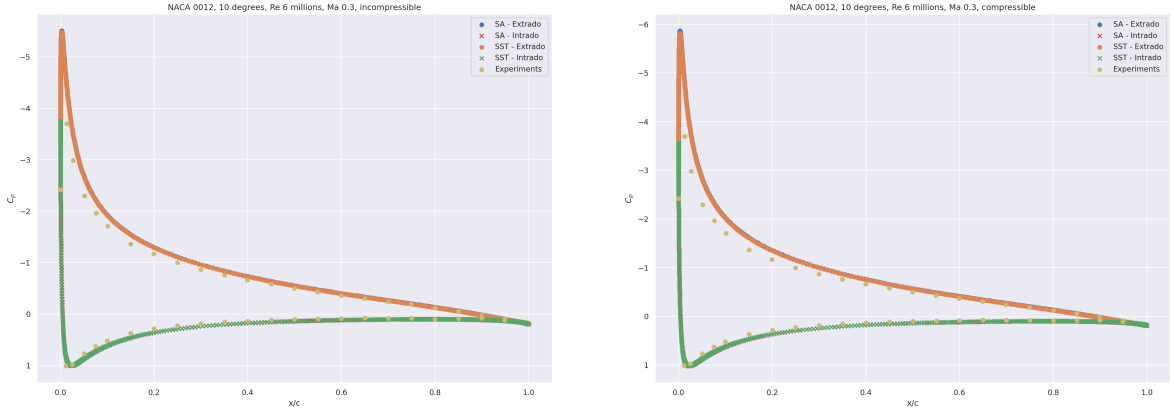


Figure 1.I.2: Pressure coefficient at the surface of the airfoil for the NACA 0012 at an angle of attack of $10°$ in the incompressible (left) and compressible (right) cases for the Spalart-Allamaras, $k-\omega$ SST models and the experiments with respect to the abscissas in chord length. The points on the upper and lower surfaces are given in different colors for the simulations.

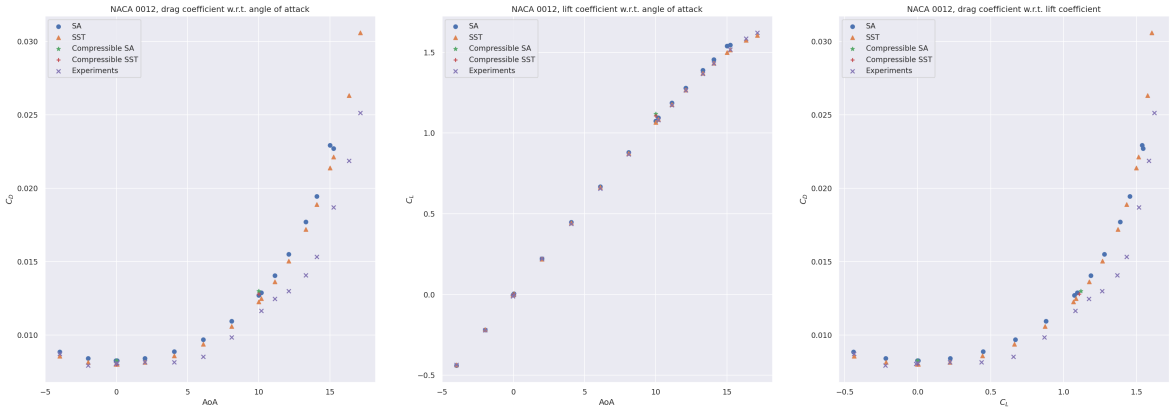


Figure 1.I.3: Drag and lift coefficients with respect to angles of attack and to each other in the case of the NACA 0012. The compressible simulations have only be done at $0°$ and $10°$.

both model are in good agreement with experimental data but the $k-\omega$ SST model looks more stable, faster to converge and more accurate than the Spalart-Allamaras.

From this point, we only run incompressible simulations as this validation case showed no distinctions between compressible and incompressible simulations. We now test our setup on another validation case in order to chose between the two turbulence models.

NACA 4412 airfoil. In this setup, the experimental data [45] are done with a NACA 4412 at an angle of attack of $13.87°$ and a Reynolds number of 1.52 million. The values of the experimental data are the one given on the NACA 4412 page of the TMR. The values found on this website are slightly different from the one found in the original papers, moreover, the normalization factor for the pressure coefficient is computed with a reference velocity U_{ref} of roughly $0.93U_\infty$ but, as in the NASA simulations, the results better fit when using a normalization factor computed with U_∞. This is underlined on the TMR page and incite us to take this validation case only as a qualitative validation.

In Figure 1.I.4, the pressure coefficient is given with a normalization factor computed with the magnitude of the inlet velocity U_∞. Both turbulence models results are in good agreement with the experiments.

In Figure 1.I.5, we look at the boundary layer of the airfoil at different abscissas. Here the x and y components of the velocity (denoted by u and v respectively) are normalized by U_{ref} and the term $u'v'$, corresponding to the shear stress term of the Reynolds stress tensor, is normalized by U_{ref}^2. We start each plot at a given point at the surface of the airfoil and take the direction of the normal of the airfoil at this point. Hence, the name $(y-y_0)/c$ for the ordinate of the plot has to be understand as the distance to the airfoil in the normal direction in chord length. Both turbulence models have difficulties to predict correctly the experimental data, this behaviour has already been pointed out in the TMR study of the NACA 4412 and our results are in good agreement with theirs. Moreover, the $k-\omega$ SST model seems to give more realistic results than the Spalart-Allmaras one.

In total, our simulations on the NACA 0012 and 4412 are in good (at least qualitatively)

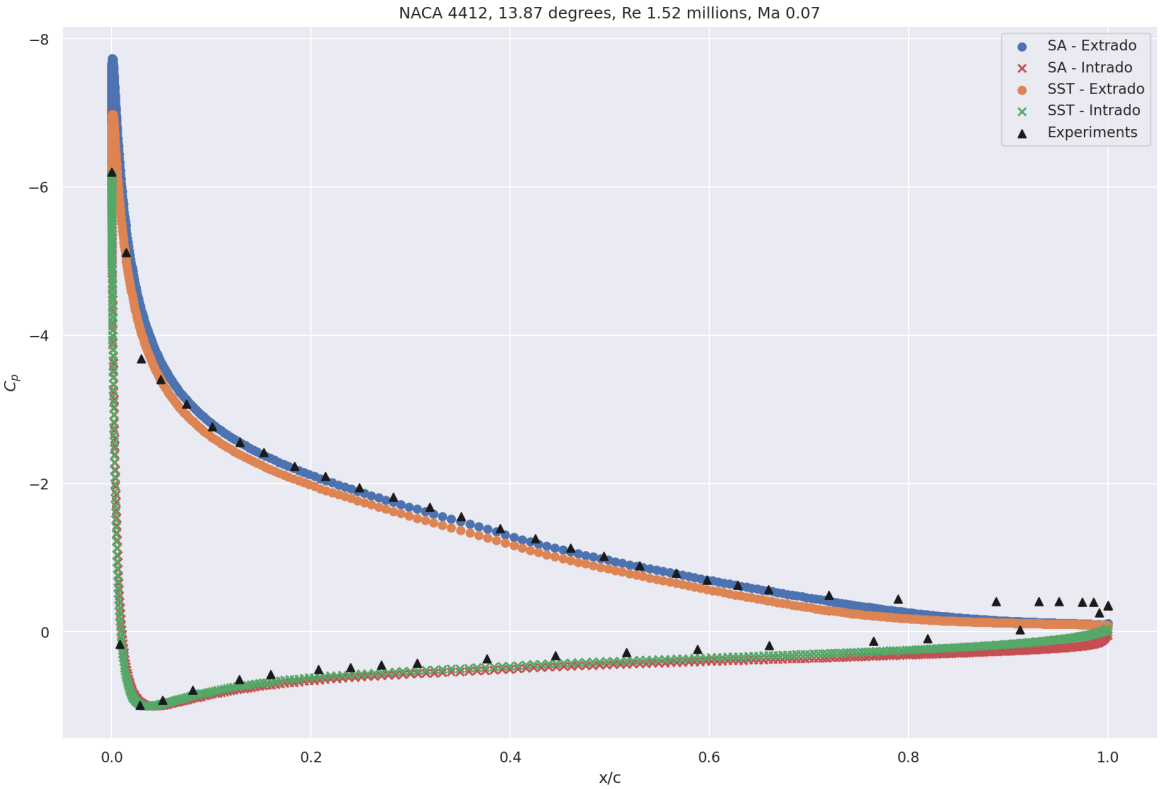


Figure 1.I.4: Pressure coefficient at the surface of a NACA 4412 with an angle of attack of $13.87°$ and a Reynolds of 1.52 million. The normalization of the pressure coefficient is computed with the magnitude of the inlet velocity U_∞. Are displayed the experimental data, the Spalart-Allmaras and $k-\omega$ SST models incompressible results.

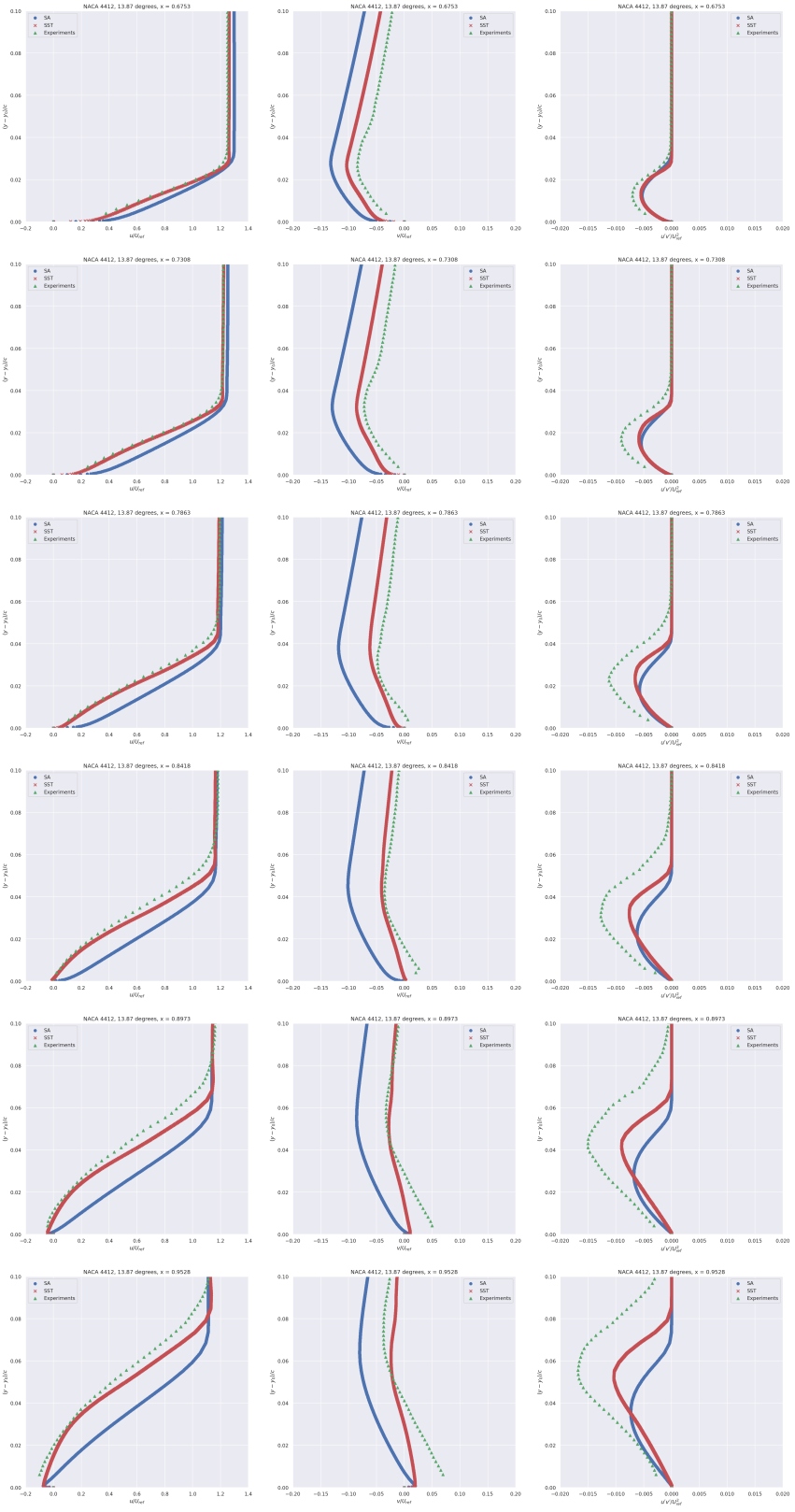


Figure 1.I.5: Boundary layer velocity components and shear Reynolds stress for different point at the surface of the NACA 4412 at a Reynolds number of 1.52 million. Each quantity is normalized either by U_{ref} or U_{ref}^2. The ordinate has to be understand as the distance to the given point at the surface of the airfoil following the normal direction.

agreement with the experiments. The incompressible $k-\omega$ SST model setup seems the best candidate for fast, stable and high fidelity simulations. In this work, we keep this setup to generate the dataset.

1.J Models architecture

For all of the tasks, the same architecture is used in addition with the same hyperparameters. Each model is preceded by an encoder and followed by a decoder both defined as MLP with ReLU activation function, no batch normalization, and with $7-64-64-8$ and $8-64-64-4$ neurons respectively, meaning an dimension of encoding of 8. Those encoder and decoder are trained together with the chosen model.

Multi-Layer Perceptron. The first baseline is another MLP with ReLU activation function and batch normalization before the activation. It has $8-64-64-64-8$ neurons.

GraphSAGE. The GraphSAGE acts on a radius graph of 32000 nodes and radii $5\,\mathrm{cm}$. It is defined with 3 hidden layers and 64 hidden features per node.

PointNet. The PointNet is copied from the segmentation task of [54]. We chose 8 neurons as a base number and we did not include any batch normalization nor dropout as it was performing badly with.

Graph U-Net. For the Graph U-Net, we defined it with five scales, downsampling by half at each scale and multiplying by two the number of features at each scales. The radii of the radius graphs are $5\,\mathrm{cm}$, $20\,\mathrm{cm}$, $50\,\mathrm{cm}$, $1\,\mathrm{m}$ and $10\,\mathrm{m}$. The last radii is chosen such that the graph at the coarsest scale is fully connected. Each of those radius graphs have a limit of 64 neighbors per node. For the downsampling, we did not use the gPool method presented in the historical paper [55] and replaced it by a random downsampling over the remaining nodes, recreating a radius graph afterwards. This leads to better results. On the upward pass, we chose to aggregate the different informations from the skip connection and the preceding scale by concatenating the features. Finally, we chose to start with 8 features at the finest scale.

The learning rate for all of those experiments is set with a one-cycle cosine [68] rate of maximum 0.001, simulations are fed one by one to the different models during training (*i.e.* 32000 nodes with an associated radius graph when needed) and the number of epochs is chosen such that for each task, we have the same number of gradient updates:

- *Full data regime:* 400 epochs
- *Scarce data regime:* 1600 epochs
- *Reynolds extrapolation regime:* 635 epochs

- *Angle of attack extrapolation regime:* 398 epochs

Ultimately, the different models are trained on 90% of the predefined training set of those different regime, the last 10% have been used as a validation set.

References

1. Otness, K. *et al. An Extensible Benchmark Suite for Learning to Simulate Physical Systems* in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)* (2021). https://openreview.net/forum?id=pY9MHwmrymR.
2. Bonnet, F., Mazari, J. A., Munzer, T., Yser, P. & Gallinari, P. *An extensible Benchmarking Graph-Mesh dataset for studying Steady-State Incompressible Navier-Stokes Equations* in *ICLR 2022 Workshop on Geometrical and Topological Representation Learning* (2022). https://openreview.net/forum?id=rqUUi4-kpeq.
3. Anderson, J. *Fundamentals of aerodynamics (6th edition)* (McGraw-Hill Education, 2017).
4. Jiang, P., Zhou, Q. & Shao, X. *Surrogate Model-Based Engineering Design and Optimization* ISBN: 978-981-15-0730-4 (Springer Tracts in Mechanical Engineering (STME), Jan. 2020).
5. Forrester, A. I. J., Sobester, A. & Keane, A. J. *Engineering Design via Surrogate Modelling - A Practical Guide.* I–XVIII, 1–210. ISBN: 978-0-470-06068-1 (Wiley, 2008).
6. Thuerey, N., Weißenow, K., Prantl, L. & Hu, X. Deep Learning Methods for Reynolds-Averaged Navier–Stokes Simulations of Airfoil Flows. *AIAA Journal* **58,** 25–36 (2020).
7. Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A. & Vandergheynst, P. Geometric Deep Learning: Going beyond Euclidean data. *IEEE Signal Processing Magazine* **34,** 18–42 (2017).
8. Pfaff, T., Fortunato, M., Sanchez-Gonzalez, A. & Battaglia, P. *Learning Mesh-Based Simulation with Graph Networks* in *International Conference on Learning Representations* (2021). https://openreview.net/forum?id=roNqYL0_XP.
9. De Bezenac, E., Pajot, A. & Gallinari, P. *Deep Learning for Physical Processes: Incorporating Prior Scientific Knowledge* in *International Conference on Learning Representations* (2018). https://openreview.net/forum?id=By4HsfWAZ.
10. Chen, Z., Zhang, J., Arjovsky, M. & Bottou, L. *Symplectic Recurrent Neural Networks* in *International Conference on Learning Representations* (2020). https://openreview.net/forum?id=BkgYPREtPr.
11. Sirignano, J. & Spiliopoulos, K. DGM: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics* **375,** 1339–1364. ISSN: 0021-9991. https://www.sciencedirect.com/science/article/pii/S0021999118305527 (2018).

12. Long, Z., Lu, Y., Ma, X. & Dong, B. *PDE-Net: Learning PDEs from Data* in *Proceedings of the 35th International Conference on Machine Learning* (eds Dy, J. & Krause, A.) **80** (PMLR, July 2018), 3208–3216. https://proceedings.mlr.press/v80/long18a.html.

13. Brandstetter, J., Welling, M. & Worrall, D. E. Lie Point Symmetry Data Augmentation for Neural PDE Solvers. *arXiv preprint arXiv:2202.07643* (2022).

14. Li, Z. *et al. Multipole Graph Neural Operator for Parametric Partial Differential Equations* in *Advances in Neural Information Processing Systems* (eds Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. & Lin, H.) **33** (Curran Associates, Inc., 2020), 6755–6766. https://proceedings.neurips.cc/paper/2020/file/4b21cf96d4cf612f239a6c322b10c8fe-Paper.pdf.

15. Kovachki, N. B. *et al.* Neural Operator: Learning Maps Between Function Spaces. *ArXiv* **abs/2108.08481** (2021).

16. Li, Z. *et al. Fourier Neural Operator for Parametric Partial Differential Equations* in *International Conference on Learning Representations* (2021). https://openreview.net/forum?id=c8P9NQVtmnO.

17. Lu, L., Jin, P., Pang, G., Zhang, Z. & Karniadakis, G. E. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence* **3,** 218–229 (2021).

18. Raissi, M., Perdikaris, P. & Karniadakis, G. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics* **378,** 686–707. ISSN: 0021-9991. https://www.sciencedirect.com/science/article/pii/S0021999118307125 (2019).

19. Krizhevsky, A., Sutskever, I. & Hinton, G. E. *ImageNet Classification with Deep Convolutional Neural Networks* in *Advances in Neural Information Processing Systems* (eds Pereira, F., Burges, C., Bottou, L. & Weinberger, K.) **25** (Curran Associates, Inc., 2012). https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

20. Carreira, J., Noland, E., Hillier, C. & Zisserman, A. A Short Note on the Kinetics-700 Human Action Dataset. *CoRR* **abs/1907.06987.** http://arxiv.org/abs/1907.06987 (2019).

21. Amodei, D. *et al. Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin* in *Proceedings of The 33rd International Conference on Machine Learning* (eds Balcan, M. F. & Weinberger, K. Q.) **48** (PMLR, New York, New York, USA, June 2016), 173–182. https://proceedings.mlr.press/v48/amodei16.html.

22. Yin, Y. *et al. Augmenting Physical Models with Deep Networks for Complex Dynamics Forecasting* in *International Conference on Learning Representations* (2021). https://openreview.net/forum?id=kmG8vRXTFv.

23. Zobeiry, N. & Humfeld, K. D. A physics-informed machine learning approach for solving heat transfer equation in advanced manufacturing and engineering applications. *Engineering Applications of Artificial Intelligence* **101,** 104232. ISSN: 0952-1976. https://www.sciencedirect.com/science/article/pii/S0952197621000798 (2021).

24. Dubois, P., Gomez, T., Planckaert, L. & Perret, L. Data-driven predictions of the Lorenz system. *Physica D: Nonlinear Phenomena* **408,** 132495. ISSN: 0167-2789. https://www.sciencedirect.com/science/article/pii/S0167278919307080 (2020).

25. Townshend, R. *et al. ATOM3D: Tasks on Molecules in Three Dimensions* in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks* (eds Vanschoren, J. & Yeung, S.) **1** (2021). https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/c45147dee729311ef5b5c3003946c48f-Paper-round1.pdf.

26. Du, Y. *et al. GraphGT: Machine Learning Datasets for Graph Generation and Transformation* in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks* (eds Vanschoren, J. & Yeung, S.) **1** (2021). https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/32bb90e8976aab5298d5da10fe66f21d-Paper-round2.pdf.

27. Freitas, S., Dong, Y., Neil, J. & Chau, D. H. *A Large-Scale Database for Graph Representation Learning* in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks* (eds Vanschoren, J. & Yeung, S.) **1** (2021). https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/5fd0b37cd7dbbb00f97ba6ce92bf5add-Paper-round1.pdf.

28. Freeman, D. *et al. Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation* in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks* (eds Vanschoren, J. & Yeung, S.) **1** (2021). https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/d1f491a404d6854880943e5c3cd9ca25-Paper-round1.pdf.

29. Gilpin, W. *Chaos as an interpretable benchmark for forecasting and data-driven modelling* in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks* (eds Vanschoren, J. & Yeung, S.) **1** (2021). https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/ec5decca5ed3d6b8079e2e7e7bacc9f2-Paper-round2.pdf.

30. Um, K., Brand, R., Fei, Y. (, Holl, P. & Thuerey, N. *Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers* in *Advances in Neural Information Processing Systems* (eds Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. & Lin, H.) **33** (Curran Associates, Inc., 2020), 6111–6122. https://proceedings.neurips.cc/paper/2020/file/43e4e6a6f341e00671e123714de019a8-Paper.pdf.

31. Mohan, A. T., Lubbers, N., Livescu, D. & Chertkov, M. *Embedding Hard Physical Constraints in Convolutional Neural Networks for 3D Turbulence* in *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations* (2020). https://openreview.net/forum?id=IaXBtMNFaa.

32. Wandel, N., Weinmann, M. & Klein, R. *Learning Incompressible Fluid Dynamics from Scratch - Towards Fast, Differentiable Fluid Models that Generalize* in *International Conference on Learning Representations* (2021). https://openreview.net/forum?id=KUDUoRsEphu.

33. Obiols-Sales, O., Vishnu, A., Malaya, N. & Chandramowlishwaran, A. *CFDNet: A deep learning-based accelerator for fluid simulations* in *Proc. ACM International Conference on Supercomputing (ICS)* (2020).

34. Gupta, G., Xiao, X. & Bogdan, P. *Multiwavelet-based Operator Learning for Differential Equations* in *Advances in Neural Information Processing Systems* (eds Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P. & Vaughan, J. W.) **34** (Curran Associates, Inc., 2021), 24048–24062. https://proceedings.neurips.cc/paper/2021/file/c9e5c2b59d98488fe1070e744041ea0e-Paper.pdf.

35. Wang, R., Kashinath, K., Mustafa, M., Albert, A. & Yu, R. Towards Physics-informed Deep Learning for Turbulent Flow Prediction. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2020).

36. Gori, M., Monfardini, G. & Scarselli, F. *A new model for learning in graph domains* in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* **2** (2005), 729–734 vol. 2.

37. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M. & Monfardini, G. The Graph Neural Network Model. *IEEE Transactions on Neural Networks* **20,** 61–80 (2009).

38. Li, Y., Tarlow, D., Brockschmidt, M. & Zemel, R. S. *Gated Graph Sequence Neural Networks* in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (eds Bengio, Y. & LeCun, Y.) (2016). http://arxiv.org/abs/1511.05493.

39. Kipf, T. N. & Welling, M. *Semi-Supervised Classification with Graph Convolutional Networks* in *International Conference on Learning Representations* (2017).

40. Sanchez-Gonzalez, A. *et al. Graph Networks as Learnable Physics Engines for Inference and Control* in *Proceedings of the 35th International Conference on Machine Learning* (eds Dy, J. & Krause, A.) **80** (PMLR, July 2018), 4470–4479. https://proceedings.mlr.press/v80/sanchez-gonzalez18a.html.

41. Cummings, R. M., Mason, W. H., Morton, S. A. & McDaniel, D. R. in, 731–765 (Cambridge University Press, 2015).

42. Center, N. L. R. *Turbulence Modeling Resource* https://turbmodels.larc.nasa.gov/. Accessed: 2022-05-19. 2021.

43. Charles L. Ladson, A. S. H. & William G. Johnson, J. Pressure Distributions From High Reynolds Number Transonic Tests of an NACA 0012 Airfoil in the Langley 0.3-Meter Transonic Cryogenic Tunnel. *NASA Technical Memorandum 100526* (Dec. 1987).

44. Ladson, C. L. Effects of Independent Variation of Mach and Reynolds Numbers on the Low-Speed Aerodynamic Characteristics of the NACA 0012 Airfoil Section. *NASA Technical Memorandum 4074* (Oct. 1988).

45. Wadcock, A. J. Structure of the Turbulent Separated Flow Around a Stalled Airfoil. *NASA Contractor Report 152263* (Feb. 1979).

46. Menter, F. R., Kuntz, M. & Langtry, R. Ten years of industrial experience with the SST turbulence model. *Turbulence, Heat and Mass Transfer* **4,** 625–632 (2003).

47. Selig, M. *UIUC Airfoil Database* https://m-selig.ae.illinois.edu/ads/coord_database.html. Accessed: 2022-08-22. 2022.

48. Weller, H. G., Tabor, G., Jasak, H. & Fureby, C. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in Physics* **12,** 620–631 (1998).

49. Caretto, L. S., Gosman, A. D., Patankar, S. V. & Spalding, D. B. *Two calculation procedures for steady, three-dimensional flows with recirculation* in *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics* (eds Cabannes, H. & Temam, R.) (Springer Berlin Heidelberg, Berlin, Heidelberg, 1973), 60–68. ISBN: 978-3-540-38392-5.

50. Doormaal, J. P. V. & Raithby, G. D. Enhancements of the SIMPLE Method for Predicting Incompressible Fluid Flows. *Numerical Heat Transfer* **7,** 147–163 (1984).

51. Ayachit, U. *The ParaView Guide: A Parallel Visualization Application* ISBN: 978-1-930934-30-6 (Kitware, 2015).

52. Sullivan, C. B. & Kaszynski, A. A. PyVista: 3D plotting and mesh analysis through a streamlined interface for the Visualization Toolkit (VTK). *Journal of Open Source Software* **4,** 1450. https://doi.org/10.21105/joss.01450 (2019).

53. Hamilton, W., Ying, Z. & Leskovec, J. *Inductive Representation Learning on Large Graphs* in *Advances in Neural Information Processing Systems* (eds Guyon, I. *et al.*) **30** (Curran Associates, Inc., 2017). https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

54. Charles, R. Q., Su, H., Kaichun, M. & Guibas, L. J. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation* in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), 77–85.

55. Gao, H. & Ji, S. *Graph U-Nets* in *Proceedings of the 36th International Conference on Machine Learning* (eds Chaudhuri, K. & Salakhutdinov, R.) **97** (PMLR, June 2019), 2083–2092. https://proceedings.mlr.press/v97/gao19a.html.

56. Sitzmann, V., Martel, J., Bergman, A., Lindell, D. & Wetzstein, G. *Implicit Neural Representations with Periodic Activation Functions* in *Advances in Neural Information Processing Systems* (eds Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. & Lin, H.) **33** (Curran Associates, Inc., 2020), 7462–7473. https://proceedings.neurips.cc/paper/2020/file/53c04118df112c13a8c34b38343b9c10-Paper.pdf.

57. Lindell, D. B., Van Veen, D., Park, J. J. & Wetzstein, G. BACON: Band-limited coordinate networks for multiscale scene representation. *arXiv preprint arXiv:0000.00000* (2021).

58. Brandstetter, J., Hesselink, R., van der Pol, E., Bekkers, E. J. & Welling, M. *Geometric and Physical Quantities improve E(3) Equivariant Message Passing* in *International Conference on Learning Representations* (2022). https://openreview.net/forum?id=_xwr8gOBeV1.

59. Harris, C. R. *et al.* Array programming with NumPy. *Nature* **585,** 357–362. https://doi.org/10.1038/s41586-020-2649-2 (Sept. 2020).

60. Paszke, A. *et al.* in *Advances in Neural Information Processing Systems 32* (eds Wallach, H. *et al.*) 8024–8035 (Curran Associates, Inc., 2019). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

61. Fey, M. & Lenssen, J. E. *Fast Graph Representation Learning with PyTorch Geometric* cite arxiv:1903.02428. 2019. http://arxiv.org/abs/1903.02428.

62. Landau, L. & Lifshitz, E. *Fluid Mechanics: Volume 6* ISBN: 978-1-4831-4050-6 (Elsevier Science, 2013).

63. Wilcox, D. *Turbulence Modeling for CFD (3rd Edition)* (DCW Industries, Incorporated, 2006).

64. Spalart, P. & Allmaras, S. A One-Equation Turbulence Model for Aerodynamic Flows. *AIAA* **439** (1992).

65. Launder, B. & Spalding, D. The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering* **3,** 269–289. ISSN: 0045-7825 (1974).

66. Wilcox, D. C. Formulation of the k-w Turbulence Model Revisited. *AIAA Journal* **46,** 2823–2838 (2008).

67. Schlichting, H. & Gersten, K. *Boundary-Layer Theory* ISBN: 978-3-662-52917-1 (Springer Berlin Heidelberg, Jan. 2017).

68. Smith, L. N. & Topin, N. Super-Convergence: Very Fast Training of Residual Networks Using Large Learning Rates. *CoRR* **abs/1708.07120.** arXiv: 1708.07120. http://arxiv.org/abs/1708.07120 (2017).