SAPIENZA
UNIVERSITÀ DI ROMA

Artificial Intelligence and Turbulence Modeling

Development of data-driven tools for RANS models in Turbomachinery Flows

Department of Astronautical, Electrical and Energy Engineering,
Sapienza University of Rome

PhD Candidate:
LORENZO TIEGHI

Supervisor:
PROF. ALESSANDRO CORSINI

Co-supervisor:
PROF. GIOVANNI DELIBRA

A dissertation submitted in partial satisfaction of the requirements for the Doctor of Philosophy degree in Energy and Environment, XXXII cycle

Oct. 2019

”To Antonio and Ester”

Contents

List of Figures

List of Tables

Abstract

To numerically simulate a turbulent flow through Computational Fluid Dynamics (CFD) has become a mandatory step in turbomachinery design and performance prediction. CFD users live in a constant compromise between accuracy of results and reasonable computational times. The most high-fidelity approach that is currently affordable is constituted by Large Eddy Simulations (LES), that grants the best representation of the physics of the flows. The tradeoff lies in the difficulty of generating proper inflow conditions, the complex numerical setups and the extreme grid refinements that are required. The latter is additionally worsened by the high Reynolds numbers that are normally involved in turbomachinery flows. Thus, the combination of long computational times with the complexity of the approach seriously impairs the applicability of LES simulations to internal flows.
On the other hand, Reynolds Averaged Navier-Stokes approach (RANS), constitute the workhorse in the CFD practice. The less restrictive grid requirements lead to faster computations and an increase in robustness. The stationary assumption provides information on the time-averaged fields, that suffice in most of the cases, especially in performance prediction. Drawback of the RANS approach is constituted by the turbulence closure of the turbulent kinematic viscosity term, i.e. a set of equations or parameters that model the physics of turbulence. Such closure necessarily embeds a source of errors. Popular two-equation closures, e.g. $k - \varepsilon$ or $k - \omega$, correlates the turbulent viscosity to the transport of the Turbulent

Kinetic Energy (TKE) and additional variables. A set of heuristically tuned coefficients are also included to maximize the effectiveness of the model. It is universally accepted that the previous hypothesis provide a fairly good agreement with physics of undisturbed flows, gradually losing consistency as long as we step away from the two-dimensional stationary flow assumption. RANS approach dramatically loses accuracy in presence of flows that are subject of unsteadiness such as adverse gradients of pressure, mixings, rotational forces or irregular geometries **??**. The correct prediction of the previous phenomena become critic in turbomachinery flows, where it is necessary to take in account their influence on the overall performance of the turbomachinery.
Many strategies have been exploited to increase the fidelity of RANS simulations, like adding additional equations, e.g. $v^2 - f$ model, including quadratic or cubic terms in the modeled Reynolds stress tensor (RST), e.g. Lien cubic $k - \varepsilon$, hybrid approaches, etc. Some of the previous solutions have been proven effective, but most of the RANS uncertainties remain yet unsolved. Despite that, we can conclude that research on the topic has reached a stagnation point, with no major breakthrough expected in the next years [1].
Machine-learning (ML) has found a deep consensus in the engineer community, becoming extremely popular in fields like robotics, maintenance and diagnosis. Nevertheless, the application of artificial intelligence to turbulence modeling is still a novelty, with the first comprehensive literature report only published in 2015 to Duraisamy et al [2]. In spite of the successful results, many questions remain unanswered and an exhaustive methodology has not been proposed so far.
After an introduction to the topic, the manuscript articulates in three different sections. In the first part, *"Turbulence Modeling in the Big Data Era"*, we report a deep literature review of works on the topic, classifying them in different categories, based on the methodology. To the best of the authors' knowledge, this is the first available attempt to organize the available contents. Three classes of approach were identified: field inversing, anisotropic modeling and derivative works.

1. Field inversing: a correction coefficient for the production or dissipation term in the TKE transport equation is inferred by means of ML.
2. Anisotropic modeling: the anisotropic part of the RST is corrected or predicted by neural networks / random forests. Turbulent invariants are exploited to this purpose.
3. Derivative application: isolated attempts or non-popular methodologies fall within this category.

We analyze each of the classes and provide a mathematical formulation of the methodology. We also highlight some key issues that are still to be solved.
In the second part of the manuscript, *"Machine-learning Tools and Techniques"*, after a brief historical introduction, we perform a dissertation on the currently available machine-learning algorithms. In particular, we focus on multi-layer multi-input neural networks that constitute our artificial intelligence further in the work. We give a mathematical representation of advanced non-trivial techniques, useful for our applications.
The third section, *"Development of Machine-Learning Assisted Tools for Turbulence Modeling"* treats three distinct applications of neural-networks to turbomachinery flows. The first two are focused on the creation of ML-assisted wall treatments for internal flows, while the last part is centered on the identification of stagnation zones in Gas Turbines (GTs) enclosure. Wall treatment is a crucial and non-trivial task in numerical simulations. In fact, CFD results can be considered reliable if the grid is fine enough for a proper resolution of the boundary layers. However, grid refinement comes at the cost of a larger mesh that reflects in increased computational time. To reduce the impact of wall refinements, wall functions are universally accepted to model part of the boundary layers, returning values of dissipation rate, TKE, turbulent viscosity, etc.. These functions were derived by direct observations of turbulent flows over flat plates and come as polynomial expressions depending from Reynolds number and wall distance. In so doing, every deflection of the flow from the flat plate assumption is not considered and it can become an

important source of errors.

A Data-driven Wall Function for Stationary Flows

ASME GT 2019-91238

RANS approach fails to predict separating flows, regardless of the source of separation, i.e. geometrical or pressure-induced. We therefore created a training database through LES simulations of canonical flows, namely a 2D periodic hill, channel flows and backward-steps at different Re numbers. Simulations were validated against data available in literature. An Artificial Neural Network (ANN) is trained in Python 3.6 to predict TKE values on the first layer of cells near the walls. A deep analysis of data preprocessing, ANN training and testing phases shows the feasibility and accuracy of the approach. The model has been implemented in an open-source environment, OpenFoam v18, as a custom boundary condition. During run-time operation of the solver flow fields are extracted, manipulated and forwarded to the algorithm, that eventually predicts TKE boundary values. Coupling between theC^{++} based solver and Python is realized through the C_{API} library. The framework is tested in two different cases and results from $k-\varepsilon$ model are compared with standard wall treatment. No significate increase in computational time is found in the ML-enhanced framework.
The first one is a two-dimensional periodic hill with three different grid resolutions, coarse, intermediate and fine. A similar geometry is included in the training dataset. ANN-based wall function corrects the overprediction of the model regardless of the mesh, showing that this approach is grid-insensitive. Reattachment length is compared with literature numerical results.
A cross-validation is performed through the analysis of the flow around NACA profiles in a cascade arrangement. Two version are considered, a standard airfoil and a modified profile with leading

edge bumps for stall control. While in the former results show no significative difference between the two wall treatments, in the modified airfoil the ANN-enhanced computations return more accurate results. In addition, flow fields and coefficient of pressure around the airfoils are reported.

A Data-driven Wall Function for Rotating Passages

ASME GT 2020-15353

The framework previously developed is here applied to rotating ducts. The effect of Coriolis and centrifugal forces are responsible of de- and stabilization of boundary layers. RANS approach, however, do not take in account rotating forces. In this work we create a wall treatment for rotating passages, commonly found in turbomachinery flows. A training database is created through LES simulation of a diverging diffuser with Rossby number = 0.41 and a diverging angle of 15deg., following experimental setup of Moore [3]. A comparison between LES, RANS and experimental results highlight the discrepancy of the RANS approach. Results from Exploratory Data Analysis (EDA) are reported. Data is filtered to obtain a zonal training with $y+<400$. EDA is used as a base to create input features that are also independent from the frame of reference. Training and Testing convergence histories show the quality of the training. The predictor returns a correction term for the TKE values on the first layer of cells. The derived model is tested with run-time computations of the same geometry with an inferior Rossby number, in cross-validation. On the last iteration of the solver, we report the prediction of the algorithm in different part of the domain. We report flow fields, accuracy and absolute relative errors for the *whole domain*, the *inner layer* and the *first layer* of cells. Results show the effect of the algorithm in the different zone of the domain, with an expected increase in accuracy as we move closer to the viscous sublayer.

Identification of Poorly Ventilated Zones in GT Enclosures

ASME GT 2019-91198

This part of the work has been conducted in cooperation with Baker Hughes, a GE company. Gas Turbines (GTs) ventilation systems of enclosures should grant the complete washing if gas leaking occurs. Given the extremely complex geometry that may lead to gas stagnation and the presence of high temperature surfaces, poor ventilation may lead to potential dangers. The aim is to build a model that, inferring from stationary fields provided by RANS simulations, classify portion of the computational domain through a Poorly Ventilated Index (PVI). PVI ranges from 1-*safe* to 16 – *extremely dangerous* and is based on the product of two term, function of temperature and concentration. The temperature term can be directly computed through numerical simulation, while the concentration of methane requires several days of URANS computations. The classifier therefore predicts the level of danger associated with the concentration of methane, reducing time requirements from several days to few seconds.
A training database is generated through multi-phase simulations of the washing process of simplified geometries. Test geometries include high Reynolds computations of plain channel flow, backward facing step, and 2D sinusoidal hill flow, at different Reynolds number, axisymmetric hill flow and confined cylinder in cross-flow. Input features are constructed from the flow variables to assure independence from frame of reference and locally normalized. The ANN is trained to solve a four-class labeling problem, mapping local fields to the corresponding level of methane concentration. The algorithm was tested on all the training case, showing an extremely high accuracy (>95%).
The model was validated on a model geometry of the GT enclosure.

ABSTRACT

Temperature, concentration and velocity fields for different sampling planes are reported. A comparison between PVI levels given by the full URANS simulations and those derived by the RANS-derived classifier show a final accuracy of 91.3%.

Chapter 1

Introduction

"Nature is the source of all true knowledge. She has her own logic, her own laws, she has no effect without cause nor invention without necessity."

Leonardo Da Vinci

Since the very origin of mankind, human beings looked at universe and asked themselves questions. Those problems were usually not easily solved, and sometimes answers were given several centuries later, or are still part of an open debate. Today, physics has provided an explanation, under the form of theories and models, for the majority of the classic problems that can be found in nature. However, scientific progress is all but at a stagnation point, thanks to the capability of pushing human observation to unimaginable scales, from atoms to entire star clusters.

The work here proposed is mainly focused on the study of tur-

bulence,*"the most important unsolved problem of classical physics"*[1]. It embeds an extremely non-linear and time dependent chaotic nature [4]. Turbulence happens in a wide range of different scales, from the atmospheric mesoscales, of hundreds of kilometers, to the nanoscales of the smallest eddies. Complete, or at least satisfying, modeling of turbulent flows is far from be achieved,thus it can still be considered as an hot topic in classic physics [5].
Further in this dissertations we will focus on turbomachinery internal flows, where the need a robust representation of turbulence is coupled with the challenging analysis of complex flows.

1.1 Internal Flows in Turbomachinery

Internal flows are distinguished from external flows due to the strong mutual interactions in the shear layers that the former presents. When trying to model such flows, ad-hoc solutions are usually exploited [6]. An example of internal flow is provided by the field in a real compressor stage (Figure 1.1 from [7]). It is universally considered as one of the most complex flows to analyze, especially in a centrifugal compressor where the effect of rotation is predominant. It is evident that such complexity cannot be taken into account by the use of blade element theory.

Bradshaw defines complex turbulent flows as those that *"cannot be adequately treated by calculation methods developed for simple thin shear layers"*[8]. Sources of deflection from the unperturbed shear layers are multiple and coexist in the same scenario (e.g. rotating passages, mixing, sudden abrupt, separations).
Even the simplest internal flow can present a non-trivial behavior. As a further example, we report the flow within a constant area duct with an inlet section constituted by a short curved nozzle (Fig. 1.2 from [9]). Boundaries of the zones are not well defined in reality, as there is no harsh transition between them. As we move downstream, from zone I to IV, the interaction between boundary layers becomes stronger and stronger. In zone I, we can observe that

[1]Richard P. Feynman - The Feynman Lectures on Physics (1964)

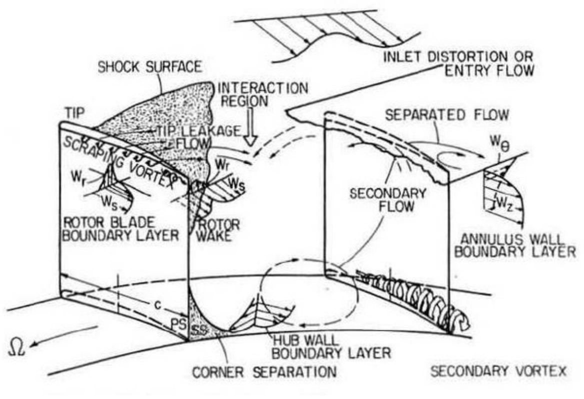


Figure 1.1: Nature of the flow in an axial flow compressor rotor passage

the core of the flow in the interior is not affected by the thin wall shear, whereas in zone III the maximum interaction is reached, and no region of inviscid flow can be found. At more than 20 diameters from the inlet, zone IV, the flow can be considered *fully developed*, i.e. flow field is no longer a function of the streamwise coordinate. In a real scenario, a fully developed flow is hardly achieved, as even ducts that are theoretically long enough are affected by inlet distortion/fluctuation. Unfortunately, flows in turbomachinery are not as easy to predict.

Turbomachinery flows can be roughly defined as a particular class of internal flows where the interaction between solid walls (rotating or not) and the fluid is exploited to perform energy exchange, in both directions. Turbomachinery flows share common features that uniquely determine their complex nature [10]. Within the most relevant we can find:

- Unsteadiness, as every flow can be characterized by fluctuations, instabilities, waves, etc. that affect the output and the overall performance of of the system.

- Coriolis and centrifugal forces are responsible of strong de-

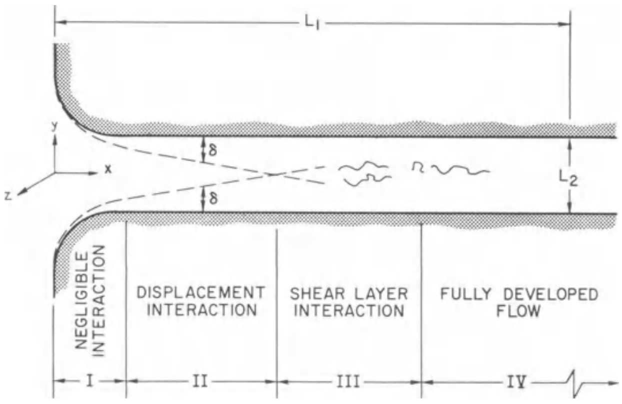


Figure 1.2: Simple internal flow development

flection of the streamlines.

- Absence of uniform free stream.
- Presence of work or heat transfer.
- Full three dimensionality of the flows, that cause a transport of momentum, mass and energy in all the directions.
- Secondary flows, caused by the geometry or by the interaction of the flow with moving surfaces.
- Strong swirls.
- Wall bounded flows.

A complete representation of turbomachinery flows is probably impossible, as the entire flow cannot be solely seen as a superimposition of the previous effects, due to their mutual interaction. Not by chance, in the design practice most of the complexity is modeled through loss functions or empirical corrections, that ease the process at the expense of accuracy/generality [11]. Two strategies are currently exploited to study turbomachinery flows, through

experimental observations or numerical simulations.
As the quote at the beginning of the chapter highlights, the formulation of fluid mechanics theory has always been based on the direct observation of natural phenomena. Modern theories can be considered as the direct heir of the 18th century models [12]. Since the development of CFD, the second half of the 20th century and on, a long debate regarding the best approach has arisen. The answer, of course, is yet far and it would probably never be discovered.

1.2 Experimental Fluid Dynamics

One of the key arguments that supports the superiority of experimental observations is that experiments capture the ground truth of the flow. This ability is also a strong improvement on what happens in numerical simulations, as no arbitrary assumption or model is naturally embedded in the process. In reality, experimental tests on turbomachinery present challenges that are not so easy to overcome. First, both time-resolved and time-averaged measurements are necessary to fully capture flow behavior. Unfortunately, no experimental technique is able to provide all information that are needed at once. Measurements with different levels of complexity and detail are combined together, from qualitative flow visualizations (smoke injection) to time-resolved measurements (hot wire) [13]. As Simoneau et al. pointed out, *"ultimately, all levels* (of complexity ndr.) *are needed to do the job. The challenge is to choose the right tool for the right job."*. A brief overview of the two classes of techniques (see [14]) is reported below.

Non-optical measurements, that includes five-hole probes, Pitot tubes and pressure transducers, are exploited to measure wall shear stress, static pressure and to visualize surface flows. The implementation is tricky as they must be inserted in a rotating frame. Extensive data post-processing is also required to isolate the contribution of a single phenomenon from global instabilities.

Optical measurements, include laser Doppler and particle image velocimetries. Such technologies can be implemented also in a rotating frame, however the main deficiency lies in the requirement of an optical access to the interior of the machine. This need seriously cripple their capability in real scenarios.

Once these difficulties have been overcome, a collection high fidelity observations is obtained.

1.2.1 Computational Fluid Dynamics

Computational Fluid Dynamics (CFD) comes in great aid for turbomachinery performance prediction and flow investigation, and it is universally accepted as the main tool for research, design and optimization. Theoretically, it is be possible to entirely build a virtual test rig, as shown by [15][16], to cover each aspect of the design process through CFD. The extreme popularity of the numerical approach, both in academia and industry, is given by the savings in terms of time and costs. The capability of simulating almost every geometrical configuration, in fact, reduces the necessity of expensive prototype construction and testing.
As previously highlighted for experimental measurements, numerical simulations comes with different level of complexity and approximation, with the Large Eddies Simulations (LES) at the apex and the quasi-1D equations at the bottom. Direct Numerical Simulations (DNS) are not included in the standard practice as high Reynolds numbers that are normally involved in turbomachinery flows lead to prohibitive computational costs. The phenomena under investigation define the complexity of the approach, as to the best accuracy corresponds the highest computational cost. If the overall performances of the machine are solely needed, fast approaches are preferred, such as RANS or lower dimension approximations. Transient, unstable, small-scale or fluctuating features, however, can only be captured by models of higher level, with all the complexity that results. The state of art, LES approach, is confined to very specific problems, with RANS and Unsteady-RANS covering the majority of numerical simulations. Lower level ap-

proaches are usually limited to fast prototyping or optimization problems [17][18].
Nevertheless, in contrast to the aforementioned advantages, CFD cannot entirely substitute experimental testing. In fact, CFD is not an exact science and several aspects are modeled or approximated. Even the simple assumption of steady flow in a RANS simulation can lead to seriously misleading results [19]. Frequent non-trivial sources of error can be ascribed to:

- Incorrect modeling of a phenomenon (e.g. low accuracy turbulence model)
- Uncertain conditioning of the problem (e.g. unknown BCs)
- Steady flow assumption
- Discretization errors (e.g. computational grid, numerical schemes)

Great efforts have been made to overcome part of the previous problems, and this dissertation can be considered as one of the trials focused on improving existing turbulence models.

1.3 Learn by Observations

Figure 1.3 from [20] shows the process of *learning by observations*. This flow of ideas has not drastically changed since the start of the scientific method (see [21]). First, a clarification of the modern meaning of *observation* is mandatory. Historically, in fact, the term experiment and observation represented different concepts, even if closely intertwined [22]. Observations were experience caught by eyes, thus subjective, prone to arbitrary interpretation and human errors. In a modern fashion, we define observations as a collection of measured variables (*observables*), either as data from the physical (real) world or obtained through computer simulations. By so doing, scientists put themselves in the condition of ”perfect observers” as no arbitrary assumption is taken in the process.

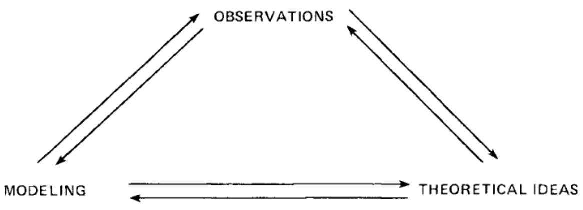


Figure 1.3: Modeling, observations and theoretical ideas

Once that enough data has been collected, *theoretical ideas* are created. We define as theoretical ideas the domain where observations are mapped, i.e. a mathematical representation of the observations is provided . *Models* eventually organize theoretical ideas in a coherent structure. Everything is embedded in a constant loop, where new observations discuss an existing model, and the model itself demands new heuristic proofs. Obviously, the key of such a process is given by the capacity of transposing observations into ideas, and only modeling comes only at the end. This flow of ideas is entirely human-driven, as the success or failure of the process is decided by the sole ability of the observer. However, questions arise when trying to model complex problems.

Observations

- *Are our experiments free of errors?*
Numerical simulations and experimental tests are not immune to error from any source. Fidelity of observation is the key ingredient to avoid misleading results.

- *Are we observing all the significant variables?*
To properly define complex phenomena, all the factors of influence must be taken in account. Turbulence, for instance, need both a spatial and a temporal representation.

- *Is our space of experiments large enough to capture the full be-*

havior of the phenomenon?
If the final aim is to derive a model, generality is the feature that the modeler should be looking for. A model should work for at least a small restriction of setups. The more general, the better.

Theoretical Ideas

- *Are we looking the data from the right perspective?*
Relationships and correlations can be hidden in the raw observables. Changing perspective, e.g. moving to the plane of phases, can significantly ease the creation of a model.

- *Is our data statistically significant?*
Duplicated, redundant and sparse data impair model effectiveness. A trade-off between the limited amount of data that one can consider at once and proper statistical representation must be found. However, knowing the boundaries of the model, a complete representation of the phenomenon can be provided.

Modeling

- *Are we using the proper mathematical representation?*
Within the family of available statistical and mathematical tools, a selection of the instrument that best fit our purpose is not trivial. Unfortunately, high accuracy demands also a complex structure of theoretical ideas, sometimes even higher than the affordable.

- *How do we improve the model?*
Improving an existing model is a challenging task. First, a deep knowledge of all the limitations of the model is required. Second, one should be able to correct model deficiencies without impairing its successes.

Even if we are able to give an answer to all the previous questions, the model would not be capable of entirely reproducing the ground truth underlying the phenomenon. In turbomachinery existing flow models present such discrepancies mainly due to:

1. The limited number of observations that are considered at once (many of the models are tuned on half-a-score flow topologies at best);

2. Difficulties encountered in representing the chaotic nature of turbulence;

3. The need to limit computational time and costs required to solve the problem;

Despite the daunting challenges, great advancements have been made in the past sixty years. More recently, particularly in the last 10 years, big data techniques have become more and more popular, leading turbomachinery flow modeling into a new direction.

1.4 Learn by Data

Big data is *"high-volume, high-velocity and/or high-variety information assets"*[2]. As evident from the historical background, the big data approach is not naturally suited for physics, being born for social media and economic studies. This also means that extra care must be taken when applying existing tools to a field of interest that is far from the initial design intention. However, the statistical approach that lies in the background is still robust and it ensures that a big data approach can be proficient as long as enough data is provided [23]. Further in the dissertation we will provide an overview of the most popular techniques.
Under the previous assumptions, the usual learn by observation approach now becomes *learn by data*. The standard scheme of Figure 1.3 can be therefore transformed, thanks to the big data perspective, to something closely related but extremely more powerful.

[2]Gartner's Master Data Management - 2018

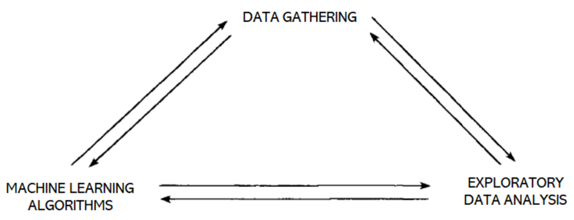


Figure 1.4: Relationships between the realms of learn by data

The collection of raw observations is substituted by *data gathering*. It must be highlighted that the two are somehow coincident, with few remarkable differences. One is surely the amount of observables that are considered: we move from few thousands for the former to several billions of the latter. Also different types of data (e.g. visual, audio, measurements) concur to create *data lakes*, repositories of organized and unorganized data of any size [24]. Data gathering is not static but dynamic: observables form a constant stream of data that feeds repositories. Hence, models can be periodically improved and adjusted as the process of data gathering goes on.
The *exploratory data analysis*, abbreviated with EDA, is the correspondent of theoretical ideas. Through statistical tools, in fact, EDA transform every pack of data of any kind and from any source into a mathematical representation. Modeling is probably the realm that is mostly affected by the new environment, as modelers have access to a completely new family of powerful algorithms and tools. To sum up we enhance the standard learn by observations approach, as we look at more data at the same time and we model it better, by making use of statistical tools.

1.5 Outline of the Dissertation

This dissertation is constituted by four different chapters.

1. *Turbulence Modeling in the Big Data Era*.
   We first perform a general classification of the predominant methodologies involving turbulence modeling by means of machine-learning. We mainly focus on *field inversing* and *anisotropic modeling*. In the second part, we report an overview on the relevant prior works on the topic.

2. *Machine-learning Tools and Techniques*.
   In the first section, we summarize the state of art of machine-learning algorithms. The aim is to justify the use of neural networks and we report the major competitor algorithms. A mathematical formulation of the multi-layer perceptrons then follows. The last part is dedicated to the description of advanced techniques for turbulence modeling.

3. *Development of Machine-Learning Assisted Tools for Turbulence Modeling*.
   This chapter proves the merit of application of machine-learning to two-equation models. It is organized in three different sections, following prior publications from the author:

   - A Data-driven Wall Function for Stationary Flows;
   - A Data-driven Wall Function for Rotating Passages;
   - Identification of Poorly Ventilated Zones in Gas-Turbines Enclosures with Machine-learning.

4. *Conclusions*
   We try to provide answers to some of the research open issues.

Chapter 2

Turbulence Modeling in the Big Data Era

”All models are wrong, but some are useful.”

George Box

This chapter focuses on the novel vogue in turbo-machinery that resulted from the A.I. revival. Quoting Box (1987 [25]), ”every model is just an approximation of the *whole truth*”. Turbulence models do not constitute an exception to this rule. Thus, the question we need to ask is not if a model is true (as it never is) but if it fits the particular application that is designed for [26].

2.1 Turbulence: A Big Data Perspective

Two farsighted reports from Slotnik and his coworkers [27][1]. investigated a future perspective of traditional RANS modeling, highlighting that despite all the efforts of the scientific community in the past years, a stagnation point has been unavoidably reached. On one side pros and cons are perfectly known and understood, while on the other very few practical solutions have been proposed to date to increase the effectiveness of existing turbulence models. A model where generality, robustness and cheapness coexist has not been developed yet. As such, in the standard CFD practice model selection is driven by the problem to solve (e.g. internal flows, transient, heat transfer). In fact, various options are usually available to the CFD practitioners, and only a toe to toe comparisons between different approaches decides the best one.
By looking at turbulence modeling through a big data perspective, however, new approaches become available. The growing availability of high fidelity data, e.g LES/DNS simulations and large experimental programs, in combination with the development of efficient statistical inference algorithms, have opened the path to innovative solutions.
Figure 2.1 gives an overview on the topics of the following sections.

Motivations behind this classification are evident if we consider each of previous the categories.

- ***Uncertainty quantification***: the approach tries to enhance RANS modeling through considering uncertainties and inadequacies that come from various sources, such as parameters or numerical approximations;

- ***Calibration***: by means of statistical inference model a best fitting suit of coefficients is eventually derived for specific applications.

- ***Machine learning***: machine-learning algorithms and methodologies are applied to directly infer from high fidelity flow

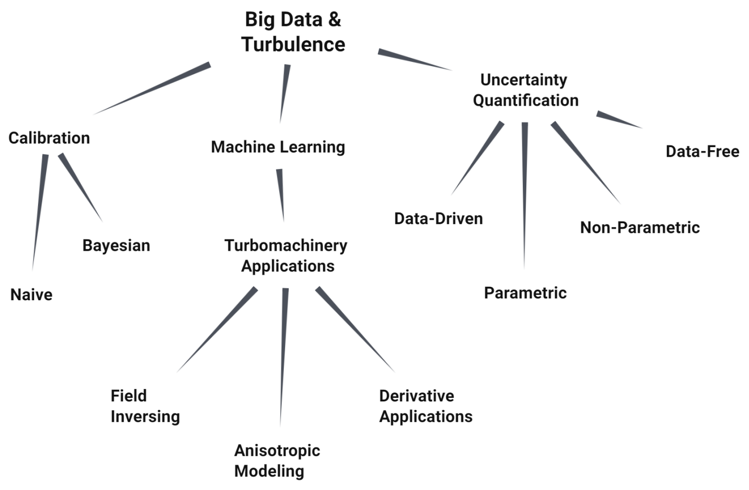


Figure 2.1: Turbulence modeling under a big data perspective.

simulations. We can further classify those efforts as:

1. *Field inversing*: a spatially varying field is included in the model and eventually calibrated. The aim of this field is to reduce the discrepancy between LES/DNS simulations and RANS. A machine-learning algorithm is trained to infer the correct field, using flow variables as input features;

2. *Anisotropic modeling*: the anisotropic part of the Reynolds stress tensor is modeled trough means of ML algorithms;

3. *Derivative approach*: in this field fall all the works that tries to indirectly extract information from the fluid dynamic fields, to various extents. In this case, the final goal is not to derive a model for turbulence, but to ex-

ploit machine-learning algorithms to discover hidden relations. They are included in the dissertation as they share common features and problems with the following classes.

2.2 A General Form for Data-Driven Turbulence Models

The former taxonomy draws strong boundaries, as the approaches and aims completely differ between each other. However, it is possible to recognize a shared underlying mathematical structure. To the best of the author's knowledge, the first attempt to represent this relationship can be attributed to Duraisamy, Iaccarino and Xiao [28]. This novelty is especially impressive if we consider its contrast with the secular development of the study of turbulence.
A general data driven model can be written as:

$$\widetilde{M} = M\left(\boldsymbol{w}; P\left(\mathbf{w}\right); \boldsymbol{c}; \boldsymbol{\theta}; \boldsymbol{\delta}; \boldsymbol{\epsilon_\theta}\right) \tag{2.1}$$

where:

- $\widetilde{M}$: the computational model, as a function of an array of independent variables w and based on a set of algebraic or differential operators P;
- c: a set of parameters of the model, the primary target of the modeling;
- θ: the data that comes with an uncertainty ϵ_θ;
- δ: a discrepancy between actual data and the data represented through the model.

Usually, the final goal of the model is predicting an interest feature or variable $q(\widetilde{M})$. The algebra in Eq. 2.1 facilitates the identification of the approximations embedded into a RANS formulation:

$$\boldsymbol{q} = \boldsymbol{q}\left(\boldsymbol{N}\overline{(\cdot)}; \boldsymbol{M}(\boldsymbol{w}; \boldsymbol{P}(\boldsymbol{w}); \boldsymbol{c})\right) \tag{2.2}$$

Four levels are naturally embedded, each contributing to the uncertainty of the model. In order of generality (with tuning being the most specific) we can find:

$A1$: *Time averaging*, that is intrinsic of the methodology. It automatically implies that the Navier-Stokes equations (indicated hereafter $N(\cdot)$ =0 need to be closed somehow. This is caused by the fact that $\overline{N(\cdot)} \neq N(\overline{\cdot})$, where the overline operator points to the temporal averaging.

$A2$: *Closure of NS equations*, as we have $\overline{N(\cdot)} = N(\overline{\cdot}) + M(\cdot)$, where $M(\cdot)$ is just the divergence of the Reynolds stress tensor $M = \boldsymbol{\nabla} \cdot \tau$. M is a function of a series of independent, averaged variables $\boldsymbol{w}$, that leads to the various closures.

$A3$: *Choice of* $\boldsymbol{w}$, obviously not all the variables can be included in the set of equations of the closure, here denoted with $P(\cdot)$. In so doing, $\boldsymbol{M}(\boldsymbol{w})$ can also be seen as $\boldsymbol{M}(\boldsymbol{w},\ P(\boldsymbol{w}))$. The physics of the problem and computational power available define the complexity of $P(\cdot)$ [29].

$A4$: *Tuning of* $P(\boldsymbol{w})$, that depends on a series of coefficients c. Calibration of the model is completely arbitrary, and it is usually performed to ensure that the model is working on all the simplest configurations (e.g. flat plate, channel flows, cylinder) [30].

To conclude, a general formulation for a predicted quantity is:

$$\overline{M} = M\left(w : P(w); c(\Theta); \delta(\Theta, \eta); \epsilon_{\Theta}\right) \tag{2.3}$$

The following sections are dedicated to analyzing each of the categories in Figure 2.1, highlighting the state of art and further developments.

2.3 Uncertainty Quantification

A complete and deep analysis of UQ techniques is far from the author's intents, however a brief overview is beneficial. The uncer-

tainties described in the previous section are also called *epistemic* or *model-form* uncertainties. An additional source of uncertainty can be derived from the variability of the input of every numerical simulation, like unknown boundary conditions, also called *aleatory* uncertainties. The final aim of the UQ is to study how the coupling of the two sources affects the output.
Looking at Eq. 2.1, the first step is to describe ϵ in a probabilistic form. The error is *propagated* through the model M and the final prediction becomes a function of the error $q(M, \epsilon)$. In so doing, the final output is no longer deterministic but stochastic, hence we derive a distribution of the predicted values $P(q)$. Current approaches to UQ can be classified into two main classes: parametric and non-parametric. In parametric approaches, all the uncertainties are introduced in the turbulence closure coefficients c. Such approach eases the process, as the implementation in popular CFD solvers is trivial. On the other hand, non-parametric approaches analyze the contribution to uncertainty of the modeled terms, e.g. turbulent viscosity, Reynolds stress tensor.

$$\boldsymbol{q} = \boldsymbol{q}\left(\boldsymbol{N}\overline{(\cdot)}; \underbrace{M(w; P(w)}_{\text{Non-parametric}}; \underbrace{c}_{\text{Parametric}})\right) \tag{2.4}$$

Non-parametric approaches are generally physics-driven and more prone to capture flow instabilities. However, computational costs, complexity and implementation difficulties present challenges. A further classification can be performed based on *data-free* (forward) or *data-driven* (backward) approaches. The former propagates the uncertainties and eventually studies the statistical distributions of the outputs. Instead, the latter infers errors and model coefficients from available data.

2.4 Model Calibration

All the turbulence closures, even the simplest algebraic, rely on some empirical coefficients. Those coefficients have been derived

Table 2.1: Classification of popular works on UQ and RANS modeling, from [31].

	Non-parametric	**Parametric**
data-free	Turgeon et al. [32], Dunn et al. [33], Platteuw et al. [34], Margeri et al. [35], Shaefer et al. [36]	Emory et al. [37, 38], Iaccarino et al.[39], Mishra and Iaccarino [40], Edeling et al. [41]
data-driven	Edeling et al. [42, 43], Cheung et al. [44], Kato et al. [45, 46]	Wang et al. [47], Singh et al. [48], Xiao et al. [49], Wu JL et al. [50, 51]

from direct observations of canonical flows, granting physics-consistency of the model in simplified scenarios. For example, citing the original $k-\varepsilon$ formulation from 1972, *"in order to estimate the form of the function f_μ, attention was first confined to the prediction of constant stress Couette Flows"* [52]. In so doing, a model is first calibrated on few canonical flows and later applied to every possible flow configuration.
Many works show the methodology that extends model effectiveness, embedding flow physics that is not included in the standard formulation, e.g. [53, 54, 55]. The non-trivial goal is to infer a new set of coefficients by the observation of a number of flows during the adjustment process, with the expectation of not worsening the model in a more general scenario. This aspect strongly reduces the feasibility of the so called *naive model calibration* (NMC), as on one hand the adjustment process requires a lot of high fidelity data, while on the other hand the improvement that is achieved is strictly limited to the physics described by the data.
For this purpose, Bayesian model calibration (BMC) has been developed, not so far from the standard approach. In NMC the source of errors is completely ascribed to the model coefficients, therefore

the model can be seen as:

$$\widetilde{M} = M\left(\boldsymbol{w}; P\left(\mathrm{w}\right); \widetilde{\boldsymbol{c_q}}\right) \tag{2.5}$$

where the goal is to improve model predictions $\mathbf{q}$ only by finding better $\widetilde{\boldsymbol{c_q}}$.

In a more general approach, BMC is a form of statistical inference where additional terms in the Eq. 2.5 are included:

$$\widetilde{M} = M\left(\boldsymbol{w}; P\left(\mathrm{w}\right); \widetilde{\boldsymbol{c_\theta}}\right) + \boldsymbol{\theta} + \boldsymbol{\epsilon_\theta} \tag{2.6}$$

By so doing, two approximation sources are additionally included in the formulation: uncertainties in data and the misfit between model prediction $\boldsymbol{\delta}$ and the data.
Here we will provide a mathematical representation of BMC. With $\boldsymbol{C}$ we denote the collection of n model coefficients $\boldsymbol{C} = \{C_1, ..., C_n\}$. From high fidelity data we extract a vector $\boldsymbol{y_e}$ of experimental observations, sampled at a set of N_p locations. The initial set of coefficients provides a prediction $\boldsymbol{y_m}(\boldsymbol{C})$. That prediction differs from the ground truth due to errors, either modeling or structural, that can be summed in a term $\boldsymbol{\epsilon}$, so that $\boldsymbol{y_e} = \boldsymbol{y_m}(\boldsymbol{C}) + \boldsymbol{\epsilon}$. We assume that the errors are uncorrelated within the different probing locations, following a centered Gaussian, i.e. $\boldsymbol{\epsilon} = \{\epsilon_i\}, \epsilon_i \sim N(0, \sigma^2)$. The σ^2 variance is a direct measure of the data-model misfit.
Let $P(\boldsymbol{C}, \sigma^2 \mid \boldsymbol{y_\epsilon})$ be the joint probability density function of the parameters and the model-data misfit on the observed data. Let $\Pi_1(\boldsymbol{C})$ and $\Pi_2(\sigma^2)$ be the initial guess of the distribution of $\mathbf{C}$ and σ^2. The likelihood $\boldsymbol{L}(\boldsymbol{y_\epsilon} \mid \boldsymbol{C})$ of observing $\boldsymbol{y_\epsilon}$, given $\mathbf{C}$ is:

$$\boldsymbol{L}(\boldsymbol{y_\epsilon} \mid \boldsymbol{C}, \sigma^2) \propto \frac{1}{\sigma^{N_p}} \exp\left(-\frac{\| \boldsymbol{y_e} - \boldsymbol{y_m}(\boldsymbol{C})\|_2{}^2}{\sigma^2}\right), \tag{2.7}$$

with $\| \cdot \|_2$ the $l^2 -$ norm of the vector. Following Bayes' theorem the calibrated distribution can be seen as:

$$\begin{aligned} P\left(\boldsymbol{C}, \sigma^2 \mid \boldsymbol{y_e}\right) &\propto \boldsymbol{L}\left(\boldsymbol{y_\epsilon} \mid \boldsymbol{C}, \sigma^2\right) \Pi_1\left(\boldsymbol{C}\right) \Pi_2\left(\sigma^2\right) \\ &\propto \exp\left(-\frac{\| \boldsymbol{y_e} - \boldsymbol{y_m}(\boldsymbol{C})\|_2{}^2}{\sigma^2}\right) \Pi_1\left(\boldsymbol{C}\right) \Pi_2\left(\sigma^2\right) \end{aligned} \tag{2.8}$$

$P\left(\boldsymbol{C}, \sigma^2 \mid \boldsymbol{y_e}\right)$ can be reconstructed by means of kernel density estimation. The larger the collection of $\{\boldsymbol{C}, \sigma^2\}$, the better the determination of the density function. Once that the density function has been calculated, the maximum likelihood of a model parameter is found, and it assures that the difference between observations and predictions $\boldsymbol{y_e} - \boldsymbol{y_m}(\boldsymbol{C})$ is minimized. By taking the mode of the posteriori probability, the corresponding calibrated model becomes deterministic. Potentiality and limits to this approach are highlighted in [56, 57].

2.5 Machine Learning

"Machine learning is the science of getting computers to act without being explicitly programmed". C. Bishop in this quote from [58], summarize the core of every machine-learning approach. By means of machine-learning algorithms data is firstly observed and parsed, afterwards transformed into a training database and eventually predictions about something in the world are made. Challenges and expectations from this approach are presented hereafter.

2.5.1 Key Issues

Despite the novelty of the topic, several key issues have already been highlighted [59].

1. Predictive standard models come as sets of equations that are a combination between observed physics and empirical correlations. It is not trivial to define where and if ML might outclass a standard representation.

2. Investigations on the topic, see [60] as example, proves that even a term of equations that is modeled almost exactly could lead to poor prediction. The balance of equations, and not only a single term, should be considered when data-modeling.

3. It is not always possible to directly use data from high fidelity simulations or experiments, as lower level models may live in

different scales of variables.

4. The information being extracted can be defined only in terms of the turbulence closure. For example, is not possible to directly apply data-mining techniques from DNS data to directly predict a field in a RANS simulation, as some information from the latter must be retained.

5. A general form of the models must be derived. Local corrections, in fact, have few practical uses, mainly restricted to optimization purposes.

6. ML should correct the deficiencies of the standard model only, and not impair it where it is working properly.

7. A ML approach need to be at least as robust as the standard model, especially if implemented in run-time computations.

2.5.2 Field Inversing

Field inversing and turbulence modeling, hereafter denoted with FIML, is currently a paradigm of machine-learning enhanced turbulence modeling. The underlying concept is that one should model a correction form for the model instead of a complete new form. It is performed through the inclusion of a correction field $\beta(x)$ to the transport equation, where x is the vector of spatial coordinates. In so doing, a ML algorithm is trained to minimize the discrepancy between an output of the high fidelity data $\mathbf{G_f}$ and the same output in the model to correct $\mathbf{G_{fi}}$, $\min(||\mathbf{G_f} - \mathbf{G_{fi}}||)$. $\boldsymbol{G}$ is usually expressed in terms of aero or thermodynamic properties. The field β is discretized in the whole computational domain as any other variable, leading to a local correction of the standard model.
An ensemble of problems $\mathbf{G_f}^{\mathbf{1}}$, $\mathbf{G_f}^{\mathbf{2}}$.. will be used as training database. Infering from those data sets is just the first step of the modeling, because β needs to vary as the inputs change, even in configurations not included in the starting $\mathbf{G_f}^{\mathbf{i}}$. Also, the dependence on x leads to a poor-conditioning of the problem, as geometries and frames of reference change dramatically within different simulations. It is

therefore necessary to find some input variables that are geometry-independent. Duraisamy expresses this process as *"converting the inference into modeling knowledge"* [2]. Thus, the domain of the inputs of β is no longer constituted by the spatial coordinates but it is substituted by the feature space η, $\beta(x) \rightarrow \beta(\eta)$. η is a function of flow variables, preferably locally normalized into non-dimensional quantities. In synthesis, raw variables are extracted from the flow, converted into input features η, fed to the ML algorithm that derives the correction field β.
Once that β has been defined, we are also interested in characterizing the impact of uncertainties in the data, model inadequacies and existing knowledge, on the inferred outputs. The following mathematical formulation follows the Bayesian approach [61]. The posteriori probability distribution $q(\beta|d)$, given a priori distribution $p(\beta)$, a data vector d, and a likelihood function $h(d|\beta)$ can be expressed as:

$$q(\beta|d) = \frac{h(d|\beta)p(\beta)}{c} \tag{2.9}$$

where $c = \int h(d|\beta)p(\beta)d\beta$. The value of β is inferred at every point of the computational domain of the RANS grid. To ease computations, $p(\beta|d) = e^{-(\beta-\beta_{\text{prior}})^T C^{-1}{}_{\text{prior}}(\beta-\beta_{\text{prior}})}$ and $h(d|\beta) = e^{-F^T C_{obs}{}^{-1} F}$ are considered Gaussian. F is a vector i elements long with $f_i = d_{i,RANS} - d_{i,hfidelity}$ the i^{th} element.β_{prior} is the prior mean of the parameters, corresponding to the base model. Under the Gaussian assumptions, the maximum a posteriori estimate (MAP) is taken to be representative of the mean of distribution. The MAP can be computed by maximizing the numerator of Eq. 2.9, or alternatively solving:

$$\beta_{\text{MAP}} = \arg\min J(\beta) = \arg\min \frac{1}{2}\left[F^T C_{obs}{}^{-1} F + (\beta - \beta_{\text{prior}})^T C^{-1}{}_{\text{prior}}(\beta - \beta_{\text{prior}})\right] \tag{2.10}$$

The resulting optimization can be solved using adjoint-driven techniques, but the posteriori distribution is not determined by sampling methods and instead is approximated by linearizing about the MAP point. By so doing, we can express the covariance as the inverse of the Hessian of the objective function $J(\beta)$:

$$C_{\text{posteriori}} = \left[\frac{d^2 J(\beta)}{d\beta d\beta}\right]^{-1} \tag{2.11}$$

To summarize FIML, it happens in two steps:

- Solution of a deterministic optimization problem with the objective function $J(\beta)$, through means of gradient based method, to obtain the MAP model.
- Building an approximation of the posteriori covariance by the inverse of the Hessian around the MAP point.

2.5.3 Anisotropic Modeling

In a RANS formulation three stresses appear in the momentum equation:

$$\underbrace{\mu\left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\right)}_{\text{Viscous}} - \underbrace{P\delta_{ij}}_{\text{Isotropic}} - \underbrace{\rho\overline{U_i U_j}}_{\text{Apparent}}, \tag{2.12}$$

where P,μ_t and ρ are the thermodynamic pressure, the dynamic viscosity and density respectively, see [62] for reference. The *additional stress* $\rho\overline{U_i U_j}$ acting in the mean turbulent flows and is much larger than the viscous counterpart. The apparent stress arises from momentum transferred by the fluctuating velocity field.
The Reynolds stress tensor is a 2^{nd} order symmetric tensor with the element on the diagonal representing the normal stresses and the off-diagonal the shear stresses. Eq. 2.12 can also be written as:

$$\overline{\tau_{ij}} = \nu_t S_{ij} - \frac{2}{3}k\delta_{ij} \tag{2.13}$$

where $k = \frac{1}{2}u^{'}u^{'}$ is the turbulent kinetic energy and S_{ij} the mean rate of strain tensor. With $\frac{2}{3}k\delta_{ij}$ being the isotropic stress, the anisotropic stress becomes:

$$a_{ij} = \overline{\tau_{ij}} - \frac{2}{3}k\delta_{ij} \tag{2.14}$$

Thus,

$$\overline{\tau_{ij}} = \underbrace{\frac{2}{3}k\delta_{ij}}_{\text{Isotropic}} + \underbrace{a_{ij}}_{\text{Anistropic}} \tag{2.15}$$

The anisotropic term is only responsible for the momentum transport. Linear Eddy Viscosity Models (LEVMs) assume that a_{ij} can be modeled as a linear function of the mean rate of strain tensor, such as:

$$\widetilde{a_{ij}} = -2\nu_t S_{ij} \tag{2.16}$$

The former approximation strongly eases the RANS approach, however it is the prime cause of poor accuracy in flows with strong anistropies. A more general effective-viscosity hypothesis was proposed by Pope in [63]. $\widetilde{\boldsymbol{a}}$ can be seen as a polynomial of ten independent tensor bases:

$$\widetilde{\boldsymbol{a}} = \sum_{n=1}^{N} G^{(n)}\boldsymbol{T}^{(n)}, \quad N = 10 \tag{2.17}$$

By defining the *mean rate-of-rotation* tensor $\boldsymbol{\Omega}$ as:

$$\Omega_{ij} = \frac{1}{2}\left(\frac{\partial \overline{u}_i}{\partial x_j} - \frac{\partial \overline{u}_j}{\partial x_i}\right), \tag{2.18}$$

we can easily write the ten bases as:

$$\begin{aligned}
\boldsymbol{T}^{(1)} &= \boldsymbol{S} & \boldsymbol{T}^{(6)} &= \boldsymbol{\Omega^2 S} + \boldsymbol{S\Omega^2} - \frac{2}{3}\left(\boldsymbol{S\Omega}\right)\boldsymbol{I} \\
\boldsymbol{T}^{(2)} &= \boldsymbol{S\Omega} - \boldsymbol{\Omega S} & \boldsymbol{T}^{(7)} &= \boldsymbol{\Omega S \Omega^2} - \boldsymbol{\Omega^2 S \Omega} \\
\boldsymbol{T}^{(3)} &= \boldsymbol{S^2} - \frac{\mathbf{1}}{\mathbf{3}}\left(\boldsymbol{S^2}\right)\boldsymbol{I} & \boldsymbol{T}^{(8)} &= \boldsymbol{S\Omega S^2} - \boldsymbol{S^2 \Omega S} \\
\boldsymbol{T}^{(4)} &= \boldsymbol{\Omega^2} - \frac{\mathbf{1}}{\mathbf{3}}\left(\boldsymbol{\Omega^2}\right)\boldsymbol{I} & \boldsymbol{T}^{(9)} &= \boldsymbol{\Omega^2 S^2} + \boldsymbol{S^2 \Omega^2} - \frac{2}{3}\left(\boldsymbol{S^2 \Omega^2}\right)\boldsymbol{I} \\
\boldsymbol{T}^{(5)} &= \boldsymbol{\Omega S^2} - \boldsymbol{S^2 \Omega} & \boldsymbol{T}^{(10)} &= \boldsymbol{\Omega S^2 \Omega^2} - \boldsymbol{\Omega^2 S^2 \Omega}
\end{aligned} \tag{2.19}$$

Machine learning-assisted invariance anistropic modeling (IAM) derives a correction in a RANS approach for the previous bases.

Given $\mathit{\Pi}_{i,hf}$ as one of the bases from high fidelity data, i.e. again DNS/LES or experiments, the goal of the machine-learning algorithm is to minimize the discrepancy:

$$\text{error} = \frac{\sum \|\mathit{\Pi}_{i,hf} - \mathit{\Pi}_{i,ML}|}{\sum \mathit{\Pi}_{i,hf}} \tag{2.20}$$

where $\mathit{\Pi}_{i,ML}$ is an invariant predicted by machine-learning and the summation is over all the training points. Please note that expression 2.20 vary from author to author and ad-hoc adjustments can be found in literature. Theoretically, this approach could be viable without recurring to the integrity bases, i.e. a machine-learning algorithm predicts each component of the Reynolds stress tensor directly. The reason behind the usage of the terms in (2.19) is the CPU time required for an algorithm training. Integrity bases, in fact, automatically grant the rotational invariance of the trained function. Using raw components has been proved to be wasteful, as pointed out by Ling et al. in [64], where rotational invariance was reached after 10 rotations in the training data in each Euler angle. Additional strategies to obtain Galielian and reflection invariance will be discussed further in the dissertation.
To summarize IAM:

- Write asymmetric part of the Reynolds stress tensor as a function of invariance bases;
- Invariance bases grant rotational invariance;
- Build a discrepancy function between high-fidelity data and RANS data;
- Train a ML algorithm to predict bases.

2.5.4 Relevant Prior Work

Literature strictly reflects the status at the start of the chapter, with three dominating schools of thought. Very few efforts have been put into achieving a machine-learning-based calibration, despite in [65] Yarlanki et al. improving the accuracy of the $k - \varepsilon$ model

by 25%. This correction is still highly local and not guaranteed to work when the training space is expanded. Duraisamy et al. successfully applied FIML to the Spalart-Allmaras turbulence model to enhance model accuracy, by correcting the production term in the $\widetilde{\nu}$ equation [2]. It can be extended to other turbulence models, as done in [66] for the $k-\omega$ model. The field of application differs from work to work, from transition to turbulence on a flat plate [67] to turbulent channel flow [68]. To bypass transition Zhang and Duraisamy investigate the similarity between Gaussian-process regression and Multi-Layer Perceptron Neural Networks [68]. The latter are usually preferred thanks to their high non-linearity in respect to other ML algorithms, as in [69]. A similar approach is followed by Singh et al. to better predict the aerodynamic efficiency on a family of NACA profiles [70]. Wang et al. in [71] are able to directly predict the velocity and turbulent field past a porous disk by means of Ensembled Karman applied to an experimental database. Anistropic modeling is also very common, due to the relative simplicity of formulation. Ling et al. in [72] shows that the use of invariance bases greatly reduces the computational costs of training a ML algorithm. Direct prediction of $\widetilde{\boldsymbol{a}}$ is performed through means of Random Forest or MLPNN algorithms. The methodology is proved to be effective both in simple problems (periodic hills / channel flows in [73, 64] and in heat transfer [74]. Similar results are obtained in [75], achieving a better prediction of Prandtl number in a crossflow.
Sandberg et al. follow an intermediate approach between field inversion and anisotropic modeling. In this case, machine-learning predicts the various components of the Reynolds stress tensor. An evolutionary algorithm is applied to derive the best function that minimizes the error between high fidelity data and a RANS model. The error is written in terms of macroscopic features (e.g. cascade deflection, Prandtl number), similarly to that seen in Duraisamy et al. An algebraic expression for $\widetilde{\boldsymbol{a}}$ as a function of integral basis is eventually derived. It is proven to be effective in a cascade flows enhancing wake mixing [76, 77], in film cooling [78] and crossflows [79]. In [80] Wu et al. applied the sample formulation to

recirculating flows, coming to similar conclusions. An interesting variation on the theme is proposed by Wang et al., that in [81] decompose $\widetilde{\boldsymbol{a}}$ into its magnitude and orientation, the aims of the ML algorithm. This framework, applied to 2D geometries (periodic hills and a square duct), leads to promising results.
Few efforts can be found regarding near wall modeling. Tieghi et al. in [82] try to enhance near wall modeling of the $k-\varepsilon$ model in run-time RANS simulations through a correction in the turbulent kinetic energy. Milano et al. in [83] train an artificial neural network to perform a non-linear proper orthogonal decomposition of the flow fields near the walls to perform intense dimensionality reduction for ML applications.

Table 2.2: Machine Learning and Turbulence

Ref. ID	**ML Alghoritm**	**Methodology**	**Turb. Model**
[70]	MLPNN	AM	Spalart-Allmaras
[67]	MLPNN	FI	Spalart-Allmaras
[66]	GP	FI	$k-\omega$
[68]	MLPNN	FI	$k-\omega$
[73]	MLPNN	IAM	$k-\varepsilon$
[72]	MLPNN/RNDMF	IAM	$k-\varepsilon$
[64]	MLPNN	IAM	$k-\varepsilon$
[75]	Decision Tree	IAM	$k-\varepsilon$
[74]	RNDMF	IAM	$k-\omega$
[79]	GEP	Hybrid FI/AM	Esplicit Algebraic SM
[78]	GEP	Hybrid FI/AM	URANS $k-\omega$
[69]	MLPNN	Hybrid FI/AM	Spalart-Allmaras
[77]	GEP	Hybrid FI/AM	Esplicit Algebraic SM
[76]	GEP	Hybrid FI/AM	URANS $k-v^2-\omega$
[80]	RNDMF	IAM	Spalart-Allmaras
[81]	RNDMF	IAM	$k-\varepsilon$
[71]	Ensembled Kalman	IAM	$k-\varepsilon$
[82]	MLPNN	Derivative	$k-\varepsilon$
[83]	MLPNN	Derivative	General
[84]	Boosting Tree	Derivative	General

Chapter 3

Machine-learning Tools and Techniques

"I'm sorry, Dave. I'm afraid I can't do that."

HAL 9000

This chapter will provide the reader a mathematical description of neural networks and popular machine-learning techniques that have been exploited further in the work.

3.1 What is Machine-Learning?

In standard programming an algorithm is designed, coded in a machine-intelligible language and eventually executed by one or more processing units. For engineering purpose, the performance

of an algorithm can be easily evaluated by the quality of its response. As soon as the program is designed, its performances are unequivocally defined, and can be improved only by changing input data. Regardless of the complexity of the algorithm, computers cannot go beyond the boundaries imposed by their programmers. An algorithm will always produce the expected output. In traditional programming you hard code the behavior of the program. In machine-learning, you leave a lot of that to the machine that learns from data.
Machine learning comes as an answer to problems, e.g. advice for products marketing, advanced medical diagnostics or self-driving cars, where a difficult task coexists with the need of a constant self-improvement of the the algorithm. In face recognition, for example, it is important to have a software that is able to correctly identify each individual in different poses and lighting conditions, while becoming better and better with the use of algorithm. Thus, we can summarize machine-learning in a synthetic but still effective way as *”[...] the science of getting computers to act without being explicitly programmed”* [1]. The discipline, intended in a modern fashion, began to flourish during the early ’90s. Today, hundreds of algorithms have been developed and optimized, with a fervent research still ongoing.
Following an original fashion (see *The Master Algorithm* from P. Domingos, 2015 [85]), whoever is a familiar with ML algorithms practice can identify himself in a different *tribes*. In [85] Domingos is searching for the *”Master Algorithm”*, that can be playfully defined by paraphrasing a famous quote as *” the algorithm to rule them all”*. His search ends with an open debate, identifying in the Markov logic networks (MLNs) the supreme learner on one side, while stating at the same time that *”I wouldn’t be so rash as to call this learner the Master Algorithm”*.

[1]Dr. Danko Nikolic, CSC and Max-Planck Institute

3.2 Machine-Learning Tribes

Whoever designs an algorithm knows that is non-trivial to state which algorithm will perform better in the assigned task. It is a common standard design practice, in fact, to compare different predictors and techniques to assess the best. Therefore, from a theoretical perspective, boundaries between tribes are sharp and clear. In a real scenario a AI expert will take into account them all.

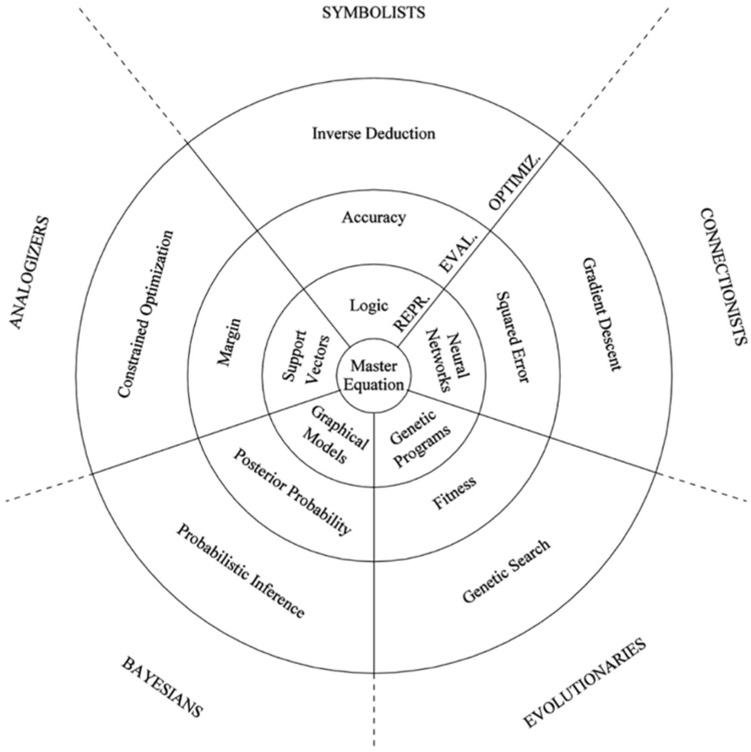


Figure 3.1: A visualization of the different ML tribes

Regardless of the tribes, each methodology is based on three different layers:

L1 **Representation:** flow of data is transposed in a set of classifiers or regressors, e.g. the language that a computer understands;

L2 **Evaluation:** the performance of the algorithm is evaluated through means of objective / scoring function;

L3 **Optimization:** the algorithm is self-optimized to get better scores during the evaluation phase.

Each tribe has a set of core beliefs and addresses its effort to specific categories of problems. Every clan bets on their *master algorithm*.

- For **symbolists**, understanding cannot be directly derived from the preexisting knowledge from scratch. They are good at manipulating symbols, as mathematicians solves equations by replacing expressions by other expressions. They don't start with an initial guess of the conclusions, but rather work backward to fill in the gaps between premises and conclusions. Not by chance, their chosen master algorithm is inverse deduction.

- **Connectionists** started by observing the learning process of the human brain, thinking that their algorithms should behave in a similar manner. An artificial brain is then build, miming the basic structure of a biological neural network. The key of the success is the goodness of the connections, that are morphed and weighted during a learning process. The master algorithm is backpropagation, which compares outputs from the connections and the reality and gradually adjust the structure in a constant improvement to the optimum.

- **Evolutionaries** have learned from the natural selection process. The key idea is that a population of initial solutions that withstand an evolution process eventually generates a family of best solutions. Virtual era after virtual era, individuals are mated together and their heirs are selected by their capacity of fitting the environment. The evolutionaries' master algorithm is genetic programming, which algorithmically replicates most of the features of natural selection.

- **Bayesians** think that uncertainty is the source of all knowledge. Learning the uncertainty is knowledge itself. They know better than other how to deal with noisy, incomplete, and even contradictory information. The solution is probabilistic inference, and the master algorithm is Bayes theorem.

- For **analogizers**, no knowledge stands alone, thus it is always possible to derive information by analyzing similarities. An individual is what it resembles. Similar phenomena behave in similar manner. The key lies in the ability of discerning between different behaviors. The analogizers master algorithm is the support vector machine, which draws boundaries and eventually infers.

In this dissertation we will focus on the connectionists. The choice is justified by several factors:

- *Non-linearity*: NNs are naturally non-linear, allowing the algorithm to solve complex and simple problems as well;

- *Generalization capability*: within the non-linear algorithms, NNs present the best performances when inferring out of the bounds of training data;

- *Overfitting resilience*: advanced techniques avoid the tendency of ML algorithms to overfit training data;

- *Literature documentation*: different authors highlight the effectiveness of NNs, additionaly providing useful information on the architecture of the algorithms (see Chapter 2).

We must also point out that as long as the problem is well conditioned and training data in statistically significant, non-linear algorithms show similar performances. As an example, it is the case of random forest regressors, that can achieve the same accuracy as NNs. Therefore, NNs were mainly selected based on their generalization capability.

3.3 The Connectionists Paradigm: Neural Networks

The idea of reproducing a self-organization organism was born into biology and neuroscience, where we find the first attempts to replicate a human brain. One of the initial steps towards an artificial intelligence can be attributed to the neurophysiologist Warren Sturgis McCulloch and to the mathematician Walter Pitts. In 1943, in fact, they wrote a cutting-edge work describing the basic functioning of neurons, with the final aim to replicate that behavior in a machine. Neurons were intended as binary, e.g. they can be either activated or quiet. The state of activation was only triggered if the weighted sum of the inputs (also intended as binary), exceeded some limit value. In the initial view of neurons, weights were fixed. By so doing, it was expected that a simple circuit could replicate the behavior of a human neuron. Sturgin and McCulloch thought that they could replicate an entire brain by building a large enough circuit.
The concept of fixed weights was overcome only in 1949, when the psychologist Donal Olding Hebb published the book *The Organization of Behavior*. Hebb theorized that weights were updated at every usage, thus the learning process was a property of the connection between neurons. Hebb also defined the famous *Hebb rule,* that constitute nowadays the fundamental algorithm for AI learning. The Hebb rule can be summarized as it follows: if two neurons are excited at the same time, the connection between them increases, otherwise it diminishes.
During the early ’50s we can find several attempts to create a ”thinking machine”, as done by Nathaniel Rochester, creator of the first IBM computer. In 1956, Claude Shannon, Bell Laboratories, and

John McCarthy, Dortmund College, instituted the *"Dartmouth Research Project on Artificial Intelligence"*, where we can find one of the first involvements of industries on the topic, treated before as a purely academic subject.
In 1958 von Neumann published *The Computer and the Brain*, where through a reductionist approach he analyzes similarities between analogical calculators and human brain. A brain is here seen as a sum of basic functions. Due to the inability of performing complex tasks, neurons can only respond to some inputs with the boolean operations: AND, OR and NOT. However, a neuron is not able to be activated repetitively, thus two responses have a delay of 10^{-4} and 10^{-2} seconds. In calculators this interval can be reduced to a minimum of 10^{-7}. However, von Neumann states, the human brain is still superior to a machine thanks to its bigger dimensions. He also concludes with the assumption that the human brain naturally includes some sort of non-linearity, althought its source has not been identified yet.
To find the introduction of the concept of *perceptron* we need to wait until 1957, when Frank Rosenblatt, from Cornell Aeronautical Research Laboratory, develops the first thinking machine, the Perceptron. The computer is able to learn and recognize pictures, and it is the oldest neural network still working. The perceptron computes a weighted sum of the inputs, subtracts a threshold and returns one among two possible values. In Rosenblatt's neurons, weights no longer have discrete values (-1/0/1), but they vary continuously. A set of weights is derived to minimize the error between predictions and observations, through a supervised learning process. The work from Rosenblatt was seriously criticized by two mathematicians, Marvin Minsky and Seymour Papert. In 1969 they publish *"Perceptrons. An introduction to Computational Geometry"*, where they demonstrate that the former structure of neurons was not able to replicate the basic XOR function. Minsky and Papert also states that even more complex structures would not able to simulate such function. The historical impact of this work is unimaginable, as it was taken as an example of the uselessness of neural networks. An halt to researches on neural networks is consequently found all along

the Seventies. The same Minsky and Papert in 1988 would negate the negative impact of their former work, and, in the third edition of *Perceptrons*, they show that the learning capability of perceptrons were dramatically enhanced. As a side note, the lack of adequate mathematical theories strongly affected ongoing researches.
Development of neural networks has been flourishing since 1986, when David Everett Rumelhart, Geoffrey Hinton and Ronald Williams introduces the concept of *backpropagation*. Through backpropagations, weights are no longer fixed but they change dynamically, granting better performances while performing complex tasks. The golden year for neural networks is 1988, when connectionism becomes a pardigm among the statistical sciences. By the end of 1988, the International Neural Network Society counts more that 2000 members that publish in its own journal.
Today, the pros of neural networks and machine-learning are known worldwide. A fast search on Google Trends [2] shows a steep rise in interest on the neural-networks and related topics in the past decade. In spite of the incredible rate at which new algorithms are invented, we are in store for a golden age of neural networks.

3.3.1 Learning as Optimization

The term *neural networks* does not uniquely define a structure but a large class of models and learning techniques. To ease the discussion we will focus on the so called *vanilla* neural network, i.e. single/multi-hidden layer back-propagation algorithms.
The goal of the *learning process* is to train an algorithm, or *predictor*, to perform complex tasks. We distinguish between three different types of learning processes:

- ***Supervised learning:*** the predictor is asked to learn from a set of given examples. The examples contains both the input values and the known output. To our extents, we will always work with supervised learning.

[2]https://trends.google.com

- ***Reinforced learning:*** a score that is a measure of the performance of the algorithm is used in place of the known outcome.It is similar to supervised learning but less popular. It is proved to be especially effective in control systems.

- ***Unsupervised learning*** tries to discover patterns and similarities in input data without knowing anything about the expected output.

Neural networks learn similarly to the human brain. It is therefore useful to provide basic concepts on the topic. Merits of the first researches on that subject go to Edward Lee Thorndike, with his work "Trial and Error Learning" [86]. This heuristic method can be summarized by the idea that, while performing a task, by increasing the number of trials we decrease the number of fails. It was exported as a problem solving technique in computer science, also called *"generate and test"*. Thorndike's theory was developed observing animal behaviors, however human beings learn in a similar manner through the *operant conditioning*, see [87]. The human brain relates a specific behavior to a reaction from the surrounding environment. In so doing, it becomes better at avoiding actions with negative consequences, through a system of reinforcements and punishments. In the experiments of Sinner, for example, when a rat pressed a level it was rewarded with food, leading to an increase in frequency of such action.
We have already stated that neural networks behave in a similar manner. However, how is it possible to mathematically define a negative consequence from a good one? In which way can we replicate the complex and mostly unknown animal connectome through a computer? How do we represent human experience?
In the following sections we will distinguish between the *architecture* of the neural network, that conceives the physical brain, and the *optimization process*, that mimics the learning process of human beings.

3.3.2 Architecture

We will now virtually ”dissect” the artificial brain in its most basic anatomic parts. Any network can be seen as a sum of basic units, called *artificial* neurons. The notation in use is:

- Scalars - small *italic* letters: *a,b,c*;
- Vectors - small **bold** nonitalic letters: **a,b,c**;
- Matrices - capital **bold** nonitalic letters: **A,B,C**;
- $\widehat{\ }$ - data from a training database, taken as *ground truth*.

3.3.2.1 Single-layer Neural Networks

On an abstract level, a neural network can be thought of as a function f that maps inputs x $\in$ R^m, also called *features,* to an output y $\in$ R^n, with m and n being the dimensions of the spaces. Inputs and outputs can assume either categorical or continual values. Thus:

$$\mathrm{f}_\theta : x \rightarrow y \tag{3.1}$$

where θ are the parameters of the model. Those can be divided into *hyperparameters*, e.g. parameters that are chosen only once and remain the same during the learning process, and *weigths* and *biases,* optimized during the training process. Fig. 3.2 shows the basic unit that builds up a neural network.

Any unit accepts as input a value x$\in$ R^3. Its neuron can be parameterized by two values a weight w and a bias b. The product of weight and input is summed to the bias to obtain the *transfer potential Σ*. The transfer potential is used as an input for the *activation function Φ*. At this point, the neuron has been *activated* and it *fires*. The output y is also called *action potential*. This process can be summarized by the expression:

$$y = \Phi\left(w \cdot x + b\right) \tag{3.2}$$

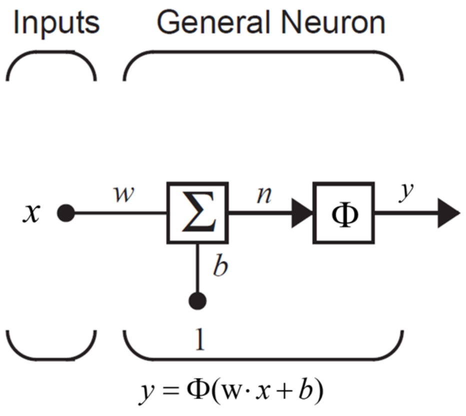


Figure 3.2: A single-input neural unit

3.3.2.2 Multi-layer Neural Networks

A multi-input neuron is just a generalization of the single input neuron(Fig. 3.3). Each of the individual inputs $x_1, x_2, ..., x_N$ is weighted by the $w_1, w_2, ..., w_N$ coefficients.

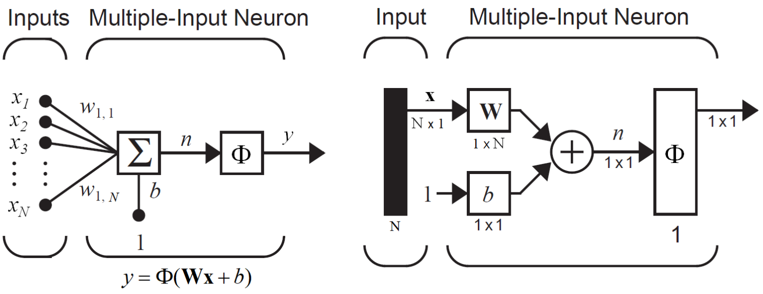


Figure 3.3: A multi-input neural unit and its dimensions

In this case, the transfer potential is given by:

$$n = \mathrm{w}_1 \cdot \mathrm{x}_1 + \mathrm{w}_2 \cdot \mathrm{x}_2 + ... + \mathrm{w}_n \cdot \mathrm{x}_n + \mathrm{b} = \sum_{i=1}^{N} \mathrm{w}_i \mathrm{x}_i + \mathrm{b} = \mathbf{W} \cdot \mathbf{x} + \mathrm{b} \quad (3.3)$$

where $\mathbf{W}$ and $\mathbf{x}$ are the weights and input matrix respectively. The action potential can be therefore written as:

$$y = \Phi\left(\sum_{i=1}^{N} w_i \cdot x_i + b\right) \quad (3.4)$$

In multi-inputs neural networks defining the dimension of each of the elements is trivial (Fig. 3.3b). In fact, all the former expressions consider each of the input as a scalar. It means that the input matrix $\mathbf{x}$ has $N \times 1$ dimensions, where N is the number of inputs. The expected outcome of the neural unit is a scalar, therefore the only allowed dimensions for the weight matrix is $1 \times N$.
In a more complex scenario, the number of neurons S could differ from the number of inputs N. Fig. 3.4 shows an example of a multi-input single-layer network, with $S \neq N$. N inputs from the first layer are fired and used in turn as inputs for the S-sized layer. By so doing, the new shape of the weight matrix becomes S $\times$ N.

$$\mathbf{W} = \begin{bmatrix} \mathrm{w}_{1,1} & \mathrm{w}_{1,2} & \cdots & \mathrm{w}_{1,N} \\ \mathrm{w}_{2,1} & \mathrm{w}_{2,2} & \cdots & \mathrm{w}_{2,N} \\ \vdots & \vdots & & \vdots \\ \mathrm{w}_{S,1} & \mathrm{w}_{S,2} & \cdots & \mathrm{w}_{S,N} \end{bmatrix}$$

The value of an *i,j* element of the weight matrix represents the strength of the connection between the ith input and the *jth* neuron. The higher the values, the stronger the connection. When the number of neurons differs from the number of inputs, instead of a single bias we find a vector **b**. The transfer potential is eventually transformed in a vector of outputs **y**. Nothing forbids the use of a different transfer function for each neuron, even though it is usually the same to ease implementation, especially in bigger networks.

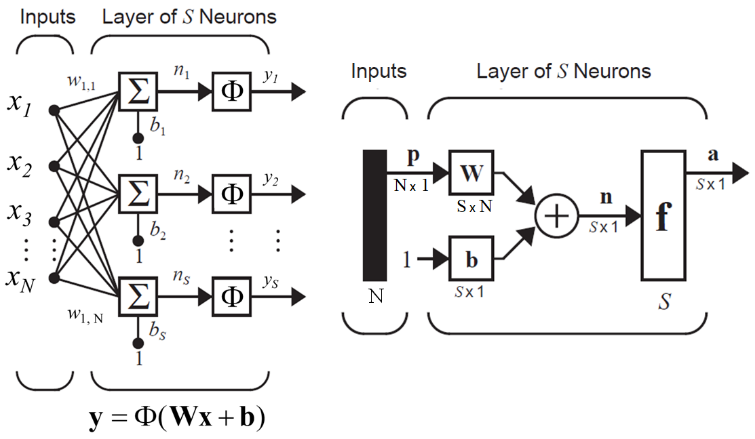


Figure 3.4: Single layer neural network and its dimensions.

Single layer neural networks, however, show strong deficiencies when dealing with complex problems, and today their performances fall behind modern approaches like Random Forests or Support Vector Machines. It is therefore necessary to deepen the complexity of the structure seen before. It is achieved through multi-layering, e.g. the outputs of a layer are exploited as inputs for a subsequent layer. Each layer has its own number of neurons, set of weights, biases and activation functions. A superscript will consequently address which layer a matrix belongs to.

The last layer is also called the *output layer*, while the other are usually denoted as *hidden layers*. Input of an intermediate layer, namely the *net input*, will be denoted with n. The final output of the network can be therefore written as a combination of the previous layer:

$$\mathbf{y}^3 = \Phi^3 \left(\mathbf{W}^3\Phi^2 \left(\mathbf{W}^2\Phi^1 \left(\mathbf{W}^1\mathbf{x}^1 + \mathbf{b}^1\right) + \mathbf{b}^2\right) + \mathbf{b}^3\right) \tag{3.5}$$

We will now provide a further insight on how neural networks are built and trained.

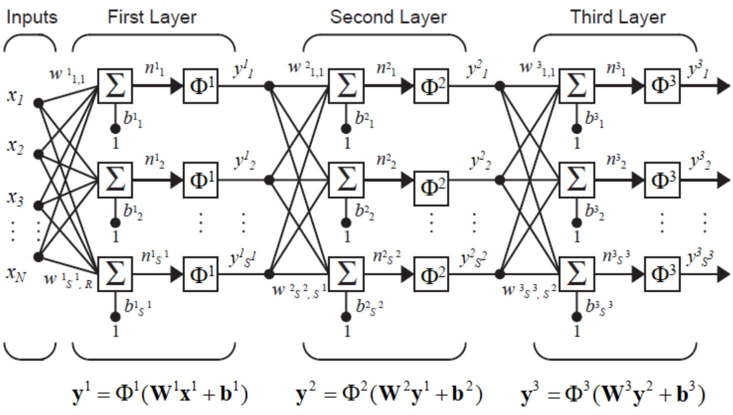


Figure 3.5: Example of three-layered perceptron.

3.3.2.3 Activation Functions

Φ is the source of non-linearity of neural networks, and its shape changes from application to application. Several activation functions have reached high popularity. The simplest ones are the linear function (Fig. 3.6a) , where the action potential is only controlled by the slope of the curve, and the Heaviside function (Fig. 3.6d), that represents a neuron that can be either activated by the full inputs or not.

The function that has achieved the highest popularity is the *logistic* function, also known as *sigmoid* (Fig. 3.6b). In modern practice preferences are shifting toward the hyperbolic tangent (Fig. 3.6c), due to higher capability dealing with outliers. Activation functions must satisfy several requirements:

- *Non-linearity:* it can be easily proved mathematically that multiple linear layers can collapse into a single layer, vanquishing the advantages of use of the multiple layers;

- *Limited response:* some functions, such as the linear or ReLu, provide an infinite response as the activation potential becomes high. In so doing they vanquish any training instance

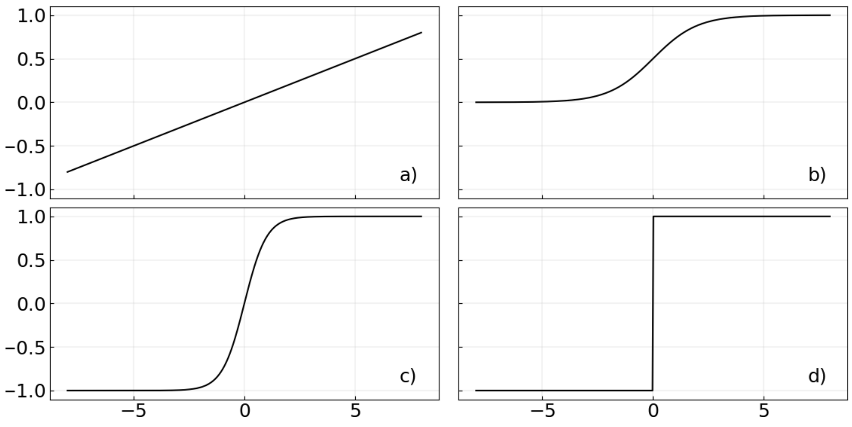


Figure 3.6: Common activation functions - a) linear: αx with $\alpha > 0$, b) logistic: $1/(1 + e^{-x}$, c) hyperbolic tangent: $(e^x - e^{-x})/(e^x + e^{-x})$, d) Heaviside function: 0 for $x < 0 \mid 1$ for $x \geq 0$

with lower values. Outliers are particularly disruptive due to this aspect;

- *Continuously differentiable:* training in an algorithm is performed through minimizing the difference between predicted output and ground truth. The gradient of the cost leads the process. Steep gradients or ravines hinder the training process, therefore activation functions should be differentiable everywhere, a part from few discontinuity points;

- *Monotonic:* it can be proven mathematically that a monotonic function has a convex error surface, e.g. a global minimum is guaranteed;

- *Smooth functions with a monotonic derivative:* this attribute is not mandatory. During the training phase, backpropagation handle the influence of each neuron in the next layer. If we use a non-monotonic activation function we introduce a chaotic behavior in the network, as when increasing the weight of a neuron it may have less influence, that is opposite of what was intended.

- *Approximates identity near the origin:* it helps weight-independent initialization. In fact, it is accepted practice to initialize weights and biases will values that are close to zero. In so doing, computed gradients of cost will be near-zero, that speed up the convergence process. Otherwise, an ad-hoc weight initialization should be performed and it seriously impair the robustness of NN design.

3.3.2.4 Error Functions

The aim of the *error functions*, also called *cost* or *loss functions*, is to reduce the complexity of training a model down to a single number. In so doing, various instances of training can be immediately compared and evaluated. Each model requires an ad-hoc selection of the loss function, and a wrong choice may lead to a unsuccessful training. It is therefore necessary that the chosen form represents the design goals. J provides a measurement of how good the model in approximating the relationship between inputs $\mathbf{x}$ and outputs. Two forms have gained popularity in machine-learning practice, thanks to their relatively simple mathematical formulation and effectiveness: *mean squared error* (MSE) and *cross entropy* (CE).
Mean squared error computes the averaged error over all the training examples. The error is measured as a simple squared difference between the expected output of the model $\hat{y}$ and the actual output of the net y:

$$\text{MSE} = \frac{1}{2N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2 \tag{3.6}$$

Squared error relates to *Gaussian noise*. In most of the scenarios, regardless of any set of weight or biases, the difference $\hat{y} - y$ will never be zero. Training data, in fact, will almost surely be affected by noise due to various independent sources, e.g. measurement errors, environmental factors. When developing an algorithm, we are trying to model relationships underlying data and not the noise

itself. In reality, our goal is to find the best set of weight and biases that takes the noise distribution into account. Following the *central limit theorem*, noise will be normally distributed, i.e. it can be represented through a Gaussian distribution. A Gaussian curve can be described by two parameters, mean and variance. If we assume that there is no systematic error in data, i.e. the distribution has a zero mean, computing MSE is equivalent to computing the variance [88]. The best predictor is the one with the narrowest distribution around the predicted values (smallest variance). This explains why minimizing MSE usually turns out to be effective in most of the regression problems, regardless of the data to model.
The learning process in classification problems relies on *maximum likelihood estimation* (MLE). MLE searches the optimum set of weights and biases by maximizing a likelihood function derived from the training data [89]. Minimizing the difference between the empirical distribution and the likelihood function is the real goal of the training phase. It is the same as minimizing the cross-entropy between the two distributions. In a classification problem we are interested in labeling a combination of input variables to a class. We therefore model the probability of a training example belonging to each class:

$$\mathrm{CE} = -\sum_{N} p_{\hat{y}}(x) \log q_{y}(x), \tag{3.7}$$

where p and q can be read as the probability distributions from the training data and from the modeled data respectively.

3.3.3 Optimization

Every component of the architecture of a neural network have been fully explained. Effectiveness of a neural network lies in the ability of organizing and optimizing weight and biases. That is done through solid optimization techniques.

3.3.3.1 Gradient Descent

Cost functions, apart from a few lucky exceptions, do not present analytic forms. Points of minimum/maximum must be found through

numerical iterative methods. To this extent, the most popular optimization methodology is the *gradient descent* (GD). The key idea of GD is that in a curvy function, if we keep moving downhill, we will eventually come to a local/global minimum. The size of the step we take is also called *learning rate* and it is a scalar value α.
Every iteration GD computes the derivative of the cost function with respect to weights and biases.

$$J'(w,b) = \begin{bmatrix} \frac{\partial J}{\partial w} \\ \\ \frac{\partial J}{\partial b} \end{bmatrix} \tag{3.8}$$

The derivative of the cost function tells us the slope of the cost function, e.g. in which direction to perform the update step. GD change values of $\mathbf{W}$ and $\mathbf{b}$ until, for each of the N^{th} feature:

$$\mathrm{x}_n = \mathrm{x}_n - \alpha \frac{\partial J}{\partial \mathrm{x}_n} J(\mathrm{x}_n) \tag{3.9}$$

In a mono-dimensional feature space it is easy to plot GD iterations toward convergence (Fig. 3.7). Unfortunately, we cannot simply optimize multi-layer neural network by running instances of a GD layer per layer. For a single-layer linear network the error is an explicit linear function of the network weights, and its derivatives with respect to the weights can be easily computed. On the opposite, we experience a signficant increase in the computational costs of the derivative in Eq.3.8 with the number of hidden layers and neurons. Consequently, convergence toward a point of optimum is dramatically slow and it can become impossible for networks with complex architectures or data with high-dimensionality. To overcome these difficulties, *back-propagation* technique is universally accepted and exploited in the common machine-learning practice.

3.3.3.2 Back-propagation

Referring to previous lexicon, the equation that describes data flow in an intermediate layer for a network with M layers is:

$$\mathbf{y}^{j+1} = \Phi^{j+1}\left(\mathbf{W}^{j+1}\mathbf{y}^{j} + \mathbf{b}^{j+1}\right) \qquad \text{for x} = 2,\ldots,M-1 \tag{3.10}$$

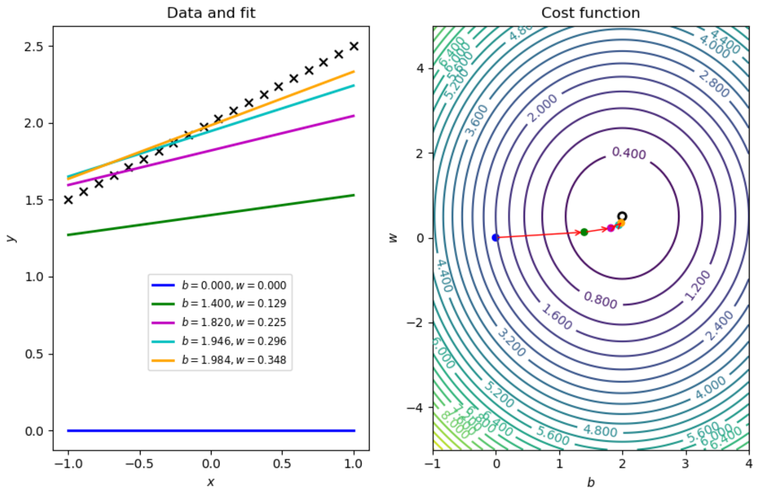


Figure 3.7: Iterations of GD algorithm when fitting data with a linear model $y = w \cdot x + b$

The expression in Eq.3.10 remains valid even for the first and last layers ($j = 0, M - 1$), with slight modifications. The first layer ($j = 0$) has external data $\mathbf{x}$ as input, while we denote outputs of the last layer ($j = M$) as y'. During a training phase, the neural network maps the k^{th} input to the $y_k^{'}$ output. $\widehat{y}_k$ is the true k^{th} target value. The goal of the algorithm is to minimize the error e, expressed through the cost function:

$$J(\mathbf{x}, \mathbf{b}) = \text{MSE}\left[e^2\right] = \text{MSE}\left[(\widehat{y} - y)^2\right] \tag{3.11}$$

It is possible to generalize Eq.3.11 for multiple outputs:

$$J(\mathbf{x}, \mathbf{b}) = \text{MSE}\left[\mathbf{e}^T\mathbf{e}\right] = \text{MSE}\left[(\widehat{\boldsymbol{y}} - \boldsymbol{y})^2\right] \tag{3.12}$$

MSE can also be written in an approximated form as:

$$J(\mathbf{x}, \mathbf{b}) = (\widehat{\boldsymbol{y}}(k) - \boldsymbol{y}(k))^2\,(\widehat{\boldsymbol{y}}(k) - \boldsymbol{y}(k)) = \boldsymbol{e}^{\boldsymbol{T}}(k)\boldsymbol{e}(k) \tag{3.13}$$

where the expectation of the squared error is replaced by the squared error at iteration k. Each iteration of the GD algorithm in the m^{th} computes:

$$w_{i,j}^{m}(k+1) = w_{i,j}^{m}(k) - \alpha \frac{\partial J}{\partial w_{i,j}^{m}} \tag{3.14}$$

$$b_{i,j}^{m}(k+1) = b_{i,j}^{m}(k) - \alpha \frac{\partial J}{\partial b_{i,j}^{m}} \tag{3.15}$$

where α is the learning rate. For a single-layered neural network, the previous expressions are iterated towards convergence with small computational efforts. In multi-layer architecture the error is not an explicit functions, therefore these derivatives are hard to compute. We can use the chain rule of calculus to compute derivatives. Taking a function *f* that is an explicit function only of the variable v_1, we can compute the derivative with respect to a different variable v_2 by:

$$\frac{df(v_1(v_2))}{dv_2} = \frac{df(v_1)}{dv_1} \times \frac{d(v_1(x_2))}{dv_2} \tag{3.16}$$

In a similar way, derivatives of Eq. 3.14 and 3.15 with respect to the net input n can be computed by:

$$\frac{\partial J}{\partial w_{i,j}^{m}} = \frac{\partial J}{\partial n_{i}^{m}} \times \frac{\partial n_{i}^{m}}{\partial w_{i,j}^{m}} \tag{3.17}$$

$$\frac{\partial J}{\partial b_{i,j}^{m}} = \frac{\partial J}{\partial n_{i}^{m}} \times \frac{\partial n_{i}^{m}}{\partial b_{i,j}^{m}} \tag{3.18}$$

Those terms can be computed in a easy way, since we already know that the net input n to a layer m is a direct function of weights and biases of that layer.

$$n_{i}^{m} = \sum_{j=1}^{M-1} w_{i,j}^{m} n_{j}^{m-1} + b_{i}^{m} \tag{3.19}$$

And consequently:

$$\frac{\partial n_i^m}{\partial w_{i,j}^m} = y_j^{m-1} \tag{3.20}$$

$$\frac{\partial n_i^m}{\partial b_i^m} = 1 \tag{3.21}$$

We can define *sensitivity* of J as the change in the i^{th} element of the input at layer m:

$$s_i^m = \frac{\partial J}{\partial n_i^m} \tag{3.22}$$

Therefore:

$$\frac{\partial J}{\partial w_{i,j}^m} = s_i^m y_j^{m-1} \tag{3.23}$$

$$\frac{\partial J}{\partial b_i^m} = s_i^m \tag{3.24}$$

In so doing, we can rewrite gradient descent in a simpler form:

$$w_{i,j}^m(k+1) = w_{i,j}^m(k) - \alpha s_i^m n_j^{m-1} \tag{3.25}$$

$$b_i^m(k+1) = b_i^m(k) - \alpha s_i^m \tag{3.26}$$

that in a matrix form becomes:

$$\boldsymbol{W}^m(k+1) = \boldsymbol{W}^m(k) - \alpha \boldsymbol{s}^m (\boldsymbol{n}^{m-1})^T \tag{3.27}$$

$$\boldsymbol{b}^m(k+1) = \boldsymbol{b}^m(k) - \alpha \boldsymbol{s}^m \tag{3.28}$$

We can write Eq. 3.22 in matrix form as well:

$$\mathbf{s}^m = \frac{\partial J}{\partial \mathbf{n}^m} = \begin{bmatrix} \frac{\partial J}{\partial n_1^m} \\ \frac{\partial J}{\partial n_2^m} \\ \vdots \\ \frac{\partial J}{\partial n_M^m} \end{bmatrix} \tag{3.29}$$

At this point, we can finally compute all the sensitivities s^m. The process is called *backpropagation,* as it describes a recurrence relationship in which the sensitivity at layer m is computed from the sensitivity at layer $m+1$. We can write the Jacobian matrix as:

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} = \begin{bmatrix} \frac{\partial n_1^{m+1}}{\partial n_1^m} & \frac{\partial n_1^{m+1}}{\partial n_2^m} & \dots & \frac{\partial n_1^{m+1}}{\partial n_1^M} \\ \frac{\partial n_2^{m+1}}{\partial n_1^m} & \frac{\partial n_2^{m+1}}{\partial n_2^m} & \dots & \frac{\partial n_2^{m+1}}{\partial n_1^M} \\ \vdots & \vdots & & \vdots \\ \frac{\partial n_M^{m+1}}{\partial nn_1^m} & \frac{\partial n_M^{m+1}}{\partial n_2^m} & \dots & \frac{\partial n_M^{m+1}}{\partial n_1^M} \end{bmatrix} \tag{3.30}$$

where each *i,j* element of the Jacobian matrix:

$$\frac{\partial n_i^{m+1}}{\partial n_j^m} = \frac{\partial \left(\sum_{l=1}^{M} w_{i,l}^{m+1} y_l + b_i^{m+1}\right)}{\partial n_j^m} = w_{i,j}^{m+1} \frac{\partial y_j^m}{\partial n_j^m} \tag{3.31}$$

$$= w_{i,j}^{m+1} \frac{\partial J^m(n_j^m)}{\partial n_j^m} = w_{i,j}^{m+1} J'^m(n_j^m) \tag{3.32}$$

J' reads as the first derivative of the Jacobian matrix with respect to the j*th* net input. Eq. 3.30 can be eventually written as:

$$\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m} = \mathbf{W}^{m+1}\mathbf{J}'(\mathbf{n^m}) \tag{3.33}$$

On the right side of Eq. 3.33, $J'(n^m)$ is the diagonal matrix of the partial derivatives:

$$\mathbf{J}'(\mathbf{n^m}) = \begin{bmatrix} J'(n_1^m) & 0 & \dots & 0 \\ 0 & J'(n_2^m) & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & J'(n_M^m) \end{bmatrix} \tag{3.34}$$

At this point, we can combine Eq. 3.33 with the chain rule of derivatives in matrix form, Eq. 3.18:

$$\mathbf{s}^m = \frac{\partial J}{\partial \mathbf{n}^m} = \left(\frac{\partial \mathbf{n}^{m+1}}{\partial \mathbf{n}^m}\right)^T \frac{\partial J}{\partial \mathbf{n}^{m+1}} = \mathbf{J}'(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \frac{\partial J}{\partial \mathbf{n}^{m+1}} \tag{3.35}$$

$$= \mathbf{J}'(\mathbf{n}^m)(\mathbf{W}^{m+1})^T \mathbf{s}^{m+1} \tag{3.36}$$

The name *backpropagation* comes from the next step. Starting from the last layer, in fact, sensitivities are propagated back layer per layer until the first one.

$$\mathbf{s^M} \rightarrow \mathbf{s^{M-1}} \ldots \rightarrow \mathbf{s^2} \rightarrow \mathbf{s^1} \tag{3.37}$$

To complete the description of backpropagation, we need to define the first step, i.e. how we compute the first derivative for the M*th* layer:

$$s_i^M = \frac{\partial \mathbf{J}}{\partial n_i^M} = \frac{\partial(\widehat{\mathbf{y}} - \mathbf{y})^T(\widehat{\mathbf{y}} - \mathbf{y})}{\partial n_i^M} \tag{3.38}$$

$$= \frac{\partial \sum_{j=1}^{s^M} (\widehat{y}_j - y_j)}{\partial n_i^M} = -2(\widehat{y}_j - y_j)\frac{\partial y_i}{\partial n_i^M} \tag{3.39}$$

We already know that

$$\frac{\partial y_i}{\partial n_i^M} = \frac{\partial y_i^M}{\partial n_i^M} = \frac{\partial J^M(n_i^M)}{\partial n_i^M} = J'^M(n_i^M) \tag{3.40}$$

therefore the sensitivity for the i*th* neuron in the last layer M becomes:

$$s_i^M = -2(\widehat{y}_i - y_i)J'^M(n_i^M) \tag{3.41}$$

expressed in a matrix form as:

$$\mathbf{s}^M = -2\mathbf{J}'^M(\mathbf{n}^M)(\widehat{\mathbf{y}} - \mathbf{y}) \tag{3.42}$$

Back-propagation is now complete. To summarize, to perform back-propagation:

1. Propagate the input $\mathbf{x}$ toward the second-last layer: $\mathbf{y}^{m+1} = \mathbf{J}^{m+1}(\mathbf{W}^{m+1}\mathbf{y}^{m} + \mathbf{b}^{m+1}) \quad \text{for m}=0, 1, \ldots, M-1$;
2. Compute the sensitivity of the last layer: $\mathbf{s}^{M} = -2\mathbf{J'}^{M}(\mathbf{n}^{M})(\widehat{\mathbf{y}} - \mathbf{y})$;
3. Back-propagate the sensitivities toward the first layer: $\mathbf{s}^{M} = \mathbf{J'}^{M}(\mathbf{n}^{M})(\mathbf{W}^{m+1})\mathbf{s}^{m+1} \quad \text{for m}=M-1, \ldots, 2, 1$;
4. Update weights and biases using the steepest descent rule: $\mathbf{W}^{m}(k+1) = \mathbf{W}^{m}(k) - \alpha s^{m}(\mathbf{y}^{m-1})^{T}$ and $\mathbf{b}^{m}(k+1) = \mathbf{b}^{m}(k) - \alpha s^{m}$.

All the process discussed thus far can be considered the core of the learning process.

3.3.3.3 Adaptive Momentum Estimation

Whoever decides to enhance turbulence modeling through means of machine-learning, soon or later encounters one of the most difficult problem to overcome, namely generalization. Real flows, in fact, come with the most various configuration and typologies. Good prediction of those flows are achieved through modeling Navier-Stokes equations, that represent a common underlying physical structure. To build an artificial intelligence, however, a training database is required. Training data is constituted of a limited number of entries, few billions at best, leading to a strong under-representation of physics. It means that a complete ML driven model is far to be achieved, due to the limited amount of computational power that is available. An accepted practice involves building algorithms that enhance the standard models only in a few configurations, while not deteriorating their quality on the others. Even so doing, getting the model to work with every possible frame of reference is not an easy task. Furthermore we must deal with the difficulty of managing sparse data, i.e. dataset entries do not smoothly populate the feature space.
ADAptive Momentum estimation optimizer (ADAM) constitute one of the best tools to deal with sparse data [90]. They key idea is very

similar to the classical stochastic gradient descent (GD). However, instead of using a fixed value, ADAM computes the optimal learning rate for each of the input features. This per-parameter optimization is based on the first and second statistical moments of the gradients, i.e. mean and variance. In fact, ADAM stores an exponentially decaying average of the past gradients m and their square m^2:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\,\nabla J_t \tag{3.43}$$

$$m_t^2 = \beta_2 m_{t-1}^2 + (1-\beta_2)\,\nabla J_t^2 \tag{3.44}$$

where t is the current iteration of optimization and $t-1$ the previous one. β_1 and β_2 constants are chosen a-priori, with common reported values of 0.9 and 0.999 respectively. ADAM update rule therefore becomes:

$$W_{t+1} = W_t - \frac{J'}{\sqrt{m_t^2 + \epsilon}} m_t \tag{3.45}$$

where ϵ is a small number to avoid division by zero. Apart from a few exceptions, ADAM performances are superior to any other optimizer, making it the benchmark for optimizers in machine-learning applications [91].

Chapter 4

Development of Machine-Learning Assisted Tools for Turbulence Modeling

This chapter will describe three different application of artificial intelligence to turbulence modeling:

- A data-driven wall function for stationary flows;

- A data-driven wall function for rotating ducts;

- A tool for the identification of poorly ventilated zones in GT turbines.

4.1 A Data-driven Wall Function for Stationary Flows

This part of the works was subject of paper GT-91238 [82]. Near-wall modelling is one of the most challenging aspects of CFD computations. In fact, integration-to-the-wall with low-Reynolds approach strongly affects accuracy of results, but strongly increases the computational resources required by the simulation. A compromise between the accuracy and speed to solution is usually obtained through the use of wall functions, especially in RANS computations, which normally require that the first cell of the grid to fall inside the log-layer ($50 < y+ < 200$) [29]. This approach can be generally considered as robust, however the derivation of wall functions for attached flow boundary layers can mislead to non-physical results in the presence of specific flow topologies, e.g. recirculation, or whenever a detailed boundary layer representation is required (e.g. aeroacoustics studies) [92]. In this work, a preliminary attempt to create an alternative data-driven wall function is performed, exploiting artificial neural networks (ANNs). Whenever enough training examples are provided, ANNs have proven to be extremely powerful in solving complex non-linear problems. The learner that is derived from the multi-layer perceptron ANN, is here used to obtain two-dimensional, turbulent production and dissipation values near the walls. Training examples of the dataset have been initially collected either from LES simulations of significant 2D test cases or have been found in open databases. Assessments on the morphology and the ANN training can be found in the paper. The ANN has been implemented in a Python environment, using scikit-learn and tensorflow libraries [93, 94]. The derived wall function is implemented in OpenFOAM v-17.12 [95], embedding the forwarding algorithm in run-time computations exploiting Python3.6m C_{Api} library. The data-driven wall function is here applied to k-epsilon simulations of a 2D periodic hill with different computational grids and to a modified compressor cascade NACA aerofoil with sinusoidal leading edge. A comparison between ANN enhanced simulations, available data and standard modeling is here performed and reported.

4.1.1 Introduction

Theoretical models for turbulence closure were historically derived by experimental observations of turbulent phenomena. Mathematics and statistics then followed, leading to an analytical formulation of the problem. Further observations eventually tuned the model, improving its robustness. However, this learn-from-observations approach was limited by the number of observations that could be considered at the same time. The growth of available computational power has made this approach available for huge datasets of turbulent flows, finally granting the capability of developing data-derived turbulent models [96].

In the past five years, a small but growing community has worked on improving existing turbulence models through a data-driven approach. All the works share a common feature, as they try to overcome the restriction imposed by the well-known Boussinesque closure in the RANS approach, granting a better approximation of the anisotropic part of the Reynolds stress tensor. Within all the machine learning techniques, neural networks have been used in turbulence modeling applications, thanks to their capability of realizing predictors as complex as needed. However, this complexity comes at a cost, as huge computational resources are required to generate a turbulence database that is detailed enough to support a proper algorithm training. To effectively achieve a data-driven model of turbulent flows, it is necessary to reach a compromise between the complexity of the considered flows and the amount of CPU time needed to generate the database. Due to this requirements, smart and general solutions to reduce the complexity of the problem have still to be found.

A dimensionality reduction of the considered flows is achieved, following the approach of [69]. Considering only two-dimensional flows, in fact, allows a faster data generation through two-dimensional period flow simulations, and it further reduce the complexity of the algorithm, as one third of the training feature are ignored. This also

increase the overall accuracy of the ANN algorithm, because as the number of considered features is increased over a certain number, the effectiveness of the predictor drops dramatically [97].
To speed up the convergence of the algorithm, a zonal training is used. In this way, only a portion of the computational grid is considered as significant for the training, leading to extreme reduction in the number of useless or redundant training examples. In the work, all generated training examples are filtered with a y+ cutoff value of 300.

4.1.2 Training Dataset

Training examples were collected from numerical simulations of three different test cases: a channel flow, a 2D periodic hill and a backward-facing step [98, 99]. For the first geometry, all the DNS data available from [100] at different Reynolds numbers are included in the database. The remaining data were generated with LES simulations.

4.1.2.1 Numerical Methodology

LES modeling relies on a dynamic one equation model from Davidson [101]. Both simulations were carried out with OpenFOAM v-17.12 using the PISO algorithm for velocity-pressure coupling, a second order time integration scheme, preconditioned conjugate gradient (PCG) for pressure and preconditioned bi-conjugate gradient (PbiCG) for U and k. Convergence tolerance was set to 1E-6 and 1E-7 for pressure and other quantities respectively. A fixed time-step was used. Courant number was run-time monitored, and it did not exceed 0.8 for the first iterations. Both simulations ran up to full convergence. Backward facing step inlet conditions were generated using a precursor channel. Statistics on the mean field were collected during the simulations.

4.1.2.2 Filtering

Once the initial database of $\approx$ 8M training examples had been generated, four different filters were applied in data preprocessing to reduce the number of redundant or non-significant observables (Fig.4.1).

- Time averaging: only the mean field of velocity was considered for the training of the algorithm. This filter dramatically reduced the number of observables;

- A dimensional reduction was performed through spatial averaging in the third dimension. It led to 125 000 training examples;
- A y+ cutoff set to 300, which selected 18 000 observations only;
- Using bins inside the computational domains leads to the final value of 2 300 examples.

Theoretically, the filtering operations listed above can be applied in any order, however spatially and temporal averaging grants a saving in computational time.

4.1.3 ANN Training

Eight significant input features was used to train a prediction model: the two components of mean velocity, the four velocity gradient components, kinematic viscosity and wall distance. Kinematic viscosity was added to include the effect of different Reynolds numbers on the predictor. The algorithm is trained on predicting turbulent kinetic energy on solid walls. The total turbulent kinetic energy (resolved plus sub-grid) has been previously computed in the numerical simulations through an in-house application, integrated in the solver. Within the family of machine learning algorithms commonly used for regression problems, a multi-layer perceptron neural network has been chosen [102]. The choice is justified by the

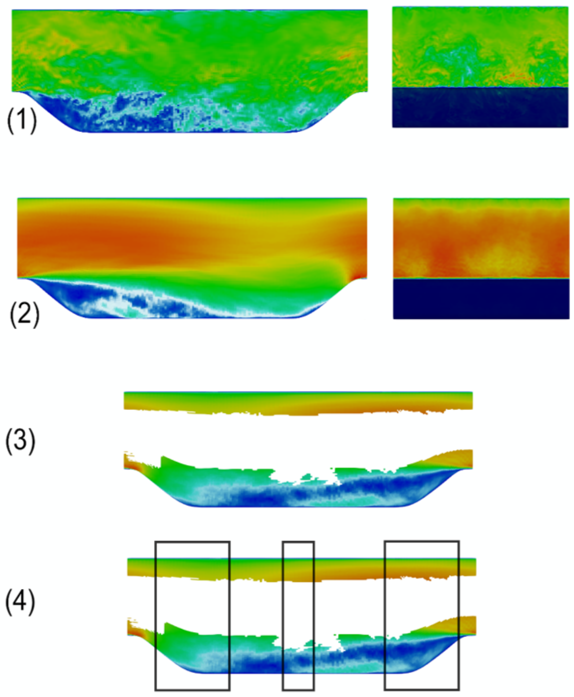


Figure 4.1: Effect of each data filter on 2D periodic hills

extreme complexity of the problem and by the high accuracy that can be reached through this approach . It has been implemented using Python 3.6 and Tensorfow libraries.

4.1.3.1 ANN Morphology and Hyperparameter Tuning

All the hyperparameters reported in Table 4.1 were heuristically tuned to grant the maximum accuracy and convergence speed. Input features have been preprocessed with a minmax criterion. A total of 25% of the initial dataset was used to test the algorithm. The final ANN entails 7 biased hidden layers. Weights wg have been initialized using a truncated normal distribution and biases using a constant value of 0.01. To grant weight initialization independency, algorithm training for the final configuration was run in parallel ten times, eventually choosing an intermediate result, as proposed by [73]. Instead of recurring to early stopping, the cost function has

Table 4.1: ANN Parameters

Nr. of neurons	30
Nr. of hidden layers	7
Nr. of training epochs	2000
Activation	Hyperbolic Tangent
Optimizer	ADAM
Cost Function	Modified MSE
Batch Size	600

been modified as proposed by [97] to include a weight decaying effect, with $\lambda = 1\text{E-}5$. It is able to reduce predictor overfitting for a high number of training epochs.

$$\text{MMSE} = \frac{1}{2N} \sum_{i=1}^{N} \left((\hat{y}_i - y_i)^2 + \lambda \left(w_i^2 + b_i \right) \right)^2 \qquad (4.1)$$

Sigmoid function was used as activator for all but the last layer, which is activated through a linear function. Adaptive Moment Estimation Gradient Descent (ADAM) was used to optimize the cost function, that automatically applies a learning rate decaying effect. Batch feeding is used to speed up the convergence of the algorithm, as weights and biases are updated four times for each training epoch.

4.1.3.2 Training Results

Fig. 4.2 shows the convergence history of the algorithm. Final cost is stably below 1.0E-6. The test dataset was forwarded during the training, to monitor overfitting. Predictor shows a high capability in predicting turbulent kinetic energy for all the test examples, with

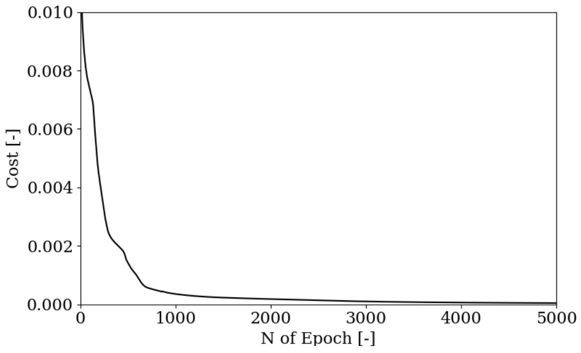


Figure 4.2: Algorithm convergence history

an overall accuracy of $\approx$ 97.9 $\%$ (Fig. 4.3). Fig. 4.4 highlights how the chosen number of training epochs is enough to provide a complete model training. After the training, test examples were forwarded to guarantee the absence of model overfitting.

4.1.3.3 Forwarding Algorithm and Implementation in the Solver

The trained weights and biases have been eventually stored and used in an algorithm that only performs a prediction of the turbulent kinetic energy. This algorithm has been embedded in OpenFOAM v-17.12 as a custom boundary condition. The Python algorithm has been directly embedded in the C^{++} code using the Python $\mathrm{C_{Api}}$ library. A scheme of how the modified solver is able to determine wall values is reported in Fig. 4.5.

4.1.4 Test Application: 2D Periodic Hills

The first application of the machine-learnt wall function is here reported. It is applied to the very well known test case from Temmermann and Leschziner at $\mathrm{Re}_h = U_b \cdot h/\nu = 10595$. Two $k - \varepsilon$ simulations on the same geometry were carried out using different

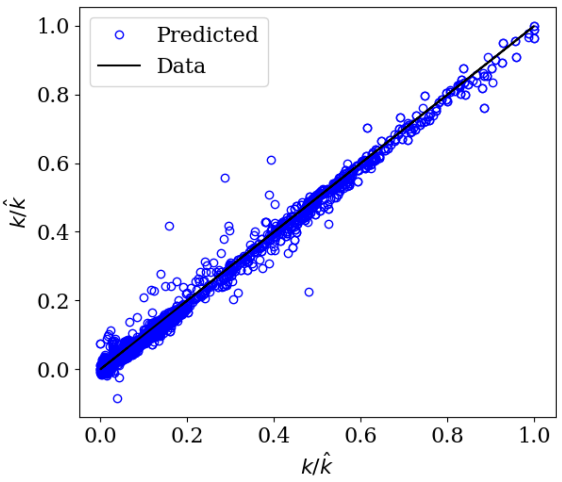


Figure 4.3: Predicted turbulent kinetic energy (blue dots) vs input data (solid black line)

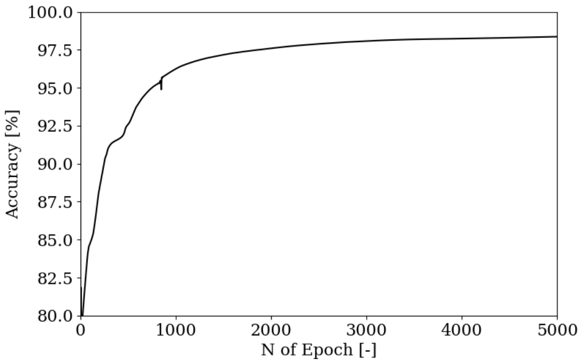


Figure 4.4: Overall accuracy vs Number of Training Epochs

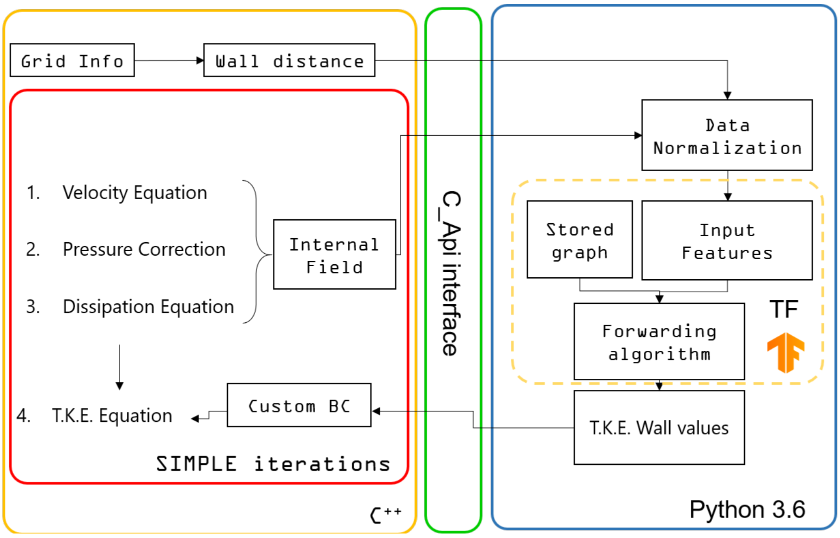


Figure 4.5: ANN-enhanced turbulent solver

wall modeling: one that computes turbulent kinetic energy from the standard expressions of $k - \varepsilon$ wall function [103] and the other that is derived from the ANN.

4.1.4.1 Numerical Methodology

To highlight the capability and the limits of this approach, three different computational grids were used for the same simulation (Tab. 4.2). They differ in the number of cells and in the first cell placement.

Span- and stream- wise periodicities are imposed through the Arbitrary Mesh Interface (AMI) coupling. In all the simulations, the flow is driven through a momentum source. No-slip boundary conditions are imposed for velocity on the two solid walls. Steady state calculations with the simpleFoam solver ran up to full convergence in approximately the same CPU time.

Table 4.2: 2D Hills computational grids

Case	Cells	First Cell Distance	Expected y^+
A	$300 \times 16 \times 2$	0.05	30
B	$300 \times 32 \times 2$	0.02	10
A	$300 \times 64 \times 2$	0.015	1

4.1.4.2 Results

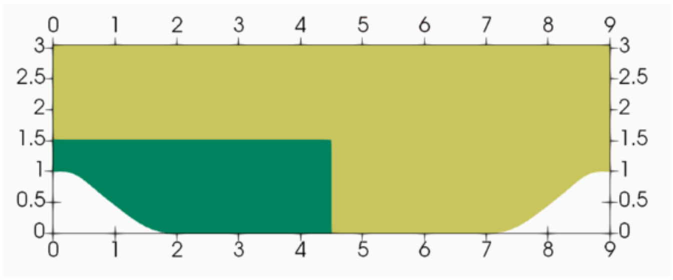
Streamlines for the various computations are shown in Figure 4.6. A true comparison with results in literature is challenging, because flow features are strongly influenced by grid refinements and SGS models used. Available LES simulations indicate the reattachment length to be between 4.5 and 4.7h, while experiments report a value of 4.21h [104], with h the height of the hill.

In this case results are strongly affected by grid refinement, as Table 4.3 suggests. Predicted values forreattachment for both models perfectly falls between the commonly reported range of x=3.8h and 4.8h for the finer grids, while grid resolution for simulation A seems insufficient for the standard WF to fall in that range, while ANN seems able to work also on this very coarse grid. The ANN modeling seems generally responsible of an under-prediction in the recirculation amplitude if compared to the standard $k - \varepsilon$ model.

Table 4.3: Reattachment length for the three grids

Grid	Reattachment length [x/h]	
	ANN WF	$k - \varepsilon$ WF
A	4.7	5.4
B	4.1	4.3
C	3.8	4.2

Here it must be highlighted how the ANN-WF seems superior to the standard wall treatment regardless of first cell placement.



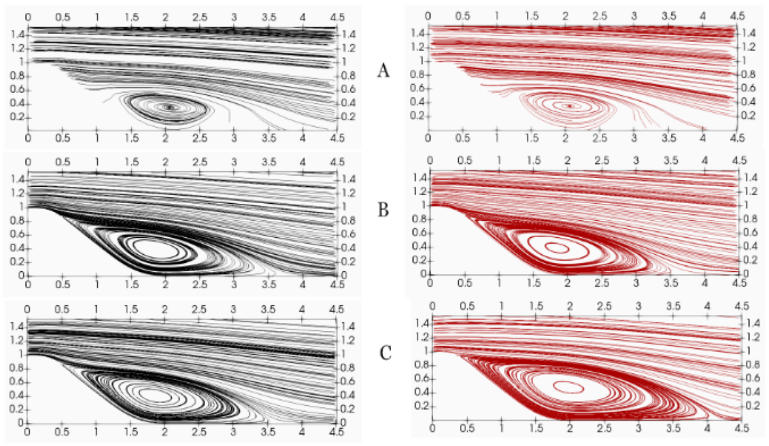
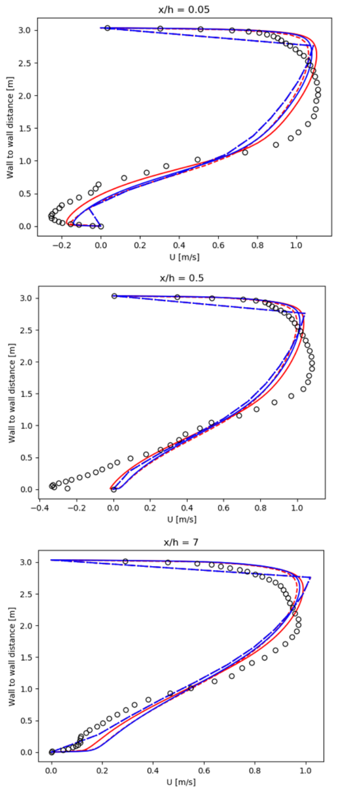
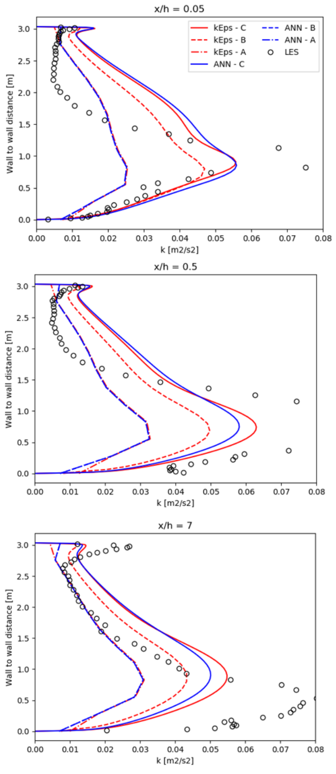


Figure 4.6: Velocity streamlines for the three grids. ANN-WF (left), $k-\varepsilon$ WF (right)

For velocity distributions, the change in turbulent kinetic values on the solid end-walls strongly affects the overall distributions, and regardless of the streamwise coordinate, the ANN-WF is providing a better match with the highly resolved LES. With the coarser mesh (grid C), the ANN is able to partially recover the correct trend near the wall, correcting the standard model. For the other two cases, differences are less marked, but the ANN is partially applying a correction to the turbulent kinetic energy that is predicted near the wall. Figure 4.8 shows that the ANN-WF far from the solid

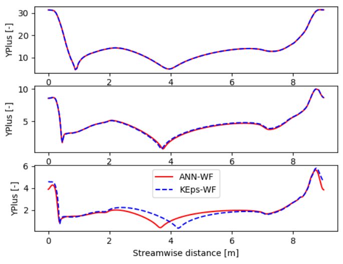


(a) Spanwise velocity

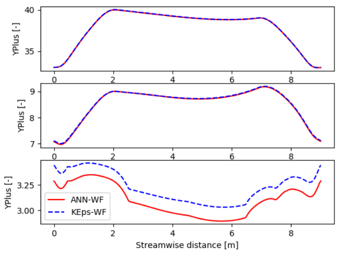


(b) Turbulent kinetic energy

Figure 4.7: Wall-to-wall profiles for x=0.05h, x=0.5h and x=7h, ANN-enhanced (blue) vs standard $k-\varepsilon$ wall function (red).



(a) Lower wall (y/h=0)



(b) Upper wall (y/h=3.125)

Figure 4.8: Streamwise distributions of y^+ values along the solid walls.

walls presents a profile similar to that of the standard wall function. Near the solid walls it seems able to correct the predicted turbulent kinetic energy values, providing a better match with the LES simulation. This is especially true for case A, where the standard model completely fails.

Fig. 4.8a reports the streamwise distributions of y+ values along the lower endwall: in the coarse grid y+ streamwise distribution is identical with for approaches. As the grid is refined, the recirculation length (represented by the lower peak at x/h=4) gradually shifts to the inlet of the domain. This trend is slightly evident in case B, however it is strongly visible for the finest mesh. Fig. 4.8b reflects the trend previously seen for the lower wall. The curves perfectly match for the coarse mesh, they become slightly different for case B, while in case C they differ in magnitude. In this the case shape of the curve is conserved by the ANN-WF, which is only affecting the y+ distribution through a reduction of magnitude of 10 %.

4.1.5 Test Application: Sinusoidal NACA 4415

The adaptive wall capabilities were stressed to the limits with a test on a modified NACA4415 airfoil, characterized by a sinusoidal

leading edge. This peculiar profile is adopted in fan applications to control noise emissions and stall inception. Test data for the isolated airfoil are available from [105].
The sinusoidal amplitude of the leading edge is equal to 2.5$\%$ of the chord. Fig. 4.9 shows a comparison between the modified profile and the base NACA 4410. Both airfoils were treated in a cascade configuration, with a solidity of $\sigma = 1.3$. This choice is motived by the fact that the ANN-WF has been trained using data from internal flows only, so a comparison based on external aerodynamic would be unfair or not really significant. The ANN is expected to perform worse than the standard model, especially in the modified geometry, due to the mismatch between the full three-dimensionality of the problem and the training data constituted only by 2D test cases. However, the choice was motivated by the author's intent to stress the capability of the model.

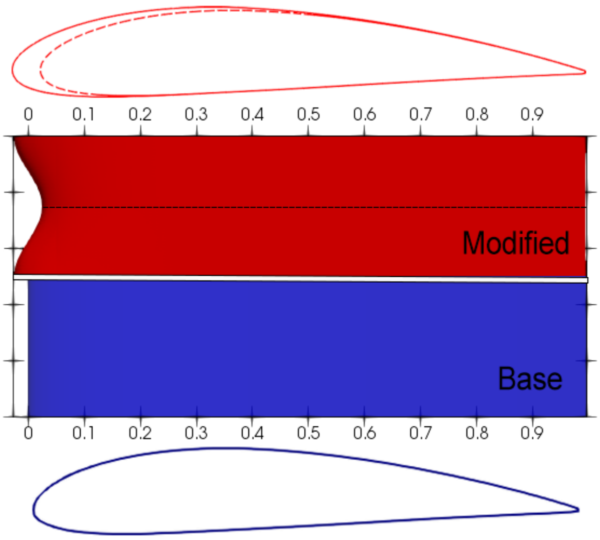


Figure 4.9: Modified NACA 4410 vs base blade profile

4.1.5.1 Numerical Methodology

The computational domain entails 46.5k cells for the baseline NACA4415, and 1.5M hexahedral cells for the sinusoidal version. Grid spacing is set to obtain $y^+ \approx 1$ for the first cell on the airfoil surfaces. The difference is caused by the fact that the base airfoil can be treated as two-dimensional and the number of cells in the azimuthal direction can be limited to only one. Each side of the airfoil is discretized using 120 cells. The computational domain extend 1.3c and 9.3c upstream and downstream of the chord respectively, and 0.5c in the spanwise direction. Inlet velocity magnitude was set to 1.0 m/s, with an inlet angle of attack of 10 deg. The turbulent kinetic energy intensity was set to 1.5$\%$. Wall treatment relied on no slip condition for velocity, on the standard epsilon wall function for ε. Two different wall treatments for k were tested: the standard k-wall function and the adaptive one. Periodicity conditions were imposed in pitch- and span- wise directions. A Convective outlet condition was set at the outlet. Turbulence closure relied on the standard k-ε model.

4.1.5.2 Results

Tab. 4.4 reports the results for lift and drag coefficients. Cascade data are not available in literature, however they are derived using an in-house code [106].

Table 4.4: Reattachment length for the three grids

	Baseline			Modified w L.E. Bumps		
	ANN-WF	$k-\varepsilon$ WF	Ref.	ANN-WF	$k-\varepsilon$ WF	Ref.
C_L	0.07029	0.07022	0.0711	1.32509	1.37819	1.311
C_D	0.00262	0.00260	0.00259	0.14192	0.13299	0.139

For the baseline airfoil, the difference between the ANN WF and the standard modeling is close to zero, with both values close to the calculated reference values. Surprisingly, in the modified geometry standard k wall function is still performing worse. Results from the

ANN are slightly different, with an underprediction of 3.8 % and an overestimation of 0.6 % in the lift and drag coefficients respectively. This trend is reflected through the pressure distribution along the chord of the airfoil, Fig. 4.10. Variations in resultant pressure distribution for the base NACA4415 are close for both wall treatments, so they collapse on the same curve. Dashed results are taken in a section of the airfoil where the chord length is coincident with the base airfoil. There, the ANN modeling is able to correctly reproduce the slope of the curve, like the cuspids at c=0.01, however the magnitude is completely different.

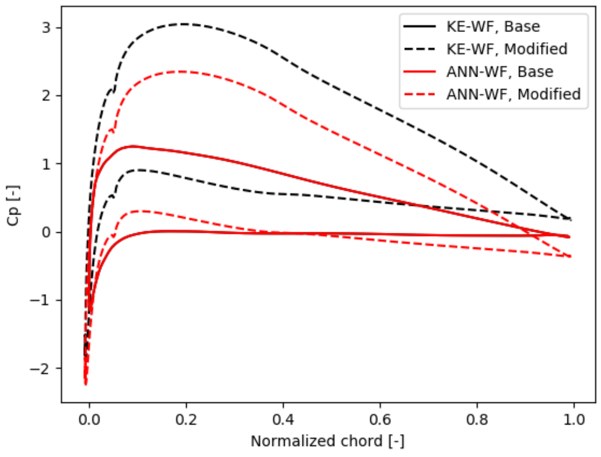


Figure 4.10: Pressure coefficient distribution vs normalized chord for a) solid black line k-ε-WF base blade, b) dashed black line k-ε-WF modified blade, c) solid red line ANN-WF base blade, d) dashed red line ANN-WF modified blade.

Fig. 4.11 shows velocity streamlines for the base airfoil for AOA = 10deg. Both simulations provided the same identical distribution, hence only one is reported here. In Fig. 4.12, the velocity streamlines for the two wall treatments lead to different results. In particular, ANN seems responsible of a different prediction in terms of the velocity magnitude on the suction side of the blade

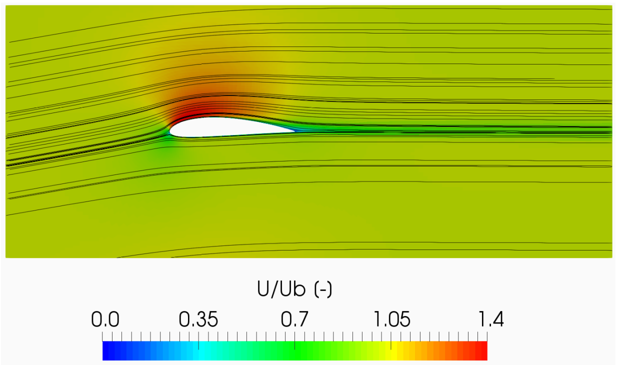


Figure 4.11: Velocity streamlines for base airfoil.

of 10%. This behavior is fully justified by the fact that the ANN is using only four components of the velocity gradient and two of the velocity field as input features, leading to an incorrect behavior of the predictor.

4.1.6 Final Remarks

An artificial neural network was trained to derive turbulent kinetic energy values from the following inputs: wall distance, velocity, velocity gradient and turbulent kinetic energy. The generation of a turbulent database was necessary to provide enough training examples, and data was collected from LES simulations of simple turbulence test cases. The application of a y+ filter and other simplifications allowed a reduction of the complexity of the problem. An optimization of the algorithm morphology was conducted, exploiting Tensorflow library. Smart solutions, i.e. batch feeding, feature normalization and weight and learning rate decays, granted the creation of a proper predictor. The derived predictor, transformed into a forwarding function, was embedded in OpenFoam v17.12 in a custom boundary condition to perform run-time computations. The custom boundary was able to derive turbulent kinetic values on the wall during iterations, in approximately the same CPU time

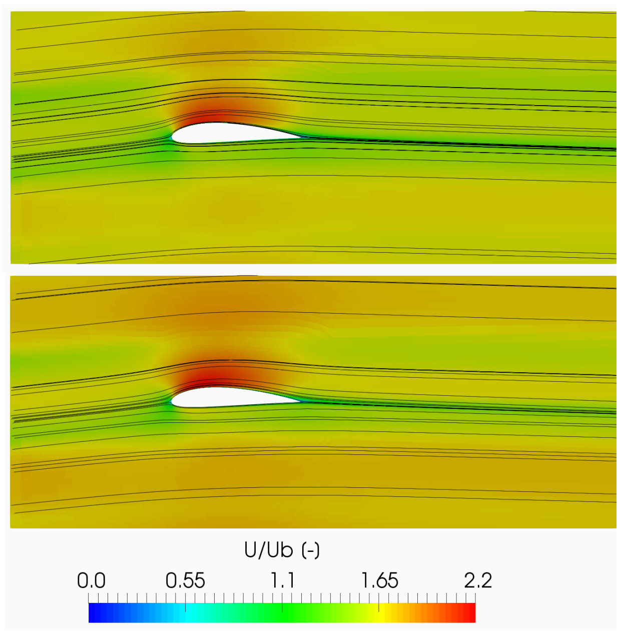


Figure 4.12: Velocity streamlines for modified airfoil, for a) standard k wall treatment, b) ANN wall function.

as the standard expression of the k wall function.
The ANN wall function was firstly applied to a 2D periodic hill flow. Three different grid resolutions were tested, with an expected y^+ values on the first cell of 1, 10 and 30 respectively. The aim was to compare performances of the ANN wall function with the standard wall treatment. The predicted values were really close for the coarse and intermediate meshes, while the coarsest grid was not able to correctly reproduce the highly resolved LES simulation. In the finest grid, ANN performed better. Streamwise velocity profiles and reattachment length are reported in the manuscript. An analysis of y^+ values versus the streamwise direction showed a similar behavior of the two models, with the greatest differences in the finest grid,

with the ANN underpredicting the reattachment length.
Eventually, the ANN was also applied to a completely different geometry: a NACA4410 profile. A modified 3D airfoil with leading edge bumps was considered in order to stress the ANN capability. This aerodynamic profile is commonly in use in the fan industry, and both airfoils were tested in a cascade configuration with an angle of attack of 10 deg. For the base profile, results of both models are practically identical. In the modified version of the airfoil, the ANN performed sensibly better if compared to the standard $k-\varepsilon$ which shows an overprediction in the lift coefficient and an underprediction in the drag force. This result is quite surprising, considering that the training of the model was achieved through 2D simulations only.
This attempt to create a data-driven turbulent wall function could be further developed to include three-dimensional flows and more physical phenomena in the training database, however preliminary results shows capability far beyond the standard approaches, that were limited by the number of observation performed.

4.2 A Data-driven Wall Function for Rotating Passages

Data-driven tools and techniques have proved their effectiveness in many engineering applications. Machine-learning has gradually become a paradigm to explore innovative designs in turbomachinery. However, industrial Computational Fluid Dynamics (CFD) experts are still reluctant to embed similar approaches in standard practice, and very few solutions have been proposed so far. The aim of the work is to prove that standard wall treatments can obtain serious benefits from machine-learning modeling.
Turbomachinery flow modeling lives in a constant compromise between accuracy and the computational costs of numerical simulations. One of the key factors of that process is defining a proper wall treatment. Many works point out how insufficient resolutions of boundary layers may lead to incorrect predictions of turboma-

chinery performances. Wall functions are universally exploited to replicate the physics of boundary layers where grid resolution does not suffice. Widespread wall functions were derived by the observation of few canonical flows, further expressed as a simple polynomial of Reynolds number and turbulent kinetic energy. Despite their popularity, these functions are frequently applied in flows where the ground assumptions cease to be true, such as rotating passages or swirled flows. In these flows, the mathematical formulations of wall functions do not account for the distortion on the boundary layer due to the combined action of centrifugal and Coriolis forces.
Here we will derive a wall function for rotating passages, through means of machine-learning. The algorithm is directly implemented in the N-S equations solver. Cross-validation results show that the machine-learnt wall treatment is able to effectively correct turbulent kinetic energy field near the solid walls, without impairing the accuracy of the RANS turbulence model in any way.

4.2.1 Introduction

Turbulence modeling through big data is raising extreme expectations in the scientific community [28]. Machine-learning, in particular, has been exploited in the past few years to overcome model deficiencies in Reynolds Average Numerical Simulations (RANS). For example, Duraisamy and his co-workers successfully improved the Spalart-Allmaras model in simple configurations [67, 66]. That methodology involves ad-hoc data-driven corrections to the anisotropic part of the Reynolds stress tensor or to the dissipation term in the nutilda equation. Similar approaches were also applied to two equation models, e.g. [70, 48]. J. Wu clearly addressed the pros and deficiencies of machine learning in fluid dynamics by looking at the tensorial representation of turbulence [80].
In turbomachinery simulations,the RANS approach predominates as workhorse. Despite this, we find fewer applications of machine learning to turbomachinery design and optimization. The reason

lies in the flows themselves, that present critical aspects such as wall-bounded flows, recirculation, rotation, high curvatures or high Reynolds number. Obtaining high fidelity data becomes extremely time consuming and difficult. Standard models, however, still fail to capture that complex physics, leading to incorrect prediction, as addressed by Denton in [19].
One of the problematic aspects to improve lies in the generality of models. Most of the models work in flow configurations that are statistically similar to the training data, constituting therefore ad-hoc solutions. Interesting progresses toward this final goal can be attribute to J. Ling and her co-workers, who found a smart and physics-based strategy to achieve frame of reference independency [73, 72]. They improved Reynolds stress tensor prediction through multi-layer perceptrons. A similar approach, using decision trees and random forests and applied to heat transfer problem, is followed by Milani in [75, 74]. Sandberg et al. applied genetic programming to Explicit Algebraic Stress Tensor Models (EASMs) in different slopes of trailing edge slots commonly used for film cooling, finding results closer to Large Eddy Simulations [77, 79]. In [78], Wheateritt design a new methodology, comparing performances of neural networks and genetic algorithms in a crossflow. Works inheriting near wall modelling and wall functions are scarce in literature, despite the importance of the topic. Wall functions are commonly accepted in RANS practice as instruments to reduce computational costs and to ease the meshing process. Milano tries to find similarity in flows close to the wall through non-linear Principal Component Analysis [83]. In 2019, Tieghi et al. derived a data-driven expression of a wall function through neural networks, improving performances of k-ε model in recirculating flows [82]. To the best of the authors' knowledge, no previous work can be found on machine-learnt enhanced wall treatments developed for rotating flows. Experimental observations show that, within a certain range of the Rossby number, the well-known law of the wall ceases to be true due to the combined effect of centrifugal and Coriolis forces [10]. Those rotational speeds are unfortunately compatible with common low-speed subsonic turbomachinery. Here we will focus on

diverging rectangular ducts with different Rossby numbers. Preliminary investigations report that standard two-equations turbulence models and wall treatments are inadequate to properly predict flow within the passage, the superimposition of flow separation and rotating forces.
A multi-layer neural network (MLNN) was trained to provide a correction for the turbulent kinetic energy at the wall, effectively substituting the standard wall treatment. MLNNs are suitable for this application, thanks to their non-linearity and high accuracy. A LES simulations of rotating ducts at a Rossby number equal to 0.41 is performed to build a significant training database. A single axis of rotation is here considered. Rossby numbers were selected to match with common reported values for fan applications. Input features were derived by locally normalizing combinations of raw flow fields. Exploratory data analysis assured quality of data and leads the process of feature selection/combination. The algorithm was trained and optimized using the Keras library [107]. The derived predictor was embedded in runtime computations of OpenFOAM v.18. Interface between C^{++} and Python 3.6 relied on the C_api library. Advanced solutions was applied to reduce training time and to prevent overfitting. The model is eventually validated through a RANS simulation of a rotating duct at a lower Rossby number. Turbulence closure is performed using the k-ε model. Test data is not included in the original dataset to better understand generality of the model. In the lasts part, a toe to toe comparison between the derived predictor and standard OpenFOAM's kWallFunction is reported. Computational time remained the same between the two SimpleFoam solvers.

4.2.2 Data Generation

The training datasets were generated through numerical simulations of a rotating radial diffuser. This problem is one among the most important in centrifugal compressors and it has been well documented and studied since the ′40s [3, 108, 109]. The geometry, illustrated in 4.13, features a squared inlet, that gradually diverges up to three times the inlet section (4.14). In addition, 4.5 gives a summary of the main duct dimensions. Dimensions of

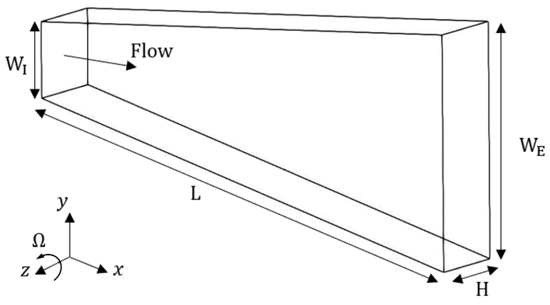


Figure 4.13: Schematic of test section and coordinate system

the diffusing duct is similar to the test section of the experimental study performed by Moore [3]. With respect to those results, we performed the simulation in kinematic similarity.

Table 4.5: Domain dimensions

Length	L	0.6096 m
Inlet Width	W_I	0.0762
Exit Width	W_E	0.2362
Height	H	0.0762
Included Angle	α	15 deg

Simulation of the whole geometry of Moore is computationally too expensive, due to the dimensions of the domain coupled with the high Reynolds number (4.6E7). We therefore decided to perform a numerical simulation of the diverging passage only, without any converging duct at the inlet. 4.6 reports the flow operating parameters for all the numerical simulations.

Table 4.6: Domain dimensions

Volumetric flow rate	Q	0.097 m^3/s
Relative inlet velocity (positive x axis)	U_I	16.72 m/s
Relative exit velocity (positive x axis)	U_E	5.39 m/s
Rotational speed	Ω	21.6 rad/s
Included angle	α	15 deg

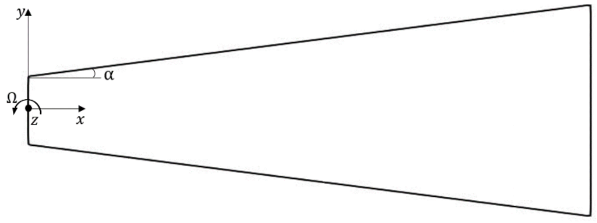


Figure 4.14: View of the diffuser

4.2.2.1 Numerical methodology

Numerical simulations were carried out in OpenFOAM v-18. LES simulation relied on dynamic one equation model from Kim and Menon [110] using PISO approach for velocity-pressure coupling, a second order time integration scheme, geometric-algebraic multi-grid (GAMG) for pressure and preconditioned bi-conjugate gradient (PbiCG) for U and k. Moreover, the RANS simulation relies on Launder-Sharma model [111] using the SIMPLE approach, with geometric-algebraic multi-grid (GAMG) for pressure and a smooth solver (GaussSeidel) for U, k and ε.

The fixed time-step of 1.0E-5 seconds was led to a maximum and mean Courant Number of 0.77 and 0.127 respectively. Grid resolution is $600 \times 85 \times 85$ cells. Wall distance of the first cell is set to achieve a y+ < 1. Proper turbulent inlet conditions was generated by means of a precursor squared duct, that entails $91 \times 85 \times 85$ cells. Volumetric flow rate is forced thorough a source momentum at the

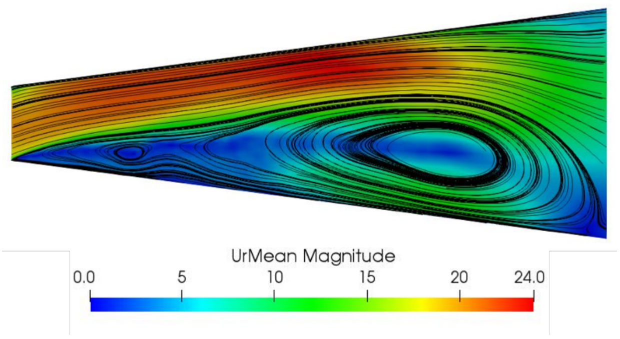


Figure 4.15: Mean relative velocity field

inlet of the square duct to match experimental flow rate.
Statistics on the mean field were collected during the simulations and exploited to assess the convergence of the simulations; 4.15 shows the mean relative velocity field. Boundaries remains the same between the RANS and LES simulations, with the exception of ε obviously not present in the latter (4.7).

Table 4.7: Boundary conditions

Patch	U	p	k	ν_t	ε
Inlet	mapped	ZG	mapped	mapped	mapped
Outlet	ZG	TP	ZG	ZG	ZG
WallS	NoSlip	ZG	kLowRe	nutLowRe	εWF

Grid independency is observed by looking at the resolved TKE in the LES simulation. To ensure that majority of turbulence scales are well resolved, the ratio between resolved and the total TKE (resolved plus modelled) needs to be above 0.8 [112]. A cross-section, shown in 4.16, reports the fulfillment of that requirement.

Under the same Reynolds number, RANS approach requires an inferior grid refinement compared to LES simulations, therefore computational grids remained the same within both approaches.

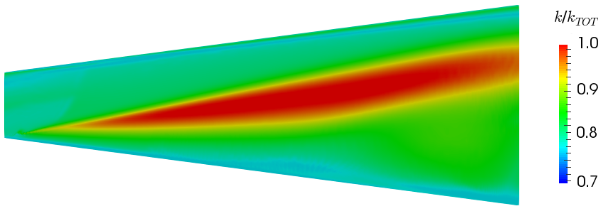


Figure 4.16: Cross-section view of the ratio between resolved and total TKE

4.2.2.2 Flow Description

In a non-rotating diffuser with the same geometric configuration, the observed flow was perfectly symmetric. As the passage underwent rotation, the flow features changed dramatically.
One of the metrics which illustrates the most peculiarity of the flow in a diffuser is the static pressure recovery. It can be computed by the difference between pressure on a solid wall and the pressure value at the inlet of the domain. In a rotating frame of reference, the reduced pressure recovery, that takes into account rotating forces, is commonly exploited. It is obtained by subtracting the centrifugal pressure field to the static pressure, i.e. $p - 0.5\rho\Omega^2 R^2$ [10]. Figure 4.17 reports the coefficient for the upper (*suction side*) and lower (*pressure side*) solid walls. It qualitatively proofs the pressure unbalance between pressure and suction surfaces. The delta between the two curves remains almost constant between the two curves, especially close to the outlet of the domain. As such, it can be inferred that the simulation is uninfluenced by the exit boundary conditions.

4.2.3 Data Preprocessing

4.2.3.1 Data Sampling and Filtering

Our algorithm was designed to provide a wall treatment. That involves that LES simulation generates a database of $\approx$ 6M that also entails zones of the computational domain of no interest. This could

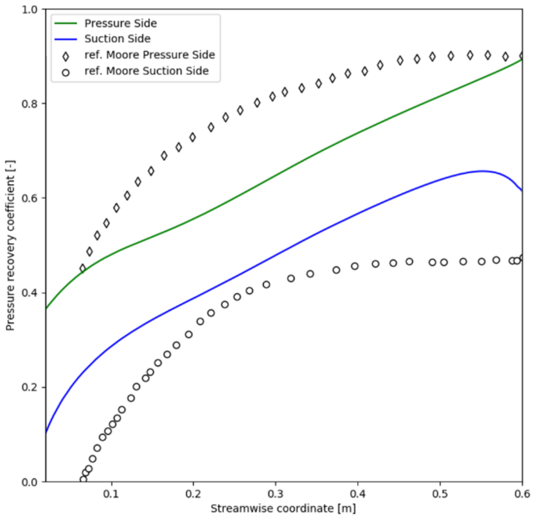


Figure 4.17: Static pressure recovery along at mid-height of passage as a function of the streamwise coordinate

worsen the algorithm accuracy/learning phase. Therefore, two filtering operations were performed (Figure 4.18). The precursor was not taken into account for data collection at any point.

1. A time averaging filter leads the LES quantities to stationary conditions and greatly reduces the number of training examples, through considering the mean field only for the algorithm training.

2. A selection of 2M observations was made by imposing a y+ cut-off of 400, in this way it is possible to train the model only near the walls, avoiding training it even in the core of the flow.

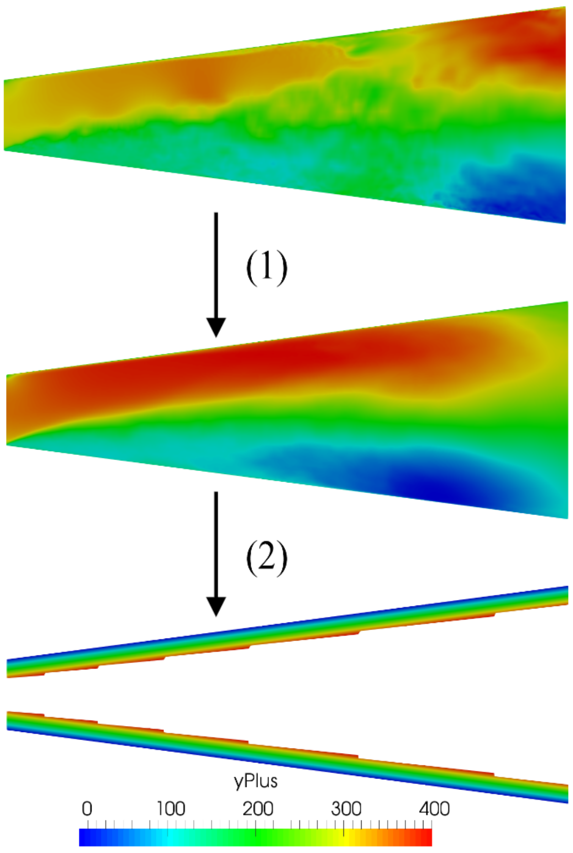


Figure 4.18: Visualization of the effect of data pre-processing on a cross-section

4.2.3.2 Features creation and normalization

Enhancing a turbulent model through machine learning presents serious difficulties in terms of feature selection. We define as feature any observed characteristic of the flow. Three key issues regarding feature selection were addressed during the work:

1. *Unavailability of data.* The first problem lies in the map between high fidelity data (LES/DNS simulations or experimental observations) and low-level fast approaches like RANS simulations. Within all the features that are available to the model, in fact, it makes sense to select the ones that can be derived from RANS computations. Small scales of LES simulations or non-stationary fields, for instance, cannot be derived during RANS simulation and find no direct application to the predictor. Therefore, only mean fields will be considered hereafter as observables.

2. *Dependence on the frame of reference.* Simple geometries, where one direction is predominant, e.g. mono-dimensional or periodic flows, can be easily modelled without too much efforts. On the opposite, complex flows can present local regions where the amplitude of secondary flows is comparable to the core of the flow. That means that a model needs to capture relationships within the features respecting the Galilean invariance. Therefore, the magnitude of vector and tensor fields was used as a predictive factor instead of the single components.

3. *Feature scales of magnitude.* Neural networks are strongly impaired whenever features present high differences in magnitude. This behavior can lead the predictor to neglect the influence of features with the smallest numerical values, e.g. turbulent kinetic energy vs dissipation rate. To avoid this, in standard machine-learning practice, features are normalized with respect to a combination of statistical parameters like mean or min/max values. Unfortunately, this is not the best approach in fluid dynamics, as flow fields can present sharp probability density functions. Standard normalizations treat the least represented scales in the data as outliers, while for our purposes these can be highly significant. Following [67], we will perform a local normalization for the j-th feature,

namely:

$$F_j = \frac{\hat{F}_j}{\hat{F}_j + k_j} \tag{4.2}$$

where $\hat{F}_j$ denotes the unnormalized values and k_j is an ad-hoc term of normalization.

The previous points lead the initial selection of features, reported in Table 4.8.

Table 4.8: Features Normalization & Creation

Name	Derivation	Normalization Type (Factor)
mag(U)	$\parallel U \parallel$	Local(U_I)
epsilon	ε	Local(U_I^3/L)
∇P	$\sum_i \partial p/\partial x_i$	zScore
∇k	$\sum_i \partial k/\partial x_i$	zScore
nut	ν_t	Local(ν)
yPlus	y^+	minMax
sum(S)	$\sum_{ij}(\partial U_{ij}/\partial x_j + \partial U_{ij}/\partial x_i)$	zScore
sum(Ω)	$\sum_{ij}(\partial U_{ij}/\partial x_j + \partial U_{ij}/\partial x_i)$	zScore

where S and Ω are the symmetric and the asymmetric components of the decomposition of the gradient of velocity.

4.2.4 Exploratory Data Analysis

The feasibility of the former feature selection was checked through Exploratory data analysis (EDA). The final aim of the model is to perform a correction in the turbulent kinetic energy near the walls. The standard calculation for TKE in LES simulations can be written as:

$$k_{\text{LES}} = \frac{1}{2}\left(u'u' + v'v' + w'w'\right) \tag{4.3}$$

where u', v' and w' denote the three components of velocity fluctuations. In the Launder Sharma turbulence model [26], k_{RANS} is derived by solving the transport equation:

$$\frac{\partial}{\partial x_j}\left[\rho k_{\text{RANS}} u_j - \left(\mu + \frac{\mu}{\sigma_k}\right)\frac{\partial}{\partial x_j} k_{\text{RANS}}\right] = P - \rho\varepsilon - \rho D \tag{4.4}$$

with P and D read as:

$$P = \tau_{ij}^{t}\frac{\partial u_i}{x_j} \tag{4.5}$$

$$D = 2\nu\left(\frac{\partial}{\partial y}\sqrt{k_{\text{RANS}}}\right) \tag{4.6}$$

In so doing, the algorithm is trained on the term δ_k, defined as:

$$\delta_k = k_{\text{RANS}} - k_{\text{LES}} \tag{4.7}$$

Figure 4.19a reports frequency plot for the turbulent kinetic energy for both approaches.

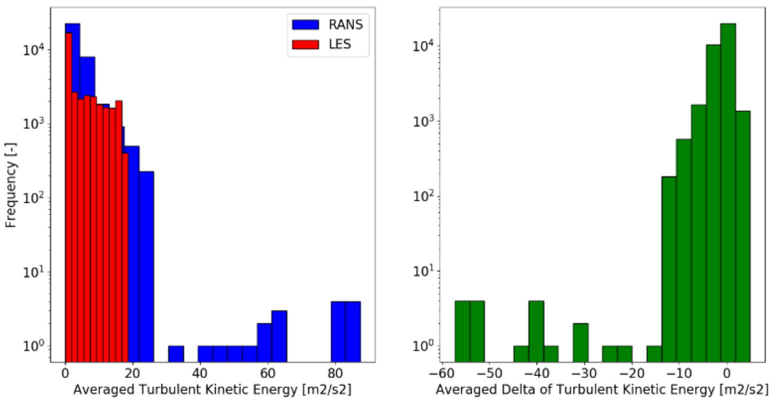


Figure 4.19: Frequency plot of: TKE for LES and RANS (a) - δ_k (b).

We can easily observe that TKE values in the RANS simulation are higher than the one in the high-fidelity approach. This is reflected by the distribution of δ_k, that shows a predominance of negative values (4.19b). To effectively build a predictive model, we must assess that there is no evident predominance between input variables and the output. In so doing, we ensure the non-predominance of any of the feature. In addition, neural networks are not suited for working with datasets that present variables that present mutual direct correlations, that are instead automatically transferred to the final model. A correlation analysis of training data has been performed and reported in Figure 4.20. We can

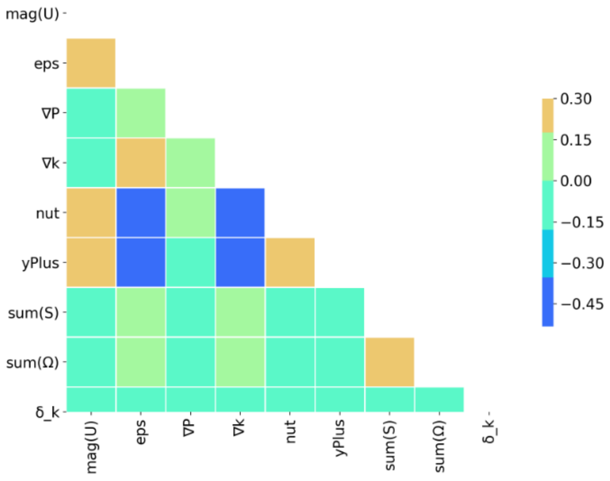


Figure 4.20: Correlation map for the eight input features.

observe that all the input features are statistically uncorrelated with the output of the model (last row in the heat map). The maximum reported value of inverse correlation is -0.17, given by the magnitude of velocity. That value is completely acceptable and should not impair the predictor. The maximum value of correlation among the input features is 0.27 and is therefore acceptable. It is attributed to the relationship between the turbulent viscosity and the magnitude of velocity. The higher magnitudes of minimum values of correlations is -0.38 and -0.365, are given by epsilon-yPlus

and grad(k)-yPlus respectively. Negative correlations have a lower impact on the model effectiveness [29]; therefore, these values can be tolerated.
A final check on the quality of data was performed through boxplots (not shown here), to verify the absence of outliers that could impair the accuracy of the model.

4.2.5 Data-driven Modeling

The training dataset eventually embedded 2M data entries. A multi-layer perceptron neural network has been chosen to build the regressor. Our choice was led by previous works on the topic that highlighted their effectiveness. Modelling relies on a multi-layer multi-input feed-forward neural network. The algorithm was implemented with Python 3.6, exploiting Keras open-source library. 4.9 reports the hyperparameters of the final setup of the network. Initialization of the model is performed through a Xavier uniform initializer for weights, while biases had a constant starting value of 0.01. Multiple training instances of the model were run, eventually choosing an intermediate set of parameters. Initial number of neurons, the same within all the layers, was set to 6, equal to the sum of half of the input features plus one. This number has been increased in the final network architecture to 16 to obtain a higher accuracy.

Table 4.9: Hyperparameters of the network

Nr. of neurons	16
Nr. of hidden layers	5
Nr. of training epochs	30
Activation	Tanh
Initial learning rate	1E-5
Optimizer	ADAM
Cost Function	MSE
Batch size	1/25th

The number of hidden layers was chosen based on the cross-correlation results. Each biased layer is activated by a hyperbolic tangent function. Gradient of cost function is backpropagated and optimized with respect to weight and biases through Adaptive Momentum Estimation Gradient Descent (ADAM). ADAM self-optimizes learning rate epoch by epoch, without any need of decaying effect [90]. A percentage equal to the 25% of the initial entries were kept as a test dataset to monitor overfitting during the training. The Training dataset was randomly shuffled before each instance of training. Batch feeding sped up the process of learning. The batch size was set equal to 1/25th of the total dataset, according to common reported values. Trends in Figure 4.21 highlight that the model is not overfitting data. The Waviness of the training curve is caused by the presence of batches.

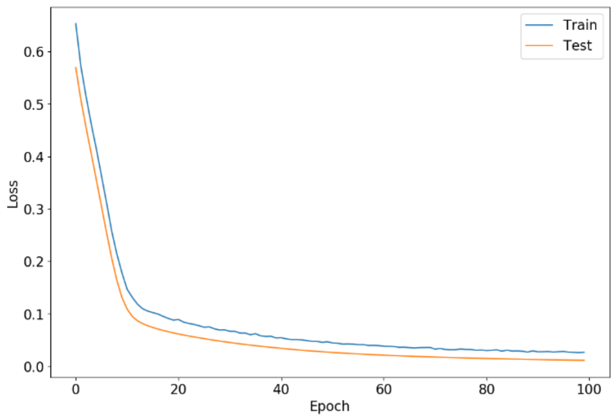


Figure 4.21: Mean squared error for test train and test dataset as a function of training epochs.

Figure 4.22 shows normalized values of TKE that were predicted from the algorithm with respect to training data. Within the whole range, relative errors remain far below 1.0%, therefore the two curves appear mostly superimposed. That verifies either the initial assumptions, e.g. feature selection, and the network architecture and hyperparameters selection, as the training was successfully achieved.

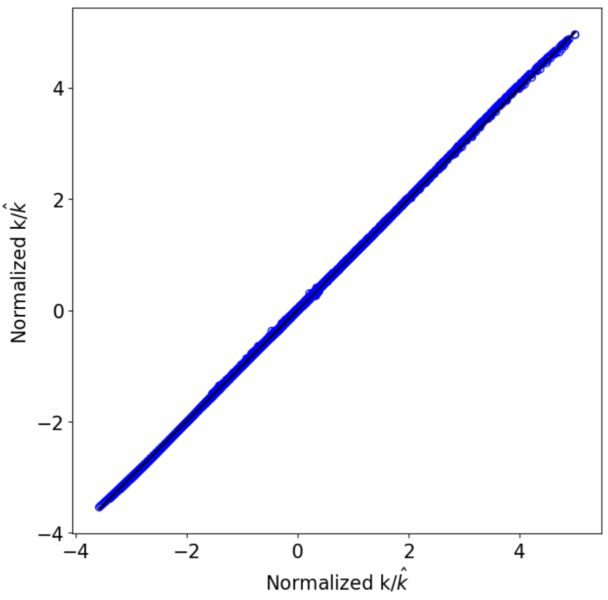


Figure 4.22: Normalized values of TKE after the training phase: predicted (blue dots) vs training data (solid black line).

4.2.5.1 Forwarding algorithm and implementation in the solver

After the training, the set of weights and biases was stored into a graph. To perform run-time predictions during standard solver iterations, we were forced to embed the Python- based algorithm in a C^{++} environment. To do so, we relied on the C_{Api} library.

Figure 4.23 reports how each iteration of the solver is working.

- Once per simulation, wall distance is derived from the computational grid. It will be later used to derive y+ values;
- For every solid wall, we derive the internal field of the first

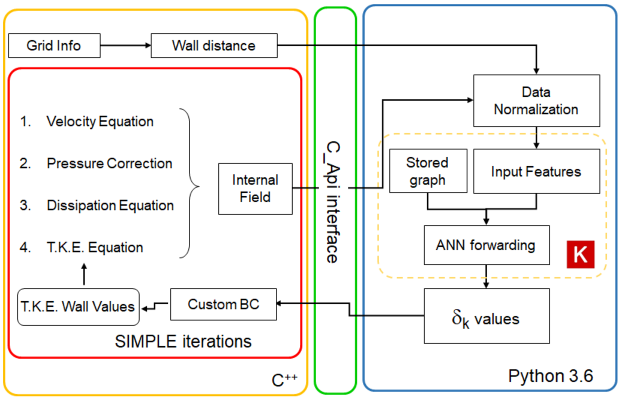


Figure 4.23: A machine-learning enhanced solver for OpenFOAM.

layer of cells. It is obtained by solving the standard equations of the k-ε model;

- Internal field is transformed into C_{Api} operators and is forwarded to the Python algorithm;
- Data is normalized and pre-processed to build input features;
- Stored graph is loaded, and input features are fed to the network, obtaining the values of δ_k for each of the cells;
- A custom boundary condition in OpenFOAM eventually corrects the values of TKE, derived from the set of equations of the turbulence models, with the output of the model [31];

4.2.6 Cross-validation Results

To test the generalization capability of the model, the machine-learnt wall treatment was applied to a simulation of the same case with an inferior Rossby number of 0.22. Given the definition of Rossby, the lower number was achieved through reducing inlet flow rate while leaving rotational velocity unchanged. We carried out

two RANS simulations with a different wall treatment: standard (kWF) and machine-learnt (ML). Standard formulation of kWF, namely kLowReWallFunction in OpenFoam, computes TKE values as an explicit form of y+ and constitutes the common practice for Low Reynolds grid [103]. An additional LES simulation gave us high-fidelity data for further comparisons, and it was not included in the training database. Computational time required to reach convergence remained the same between ML and kWF. The aim is to discover how the model is behaving out of the training data. Cross-validation is necessary to assess the generality of the model out of the training. We therefore report a zoom-in, from the general prediction of the whole computational domain down to the design goal of the algorithm.
Figure 4.24 shows the outcome of the model for the last iteration of the solver. The whole computational domain was fed to the algorithm. The model has good agreement with the true values, directly computed by the differences between the TKE in RANS and LES approaches. The normalization applied to δ_k, i.e. z-score, asses that the most frequent values are the ones closer to the zero of the axes, where cells have values equal to the mean of data. In so doing, we can observe that even in the cross-validation configuration, the model can statistically reproduce the behavior of δ_k, confirming the validity of the choice of the TKE delta as output feature. However, at the extremes of values (top right and down left of Figure4.24), model presents higher deviation from the true values. This issue can be easily ascribed to the fact that the model is predicting values in a part of the flow not originally present in the training database. It also suggests that the model may obtain a general validity, e.g. far from the wall, with an extended training.

By looking at a portion of the computational domain with y+¡400, we can observe how the model is behaving closer to solid walls. Figure 4.25 reports TKE values for the mid-height suction wall section between 0.2 and 0.5 meters from the inlet of the domain. LES and RANS simulations (Figures 4.25a and 4.25b respectively) differ especially in terms of magnitude. As expected, we also observed a tendency in RANS approach to overpredict the sep-

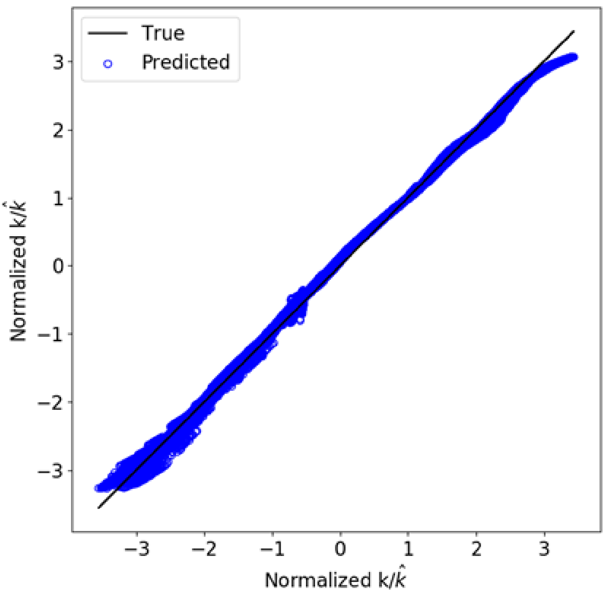


Figure 4.24: Model outcome for the whole computation domain of δ_k for the last time step of the cross-validation simulation.

aration over the suction side. Predicted δ_k (Figure 4.25c) versus the LES values (Figure 4.25d) reflects the trend seen in 4.24. We also highlight some discrepancy in Figure 4.25c that can be attributed to small errors in the predictor. In this visualization, the whole filtered computational domain was forwarded to the algorithm, obtaining a correction field from the solid wall to the outer layer.

As a final remark we provide the Relative Absolute Errors (RAE) for the three datasets. RAE is expressed for the j-th example by the ratio between the absolute of residuals, i.e. predicted minus true values, and the corresponding true value. Regarding that, a perfect predictor shows an averaged RAE equal to zero. Errors listed in Table 4.10 confirm that the model accuracy is drastically lower when the full domain is predicted. This is expected as the

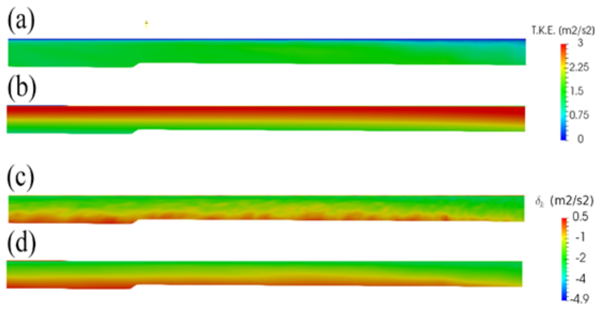


Figure 4.25: Averaged TKE for LES (a) and RANS simulations (b), predicted (c) and true δ_k (d)

model was not provided examples on the core of the flow. Filtered and First cells subdomains, however, show a similar behavior, with a maximum computed RAE of 0.092. Through a comparison with the standard approach we observe that, in this case, the use of a machine-learnt correction for the TKE may not lead to satisfactory results far from the walls, but this trend could be inverted with an aimed training. From the viscous sublayer to the outer layer ML provides a strong correction for k, with averaged RAEs far below the standard wall treatment.

Table 4.10: Relative Absolute Errors in Cross-Validation

	ML Treat.		Standard Treat.	
Subdomain	Aver. RAE	Max. RAE	Aver. RAE	Max. RAE
Full	0.073	0.171	0.094	0.211
Filtered	0.012	0.092	0.145	0.224
First cells	0.011	0.071	0.178	0.245

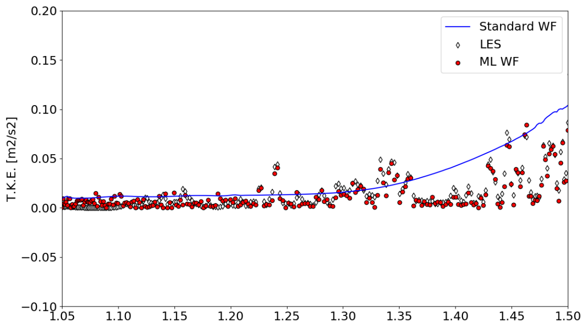


Figure 4.26: Comparison of the standard WF vs ML for the mid-height in the last iteration of the cross-validation case, between 0 and 0.6 meters from the inlet

4.2.7 Final Remarks

A machine-learnt wall treatment for TKE was derived. Training data was generated through high fidelity numerical simulation of rotating diffusers. These simulations were chosen as cases where traditional RANS approaches struggle to properly reproduce physics, especially with respect to the turbulent kinetic energy. A simplify geometry reduced the computational cost of the process. Despite the simplification, the effect of rotating forces was still preserved and marked, suggesting that the computational domain can be used in the future for extended studies. An analysis of the resolved grid scales and the pressure recovery coefficients provided us information on the reliability of the simulation.
We exploited data pre-processing to reduce computational costs of the algorithm training and to increase the accuracy of the model. The filtering operations that we performed on the dataset were eventually effective for a correct training of the model. This can possibly be used as a paradigm for all the models used for wall treatments. The choice of δ_k as a predictive target was investigated through frequency plots and correlations maps. In addition, this proved to be an effective way to assess that the training data is free

of outliers. Features were manipulated to assure Galilean invariancy. Results from the test data were extremely good, meaning that the learning was successfully achieved. That was also confirmed by looking the algorithm metrics as a function of the training epochs. The derived wall treatment was implemented in run-time computations of OpenFOAM v-18, as a custom boundary condition. The custom boundary was tested in a cross-validation case, namely a RANS numerical simulation of a similar geometry with an inferior Rossby number. It was compared to the standard wall treatment, using an additional LES simulation as a high-fidelity reference. We observed that the model was not effective in predicting δ_k far from the walls. This behaviour was expected as the model itself was not included in the original dataset training. Looking at the region of the fluid closer to walls, instead, our model achieved a higher accuracy than standard treatment. By investigating the δ_k field we observed some small predictive errors, but the RAE metric points out that this error is still below 1%. This behavior is also reflected in the first layer of cells, were the model showed similar metrics. This methodology can be easily extended to any geometry, as long as high-fidelity data is available. The first step is discovering where and how to correct physics. Once the correct elements are laid out, in term sof feature selections, algorithm, learning parameters and mathematical formulation, machine learning becomes a challenging alternative to standard modeling.

4.3 Identification of Poorly Ventilated Zones in Gas-Turbines Enclosures with Machine-learning

Ventilation systems are used in gas turbine packages to control the air temperature, to protect electrical instrumentation and auxiliary items installed inside the enclosure and to ensure a proper dilution of potentially dangerous gas leakages. These objectives are reached only if the ventilation flow is uniformly distributed in the whole volume of the package, providing a good air flow quality as

prescribed by international codes such as ISO 21789. To evaluate the effectiveness of the ventilation design, numerical computations are performed for several purposes, one of which is the identification of poorly ventilated portions of the enclosure. In fact, it is essential to accurately detect the regions which are less ventilated, since they could be prone to the accumulation of an accidental fuel gas leak. There are different approaches to identify these portions, such as decay regression or inlet source analysis, that require unsteady simulations of the flow field inside the package. The present work discusses the implementation of a new methodology using machine learning and artificial neural networks (ANN) to detect the poorly ventilated regions where a gas cloud can accumulate. The concentration of fuel gas is estimated starting from a steady-state computation without running a more expensive unsteady computation. The entire process is built around an accurate training of the ANN using a proper set of simpler test-cases that have been identified to match the characteristics of the gas turbine enclosure. During the training phase accuracy and overfitting of the ANN were monitored to ensure robustness of the method. The procedure is then applied to a real case scenario and the results are presented in this paper highlighting the main advantages of this approach respect to a conventional use of CFD analysis. Computations of the flow fields are carried out using OpenFOAM with RANS and U-RANS approaches, while the ANN is developed and trained in Python.

4.3.1 Introduction

The ventilation is an essential auxiliary system for gas turbines when they are installed inside an enclosure [113, 114, 115]. In fact, incorrect operations of ventilation inhibit the start of the gas turbine, while a loss of ventilation causes an emergency shut down, with clear consequences for the availability of the gas turbine train. The ventilation system is therefore essential, since it guarantees to cool down the instruments and equipment installed inside the package, that are used for the control of the engine. Moreover, a ventilation

flow is required to dilute the accidental fuel gas leaks potentially present inside an enclosed volume. Since inside a gas turbine package, instruments and other items are located everywhere and a potential leakage source from the fuel gas system can occur in a lot of possible locations, such as valves, flanges and piping, it is required to the ventilation flow to be uniformly distributed, e.g. ISO 21789 [116]. This requirement represents the main challenge during the design of a gas turbine package, because the geometry of the fluid domain is very complex and it is not so obvious that the flow entering in the enclosure from one or two inlets will be able to reach the whole volume [117]. This is the reason why, in the past, the design concept passed from a simple count of air volume exchange rate to a quality of the flow distribution [118, 119].
The verification of the effectiveness of the ventilation in terms of cool down and therefore of temperature reached by critical items, is easily detectable during tests using thermal strips or temperature sensors. On the other hand, an experimental acknowledgement of the influence of the ventilation on an unexpected fuel gas leakage is more complicated to perform. First of all, it would require to generate a credible sonic fuel gas leak inside the package without adopting invasive methods, then all the safety aspects should be handled and the more appropriate measurement approach should be selected. The mentioned matters make a computational analysis (CFD) the most suitable tool to be largely used for designing and verifying the effectiveness of the ventilation system. In this ambit, several methods could be considered to detect the poorly ventilated zones inside a package.
During the past years, BHGE developed, internally tested and validated several methods in to define the best compromise in terms of accuracy and computational time. In the present work, it is shown an innovative method developed taking advantage of the potentiality of the machine learning based on classification of temperature and flow patterns inside the enclosure.
In this paper a series of test cases are selected to study the purge of domains with simple geometries and peculiar flow characteristics (attached flow, geometry- and pressure-induced separation and so

on). From these cases the authors studied how the mass of methane dropped over time inside the domain and acquired information on the time evolution of concentration. These were used to identify the poorly ventilated zones of the domain that can be then correlated to the dilution level of any accidental gas leak and to provide a criterium to derive a domain criticality map. All the acquired data were used to train an artificial neural network (ANN) to define the effectiveness of the ventilation from velocity and temperature fields. Finally, after proper validation ANN was tested on a model of gas turbine enclosure. Finally, the purge of the same geometry was carried out to verify the ANN method.

4.3.2 Rationale and Selection of Test Cases

The rationale behind this project is the following: since computing the purge of a gas turbine enclosure is time consuming and requires a significant computational power, the authors want to train an ANN to classify the cells inside the package with a Poor Ventilation Index, starting from the averaged flow and temperature features obtained from a steady-state computation.
To do so, we need to train the ANN to recognize different flow and temperature patterns, providing a suitable number of examples related to the characteristic flow phenomena encountered in the enclosure. To this aim, the authors selected a series of test cases, mainly from available literature.
Test geometries include high Reynolds computations of plain channel flow (CF) [120], backward facing step (BFS) [121, 122] and 2D sinusoidal hill flow (2DH) at different Reynolds number [123, 124], axisymmetric hill flow (3DH) [125] and confined cylinder in cross-flow (CYL), [126]. This selection was driven by the necessity to explore different flow features such as attached flow, geometry-induced and pressure induced separation, three-dimensional shedding in simple geometries. Applying different Reynolds numbers allow to instruct the ANN to sort between different flow patterns that can occur in the same geometry and, as highlighted in the following, can result in a completely different purge dynamics of

the domain. Further information was collected by changing the turbulence intensity at the domain inflow.
As starting point, it was also decided to test this method to a simplified geometry to assess the capability of the approach.

4.3.3 Numerical Setup

Numerical computations were carried out using the open-source library OpenFOAM v1712. The compressible rhoSimpleFoam steady-state solver for turbulent flows was used to derive the average flow and temperature fields to be used in the training phase. Convective terms were computed using the QUICK scheme for velocity and turbulent quantities, gradients and laplacians were computed using the Gauss linear scheme.
The unsteady purge of the domains was then computed using the pressure, velocity, k and ε average fields previously computed as initial conditions, filling the volumes with methane and pushing fresh air at the inflow using the interIsoFoam solver for two-phase flows computations. Convective terms were computed using the QUICK scheme for velocity and turbulent quantities, gradients and laplacians were computed using the Gauss linear scheme. Time integration scheme was Crank-Nicholson with an adaptive timestep set to adjust the maximum CFL number to 1.0. During the computations we stored at runtime a series of data that included the total mass of methane still present in the volume, its center of mass, and the local value of t_D, here defined as the total amount of time for which the methane concentration of the cell was between lower and upper flammability levels (LFL and UFL). Linearized equations were solved using a conjugate gradient solver with DIC preconditioner on pressure and DILU preconditioner on other equations. Convergence threshold was set to 10-5 for pressure and 10-7 for other equations. In both steady-state and purge analysis, turbulence closure relied on the standard $k-\varepsilon$ model of Launder with adaptive wall functions that allowed to switch between low- and high- Reynolds approach. A summary of grid density and final y^+ for each case is given in

Table 4.11.

Table 4.11: Summary of test cases

case	Re	inlet TI	cell count	average y^+
CF	50500	N.A.	300x128x4	1
BFS1	5100	6%	401x97x6	1
BFS2	5100	10%	401x97x6	1
BFS3	5100	6%	401x97x6	1
BFS4	132000	6%	401x98x5	7
2DH1	10595	N.A.	300x65x6	1
2DH2	37000	N.A.	160x100x60	2
3DH	6500	5%	138x30x79	8
CYL	140000	5%	278x128x20	11

4.3.4 CFD Results

4.3.4.1 Steady-state computations

Results for each test case were validated against available data from referenced papers. Here we limit the discussion of validation to BFS1 and 2DH1 cases as examples of the capability of the approach to reproduce the velocity field within the accuracy of the RANS model.
In Figure 4.27 wall-to-wall velocity profiles at different streamwise positions are shown for BFS1 giving a direct comparison with DNS data. As it is well known, the $k - \varepsilon$ model slightly overpredicts the length of recirculation after the step, but in general the result is accurate with an acceptable approximation of the flow characteristics for the industrial purposes.

Entirely similar profiles are obtained with 2DH1 case, as shown in Figure **??**. In fact, the RANS modelling again slightly overpredicts the recirculation length respect to DNS, but again this is the expected level of approximation of this model.

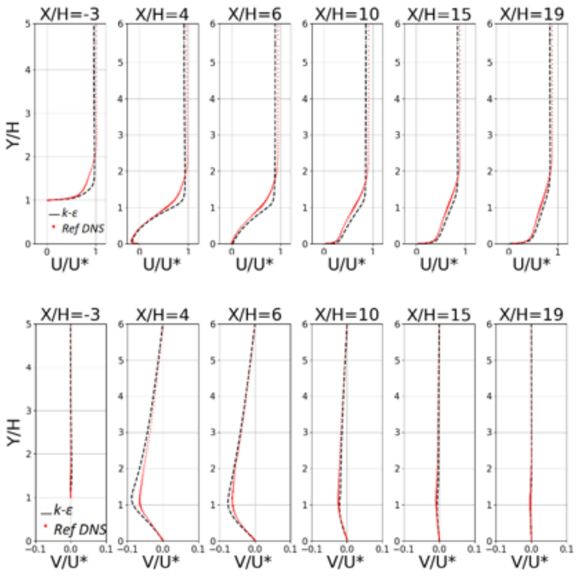


Figure 4.27: Streamwise (top) and wall-to-wall (bottom) velocity component profiles for case BFS1. Red: reference DNS, black: current results from $k-\varepsilon$ computations.

4.3.4.2 Two-phase transient computations

The average flow field computed with the steady-state approach was used as initial condition for the purge analysis by filling the volume with methane and introducing fresh air at the inlet of the domain. Computations were carried out until the volume was filled with air or, in the cases where a complete gas purge was not possible, when the amount of methane in the domain reached a constant value.
As expected, for channel flow (CF) there is not much evolution inside the volume as methane is pushed out by the incoming air. However, as seen in Figure 4.29, after a flow through there is still about 17% of the original methane inside the volume, mainly trapped in the developed boundary layers. A little more than 2

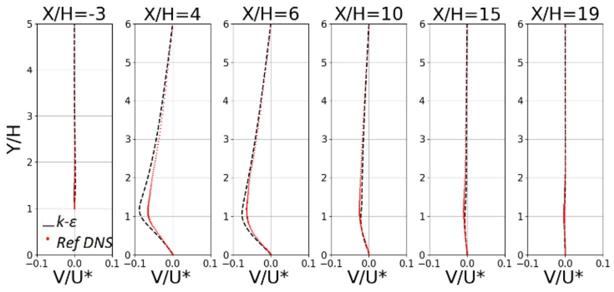


Figure 4.28: Streamwise (top) and wall-to-wall (bottom) velocity component profiles for case 2DH1. Red: reference DNS, black: current results from $k-\varepsilon$ computations.

flow-throughs are necessary to completely clean the environment from the gas.

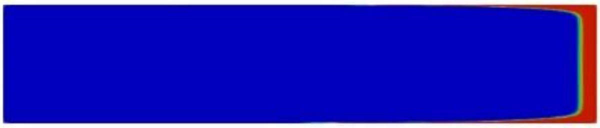


Figure 4.29: Methane concentration after one flow through time.

When dealing with the backward facing step (BFS), the different geometrical configurations and boundary conditions (see Table 4.11) determine different purge histories as reported in Figure 4.31. BFS1 and BFS2 differ from BFS3 and BFS4 for the different height of the step, which is respectively one fifth and one third of the total channel height.

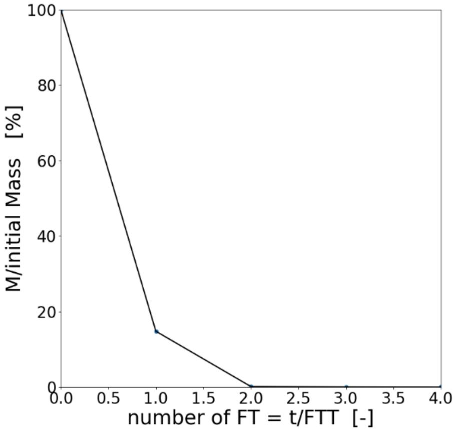


Figure 4.30: Discharge of methane in channel flow.

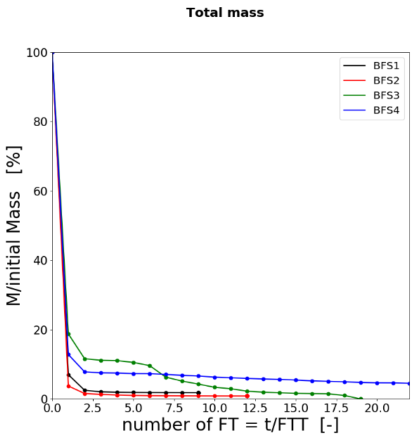


Figure 4.31: Discharge of methane in BFS cases.

Here, two different characteristics need to be highlighted. The first one is the sudden initial drop of methane concentration, that occurs over the same amount of time, corresponding to 1 flow passage for all the four configurations. The second one is that the three cases for which a considerable amount of methane cannot be expelled (corresponding to BFS1-2-4) an almost asymptotic behavior is reached after 2 flow-throughs. In Figure 4.32 snapshots of the concentration of methane in the BFS cases are given, to visualize the entrapped mass of gas still present after the air purge. The difference between BFS1 and BFS2 is only due to the influence of inlet turbulence intensity, while for BFS4 it is evident the effect of the Reynolds number. However, the main effect seems to be related to the aspect ratio of the domain as the only case for which total purge is possible is that of BFS3.

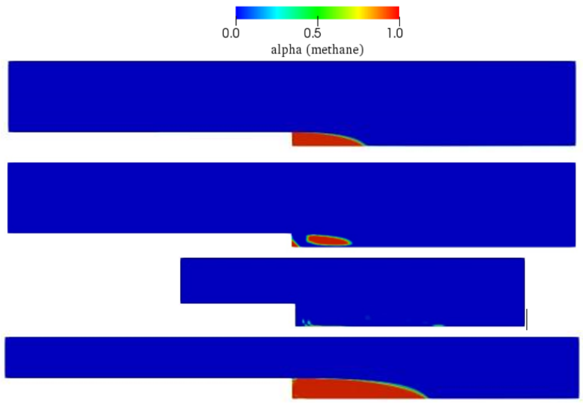


Figure 4.32: Asymptotic distribution of methane for BFS1, BFS2, BFS3 and BFS4 respectively from the top to the bottom.

The purge of the two-dimensional hill (2DH) cases confirms that the washing mechanism is strongly dependent on the Reynolds number. Figure 4.33 shows the purge for 2DH cases and the concentration trends clearly indicate that the methane cannot be completely washed out from the domain for 2DH1, while this is possible for 2DH2.

In Figure 4.34 snapshots of the concentration of methane for

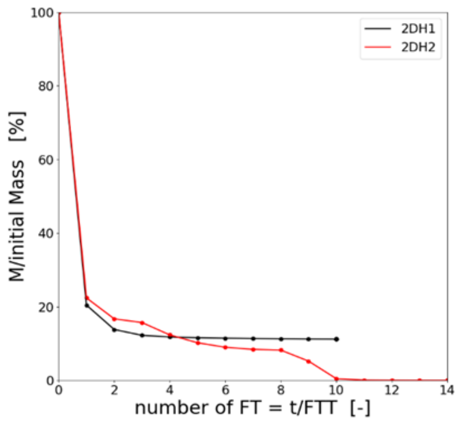


Figure 4.33: Discharge of methane in 2DH1 and 2DH2 cases.

2DH1 and 2DH2 domains are shown after 14 flow-throughs: for the first case the recirculating flow entraps a bubble of methane that is not washed away, while at higher Reynolds number there is a full purge. The interesting part of this mechanism is that the recirculating flow engulfs the methane and shield it from direct exposure to the hot surface below.

Referring to the three dimensional hill case (3DH) and the cylinder in cross flow (CYL), it is possible to note from Figure 4.35 that 1 flow through time (FTT) is sufficient to purge most of the domain for both cases, while a second flow through time is needed to achieve an asymptotic behavior.
The dynamics of purge for 3DH case is shown in Figure 4.36, where the concentration of methane is shown after 1, 2 and 5 flow-throughs times in 3 different planes corresponding to midspan, z/h=1 and z/h=2. Away from the hill, the only methane that gets trapped is inside the boundary layers on the endwalls.

The wake of the obstacle, however, releases bubbles of methane with flammable concentration and in the aft section of the hill a small amount of methane is present even after 5 flow throughs.Time

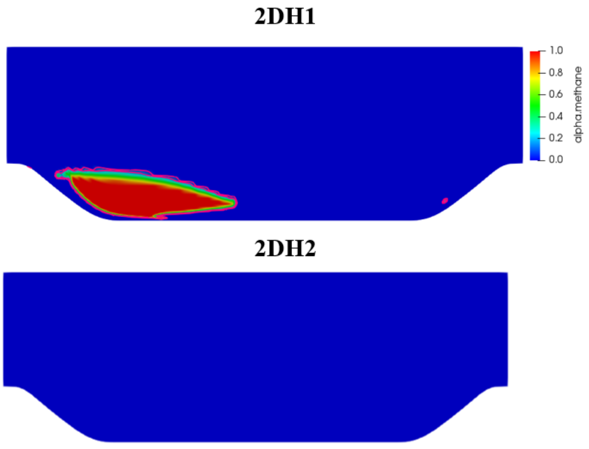


Figure 4.34: Asymptotic distribution of methane for 2DH1 (top) and 2DH2 (bottom).

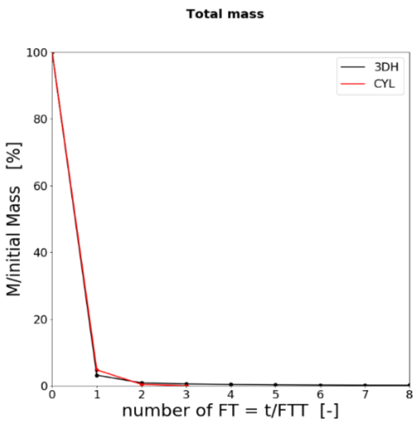


Figure 4.35: Discharge of methane in 3DH and CYL cases.

history of the gas purge and their close similarities give a fairly good and useful insight, but also helps in drawing some conclusions on the poorly ventilated zones and relative hazardous considerations. From this viewpoint, it has to be also considered that a real ignitable condition would be present only with a proper concentration of methane (between LFL and UFL). Therefore, looking at the evolution of the gas concentration over time we can visualize ignitable fringes of methane while the purge is going on and inevitably those spots are positioned in boundary layers and recirculating or low speed regions. So, going back to the discharge histories, we can conclude that the ignitable portions of the domain are generally less extended during the period between 0 and 1 flow-throughs time than those between 1 and 2, which in turn are less extended than those after 2 flow-throughs.

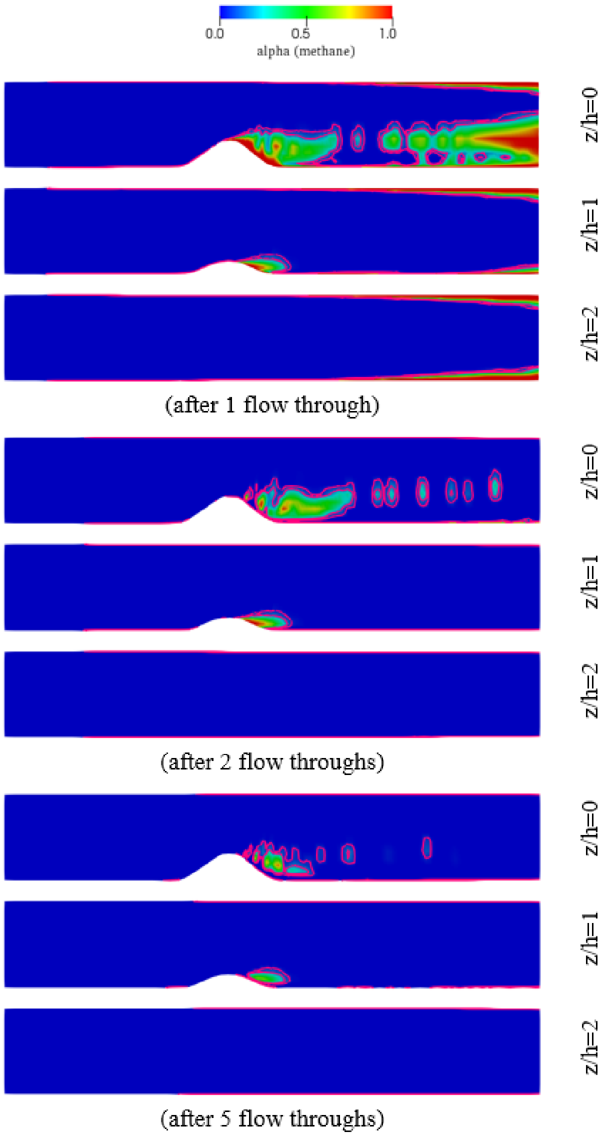


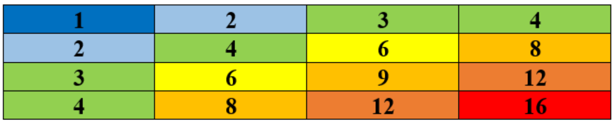
Figure 4.36: Discharge of methane in case 3DH, after 1 FT (top), 2 FT (center) and 5 FT (bottom) at different spanwise immersions.

4.3.5 Poor Ventilation Index map

The screening analysis of the poorly ventilated zones inside the gas turbine enclosure is an essential step of the entire assessment of the ventilation system effectiveness required by ISO 21789 [116].
There are several approaches adopted in other fields, such as for civil building applications, where the quality of ventilation is evaluated by a parameter defined as the mean air age, that is proportional to the residence time of the air particles inside the domain to be studied. An older age corresponds to a lower quality of the flow and so to a poorly ventilated zone. For gas turbine package applications, this approach is not completely suitable, since the velocity field is only one factor of the complete analysis to be performed. In fact the really important poorly ventilated zones of a gas turbine enclosure are only those where a fuel gas accumulation may occur, with a concentration of methane within a specific range and where sufficiently high temperature is present. To consider simultaneously these aspects, the Authors synthetized them into a thresholds matrix, which allows to classify the different zone of a CFD domain through a Poor Ventilation Index (PVI) map.
To build this map, it was required to identify levels of fuel gas concentration (L_α) based on the value of the discharge time (t_D). Then, a second level related to different thresholds of absolute temperature (L_T) was defined according to the Auto Ignition Temperature (AIT) of the fuel gas. For both parameters, four levels were considered with a progressive integer value from 1 to 4 proportional to the severity of danger. Combining these levels (L_α and L_T) with a specific rule (Figure 4.8) for each cell of the domain, it is possible to define a scale of Poor Ventilation Index (PVI) map, from a minimum of 1 to a maximum of 16, Figure 4.37. This scale is directly dependent on the fuel gas concentration history and temperature field, while it is implicitly dependent on the velocity, which influences the fields of these two parameters. Finally, each level is multiplied by a coefficient varying from 0 to 1, which is dependent to the particular application case.

$$\text{PVI} = \beta_\alpha L_\alpha \cdot \beta_T L_T \tag{4.8}$$

Applying this method to the simulation results of the two-dimensional hill (2DH1 and 2DH2) case and adopting both β coefficients equal to 1, nine different possible map values are achieved between 1 and 16, according to Figure 4.37. Figures 4.38 and reffig:gl14 show the fields of these levels (L_α and L_T) and the resultant PVI map.



1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16

Figure 4.37: PVI Levels.

At lower Reynolds number, Figure 4.38, total purge of the domain is not possible, and therefore there is a portion of recirculating flow with values of L_α equal to 3 and 4. However, the heat removal mechanism is quite efficient and the value of LT is always equal to 1, leading to a final PVI that does not exceed 4.
At higher Reynolds number, Figure 4.39, the overall ventilation system performs better, as the domain is purged from methane completely in a short time leading to a L_α equal to 1 everywhere. Also the heat removal is efficient and the only region where LT reaches a level of 2 is in the reattachment region of the separation bubble. When both factors are combined into the PVI the index does not go above 2, and we can conclude that this situation is even better than that of low-Reynolds number and both lead to satisfactory results in terms of PVI.

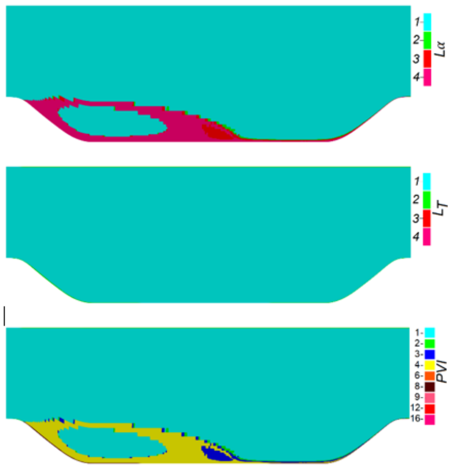


Figure 4.38: L_α and L_T (center) and PV map (bottom) for 2DH1 case.

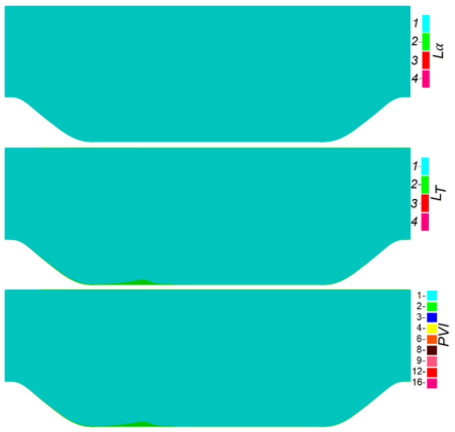


Figure 4.39: L_α and L_T (center) and PV map (bottom) for 2DH2 case.

4.3.6 Machine Learning Module

Our final aim is to build and test an artificial neural network (ANN) able to build a poor ventilation (PVI) map directly from steady state solutions of the flow and temperature fields, as in a real-case scenario of a gas turbine enclosure the computational requirements of an unsteady purge analysis would be quite long due to the mesh size. To this aim we built and trained an ANN to derive the PVI map from steady-state flow and temperature features. Training data were derived from the aforementioned test cases, with a total of about 5M training samples, that were pre-processed using a Z-score normalization.
Here, a single-label multiclass classification problem was solved, as every cell has a univocally-associated level. As input features, velocity and temperature fields were provided to the ANN, for a total of 4 input features. The algorithm was implemented in Python using the *sklearn library*. ANN morphology was optimized to obtain the maximum predictive accuracy and the best training convergence. The final structure of the ANN has 6 layers with 30 neurons, each activated by a RELU function; the last layer uses the softmax activation function. In this way, the network computes a probability distribution over the 9 levels for each. The highest level was considered as the final state of the cell. Cross-entropy cost function was optimized by an adaptive gradient descent optimizer, automatically assuring a learning rate decay during each raining epoch. Batch feeding, with each batch equal to one fourth of the total data, was exploited to grant a faster convergence. After 2000 training epochs, algorithm reached full convergence, with an overall accuracy of 93.2%. Initial database of training examples was split to obtain a 30% of test examples. training epochs. After the algorithm training, test examples were used as input and they provided a negative check on possible overfitting of the predictor. The ANN was validated against a model geometry for the gas enclosure.

4.3.7 Test Model Case

The test model geometry was derived from available literature, starting from [127]. After validation of results with comparison of available velocity profiles, the model was scaled to a size comparable to that of an enclosure and the air velocity increased to that of a normal ventilation system. To best resemble the proper layout of a gas turbine enclosure, the inlet and outlet of the domain were extended. The final geometry is shown in Figure 4.40, while in Figure 4.41 a sketch with different reference planes is shown, corresponding to the planes that will be used for the following discussion.

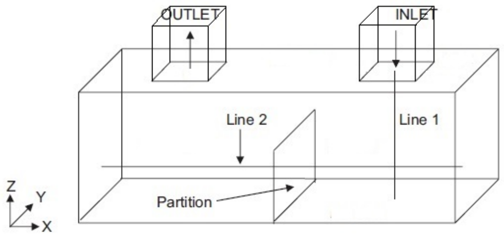


Figure 4.40: Isometric view of the domain geometry.

Then, a RANS computation was performed to generate the average velocity and temperature fields to be fed to the ANN. These are shown in Figure 4.42 and Figure 4.43. The most important flow features that characterize the motion inside the test case are:

- the high speed region corresponding to the inlet, that results in a fast jet impinging on the bottom (heated wall);
- the separation due to the presence of the partition wall, that separates the inlet and outlet regions;
- the low-speed region below the outlet chimney, due to the separated flow behind the partition wall.

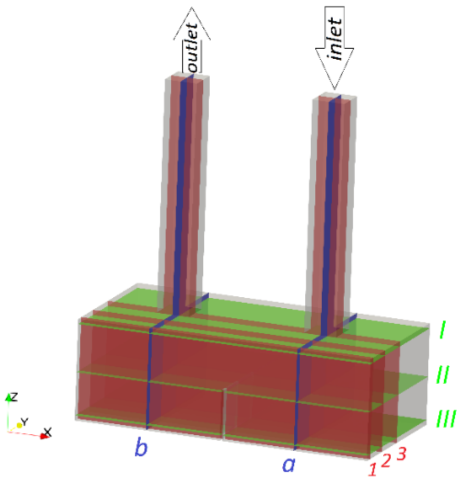


Figure 4.41: Sketch of the computational domain with 8 reference planes (a,b,1,2,3,I,II,III) used to show the numerical results.

These reflect in the temperature distribution, Figure 4.43, and in particular:

- the inlet jet with fresh air is characterized by low temperature and is able to remove heat from the bottom endwall where it impinges;
- comparing the half of the domain below the inlet with that behind the partition wall, it is apparent that the first is cooled down by the inlet jet, while the latter is characterized by higher temperature due to the slower heat removal in the stagnant flow region.

Once the velocity and temperature data are given as input to the trained ANN, the PVI map shown in Figure 4.45 was computed. Here we recognize that the chamber below the outlet pipe shows higher levels of PVI, due to the combining factor of recirculation and high temperature, while in the chamber below the outlet the PVI is

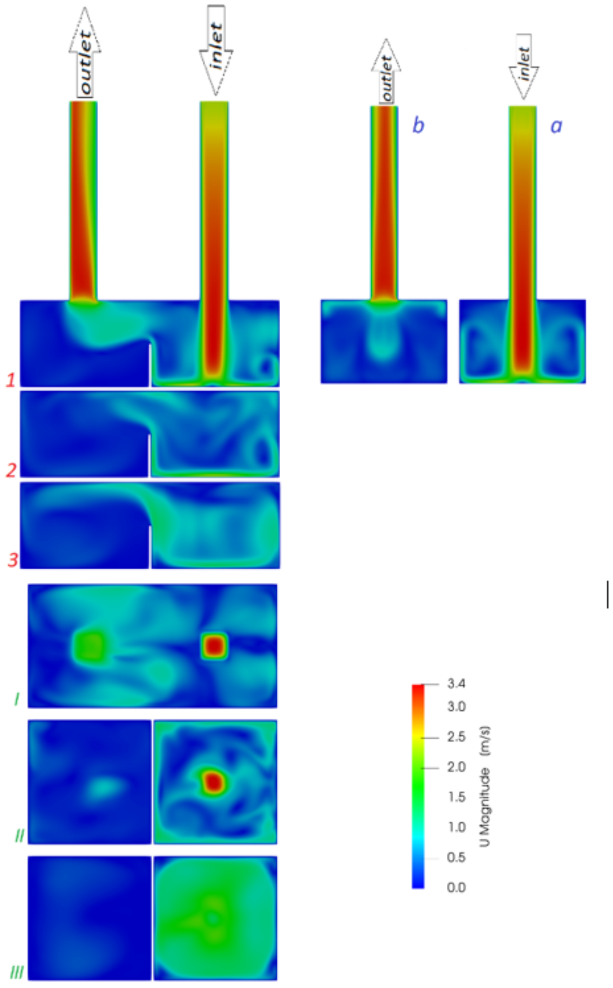


Figure 4.42: Velocity magnitude field for the enclosure model case.

high only near the bottom (heated) endwall where the expansion of the air jet coming from the inlet is blocked by the vertical walls.

The more complex and threedimensional flow features results, with respect to the previous cases, to higher PVI levels as a combination of all the mentioned factors.

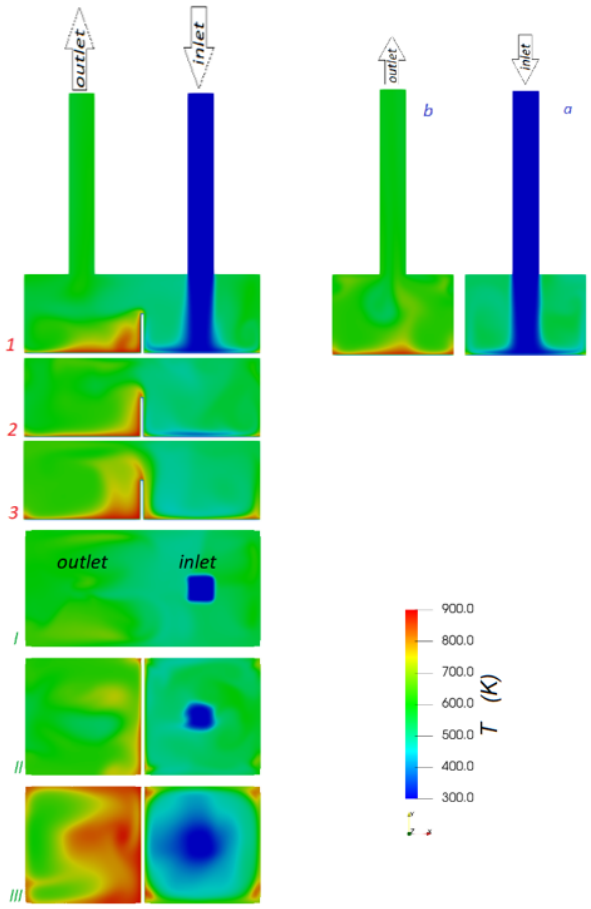


Figure 4.43: Temperature field for the enclosure model case.

Later, the authors run an unsteady computation with the two-

phase solver to study the purge of the enclosure model with the standard strategy. A first important finding, Figure 4.44, is that the discharge history still follows the same trend that was found for the previous cases, with a sharp drop over the 1st flow through time, a second drop over the 2nd and an asymptotic trend after that.
Concentration of methane and air after 1 and 2 flow-throughs are shown in Figure 4.45. Here it is interesting to notice that after the 1st flow through the region below the inlet jet appears already well washed, with only limited amount of methane distributed in large bubbles. However, these are surrounded by dangerous concentration and therefore poses a risk of flammability. After 2 flow-throughs the same zone has only limited amount of methane left, while the chamber below the outlet ventilation pipe shows a trend similar to the inlet chamber after 1 flow through.
From the temperature map, Figure 4.43, and the map, Figure 4.47, it was possible to evaluate the L_T and L_α values and therefore to build a danger map similar to that of Figure 4.33. A one-to-one comparison lead to an accuracy of 91.3% of the ANN solution.

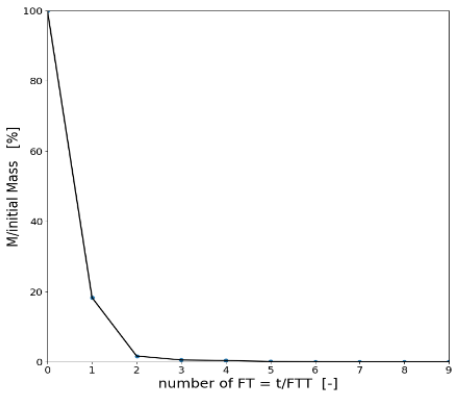


Figure 4.44: Discharge of methane in enclosure model case.

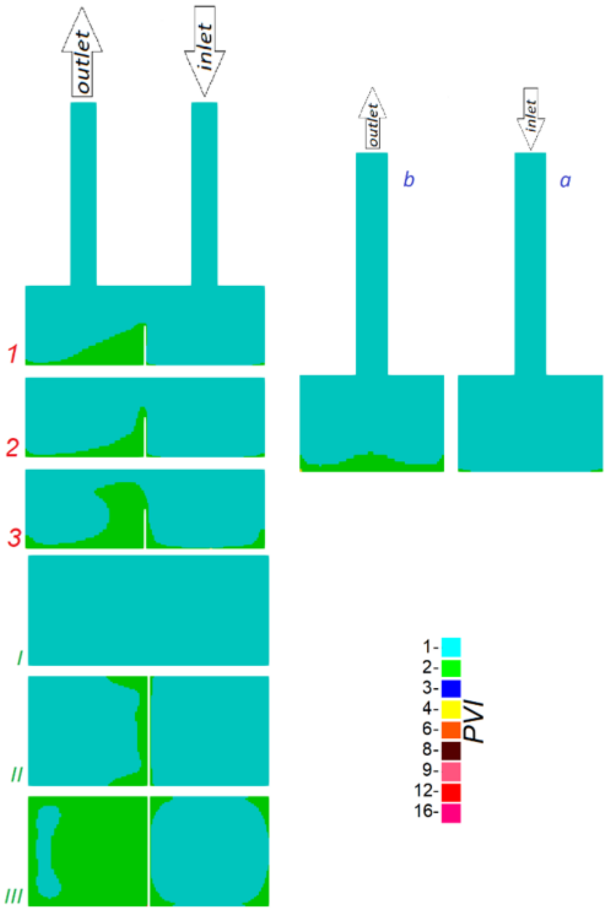


Figure 4.45: PVI map for enclosure model case.

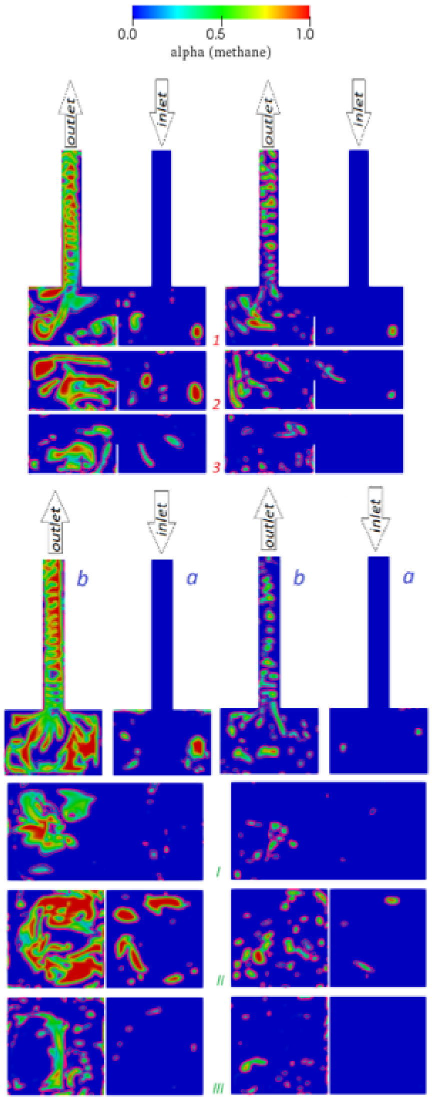


Figure 4.46: Concentration of methane for the test case after 1 (left) and 2 (right) flow through times.

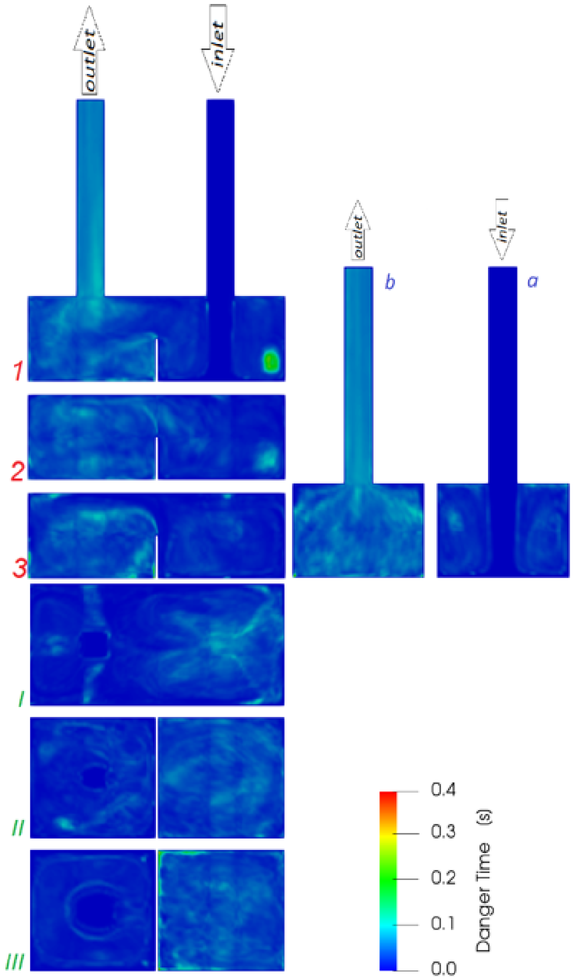


Figure 4.47: Discharge time (t_D) distribution for the test case.

4.3.8 Conclusions

To develop a new methodology for the definition of the poorly ventilation zones in gas turbine enclosure, the authors studied the purge of methane from a series of test cases, that were selected to represent different flow phenomena typical of the enclosure. From the analysis of these purges a typical evolution of the mass of methane was detected inside the enclosure that can be related directly to the amount of fresh air fed to the domain. From this behavior we derived a level of criticality associated to the time spent by each cell of the domain with an ignitable concentration of methane. Similarly, a level of criticality was associated to the temperature field with levels related to the auto-ignition temperature of the fuel gas. A Poor Ventilation map of the domain was then derived as a function of these two indexes.
Later an artificial neural network was designed to classify the level of poor ventilation from the velocity and temperature field. The network was optimized, trained and tested over the data from the test cases. After it was validated on a test model and demonstrated to be able to correctly predict the poorly ventilated zones inside the domain. The major takeaway of this work is that, once the ANN was trained and tested, it is possible to avoid an unsteady purge analysis of the enclosure. This means that the information that was derived by this time-consuming computation, lasting several days, can be extrapolated in post-processing from the steady state solution in a couple of minutes.

4.3.9 Acknowledgments

We thank Dr. Lucherini, Dr. Minotti and Dr. Rossin from BHGE, a GE Company, for the support and funding of the project.

Chapter 5

Conclusions

This work has treated several cutting-edge and crucial aspects of turbulent modeling and turbomachinery. The final aim was to prove how data-driven tools can be extremely useful in CFD. Industrial CFD, and RANS approach in particular, is the workhorse of optimization and validation of turbomachinery design. Although the efforts put in developing faster and more efficient codes, higher fidelity approaches (namely DES or LES) still present a significant cost for the majority of application. Three needs must be satisfied:

- *Generality of model;*
- *Robustness of model;*
- *Speed of computations;*

So far, no existing RANS model excels in each of the three points. Despite that, research on the topic has reached a stagnation point, and no major breakthrough can be found in the past decades. Machine-learning come in aid, showing the ability to overcome some of the deficiencies of the models. As a new approach, however, it is still an open debate how and where to apply an artificial intelligence. In the work we focused on the $k - \varepsilon$ turbulence model, the most popular among all the two-equations models.

The first part of the work reports an analysis of the majority of the works already published on machine-learning enhanced turbulence modeling. Statistical inference from huge dataset is not a novel topic in turbulence modeling. All the models currently available, in fact, were derived from physics and eventually tuned to fit experimental observations. A similar approach has been followed in the past years through machine learning. To the best of the author's knowledge, the classification here reported is the first attempt to review and organize the different works published on the topic so far. We opted for a three-categories classification, based on the methodology.

- *Field Inversing* - a part of the production or dissipation term in the T.K.E. equation is incorrect and is addressed as source of error in RANS simulation. A correction for these term is derived and exploited to correct two-equation models;

- *Anisotropic Modeling* - discrepancy between RANS and higher fidelity simulation is attributed to the isotropic turbulence assumption. Turbulence invariance as input feature grants independence from the frame of reference.

- *Derivative Application* - few works have tried different approaches to the problem. Methodology is usually a mix between the previous two.

We also report some key points that were discovered from literature analysis:

- There is no a dominant methodology, with many approaches only superficially investigated;

- Non-linear algorithms, e.g. *neural networks, random forest regressors, ensemble models,* share behavior and accuracy;

- Multi-layer perceptrons and random forests have the most popularity within machine-learning algorithms, followed closely by genetic programming;

- Achieving frame of reference independence is a non-trivial task, and the only viable solution proposed so far is the use of turbulence invariance;
- All the methodologies have been applied to relatively simple configurations, the feasibility of machine learning in complex flows has not been fully investigated;
- Some authors claim that run-time computation of the machine-learning enhanced fields may lead to a worsening of the RANS approach, in contrast with other works;
- All the works agree that data-preprocessing and Exploratory Data Analysis are fundamental to assure the statistical significance of data and to maximize model effectiveness.

In the second part, a full mathematical representation of neural networks is provided. Multi-layer perceptrons neural network have been identified as our master algorithm for turbomachinery applications. The choice was justified by several factors:

- *Non-linearity*, as the phenomena of interest cannot be linearized;
- *Overfitting monitoring*, to improve the generalization capability of the models;
- *Fast implementation*;
- *Ability to handle sparse data*.

In the third part, neural network were directly applied to turbulence modeling, in three different cases: two *wall treatments* and a *multi-phase flow*. Standard two-equation models fail in presence of adverse gradient of pressure and recirculating flows. For example, in a 2D periodic hill, standard RANS approaches overpredict the reattachment length. That can lead to strong error in predicting real flows.
In the first part of the work we created a database through LES simulations of flows that share similar behaviors. The series of

filtering operation to reduce redundant data and fasten training time proved to be effective. A multi-layer perceptron was trained to map local features to the T.K.E. values of the first cells. We created an architecture for machine-learning enhanced computations of Launder-Sharma model in OpenFoam v18.

- C_{API} library perfectly handle data flow between the two environments;
- Computational time remains the same;
- It is necessary to create a custom boundary condition to run-time update T.K.E. values.

The first wall-function was than tested on two different cases. The 2D periodic hill was included in the original dataset. We adopted three different grid resolution. ANN-treatment proved to be more effective than the standard treatment in predicting reattachment length. However, no treatment obtained satisfying results with the coarsest grid. Thus we concluded that:

- The algorithm is mostly insensible to grid refinement and it can work with different grids;
- Requirements in term of grid resolution are identical for the two approaches.

The second application was a NACA profile and its modified version with leading edge bumps, both in a cascade arrangement. Thus, we applied the first wall function on a cross-validation case, not included in the original database. It was done to stress the framework and test the generalization capability of the model. We drew two important conclusions:

- In the simulation of the base profile we obviously observed a two-dimensional flow. Results from the standard and ANN treatment could be superimposed and were coincident. Investigation on the full flow fields did not highlight significant differences. Its importance is high as the ANN was not worsening the base model.

- The simulation of the modified profile was instead fully three-dimensional. In this case a better prediction of the T.K.E. values led to a more accurate estimation of the pressure coefficient on the airfoil. It proved the generalization capability and therefore the feasibility of the approach.

The second wall treatment was developed for rotating flows. In fact, turbomachinery flows can be roughly approximated as rotating flows in forced passages. As in the previous work, our algorithm acted on the T.K.E. values near the walls, with a correction term δ_k. Results from CFD showed that the $k - \varepsilon$ model was insensible to the effect of rotation which was in contrast with experimental results. A deep analysis of data pre-processing and treatment has been reported.

- We could achieve independence from from the frame of reference without exploiting turbulent invariants by using scalar fields only;

- In this application, the filtering operations had a positive impact on the model. That suggests that filtering can be further extended to future works;

- Exploratory Data Analysis points out that flow fields do not show strong mutual interaction and no additional feature manipulation/creation is required;

- A mix of local normalization and zScore leveled the scales of magnitude of all the features;

- Advanced techniques, such as modifying the cost function, batch feeding and cross-validation overcame common issue in ANN training.

Training and testing phases of the algorithm were satisfactory and the reached convergence. The model was eventually used in run-time computation in the same framework of the recirculating flows. An in-depth analysis of the flow fields showed that:

- In spite of the zonal training, the model predictive capability was excellent in the whole domain. This suggest that, with additional training and a modified mathematical formulation, it may be possible extension of the methodology to the full domain;

- In the inner layer, the effect of a better prediction of T.K.E. values is evident, with rotating forces that affect the boundary layers.

In the final part of the work, we developed through ANN a model to automatically detected and classified dangerous zones in Gas-Turbine enclosure. To do so, we first trained the algorithm on canonical flows that share common features with the full domain, such as attached flow, geometry-induced and pressure induced separation, three-dimensional shedding. We eventually verified the effectiveness of model in a simplified configuration of the GT enclosure. Further comments on the work are reported below.

- By analyzing the center of mass of methane, we discover a common pattern of during the purge of the phase;

- Machine-learning was capable of reconstructing a full-three dimensional flow as a superimposition of simpler configuration, drastically reducing the complexity of the problem;

- Machine-learning was able to handle transport of multi-phase flows;

- Despite being desirable, high-fidelity data are not always necessary. In fact, we treated the problem with data as accurate as the one required by the final application, *i.e. data from RANS simulations*;

The three cases pointed out that machine-learning enhanced turbulence modeling is not only possible, but extremely effective in the turbomachinery field. The development may be slow, given the relatively novelty of techniques and instruments, but the growing community of researchers is relentless searching for new solution and further advances. We surely find ourselves in the big data era of turbulence modeling.

Acknowledgments

First of all, I would like to express my sincere gratitude to my supervisor Prof. Alessandro Corsini, for his guidance and trust during my Ph.D. studies. It has been an honor to be his student.

Endless thanks go also to my co-supervisor Prof. Giovanni Delibra, for everything he taught me.

A special thank goes to Prof. Johan Van Der Spuy, for his insightful comments and encouragement in the past months.

Last but not least, I want to thank my workmates for all the fun we had in the past three years.

Un ringraziamento alla mia famiglia, per l'incessante supporto durante tutti questi anni.

La mia più profonda gratitutine a Renato, Francesca, Marco, Spino, Giuseppe, Giorgio, Federico, Irene, Edoardo, Michele, Luca, Francesco, Lorenzo, Anna, Giulia, Marta, Riccardo, Petty. Senza di voi questo percorso sarebbe stato decisamente più difficile e meno divertente.

Infine, un grazie speciale a Gino, David, Tommaso, Valerio, Fabrizio, Paolo, Graziano e Lucio, amici e colleghi con i quali ho avuto l'onore di condividere quotidianamente gioie e disavventure.

Bibliography

[1] J P Slotnick et al. *CFD Vision 2030 Study: A Path to Revolutionary Computational Aerosciences*. 2014.

[2] H. Karthik Duraisamy. *A Framework for Turbulence Modeling using Big Data*. 2017.

[3] J. Moore. "A Wake and an Eddy in a Rotating, Radial-Flow Passage—Part 1: Experimental Observations". In: *Journal of Engineering for Power* 95.3 (July 1973), pp. 205–212. ISSN: 0022-0825. DOI: 10.1115/1.3445724.

[4] Stephen B Pope. *Turbulent flows*. Cambridge: Cambridge Univ. Press, 2011.

[5] P. Davidson. *Turbulence: An Introduction for Scientists and Engineers*. Oxford University Press, 2015. ISBN: 9780198722595.

[6] M Nallasamy. "Turbulence models and their applications to the prediction of internal flows: a review". In: *Computers & Fluids* 15.2 (1987), pp. 151–194.

[7] Y.-H Ho and B Lakshminarayana. "Computation of Three-Dimensional Steady and Unsteady Flow Through a Compressor Stage". In: June 1996, V001T01A026. DOI: 10.1115/96-GT-070.

[8] P Bradshaw. "Variations on a theme of Prandtl(Complex turbulent flows as perturbations of classical thin shear layers and application of Prandtl approximation)". In: *AGARD Turbulent Shear Flows 10 p(SEE N 72-20273 11-12)* (1972).

[9] J. P. Johnston. "Internal Flows". In: *Turbulence*. Ed. by Peter Bradshaw. Berlin, Heidelberg: Springer Berlin Heidelberg, 1976, pp. 109–169. ISBN: 978-3-662-22568-4. DOI: 10.1007/978-3-662-22568-4_3.

[10] E.M. Greitzer, C.S. Tan, and M.B. Graf. *Internal Flow: Concepts and Applications*. Cambridge Engine Technology Series. Cambridge University Press, 2007. ISBN: 9781139451116.

[11] J. D. Denton. "Loss Mechanisms in Turbomachines". In: May 1993, V002T14A001. DOI: 10.1115/93-GT-435.

[12] Gregory A Tokaty. *A history and philosophy of fluid mechanics*. Courier Corporation, 1994.

[13] RJ. Simoneau et al. *Turbomachinery*. Aircraft Propulsion and Power. NASA United States, 1987.

[14] C. Tropea, A. Yarin, and J. Foss. *Springer Handbook of Experimental Fluid Mechanics*. Jan. 2007. ISBN: 9783540251415. DOI: 10.1007/978-3-540-30299-5.

[15] Tommaso Bonanni et al. "Modelling of Axial Fan and Anti-Stall Ring on a Virtual Test Rig for Air Performance Evaluation". In: June 2016, V001T09A005. DOI: 10.1115/GT2016-56862.

[16] Alessandro Corsini et al. "A CFD-based Virtual Test-rig for Rotating Heat Exchangers". In: *Energy Procedia* 82 (2015). 70th Conference of the Italian Thermal Machines Engineering Association, ATI2015, pp. 245 –251. ISSN: 1876-6102. DOI: https://doi.org/10.1016/j.egypro.2015.12.029.

[17] MV Casey. "The industrial use of CFD in the design of turbomachinery". In: *In AGARD, Turbomachinery Design Using CFD 24 p (SEE N95-14127 03-34)*. 1994.

[18] MV Casey. "Computational methods for preliminary design and geometry definition in turbomachinery". In: *In AGARD, Turbomachinery Design Using CFD 22 p (SEE N95-14127 03-34)*. 1994.

[19] J. D. Denton. "Some limitations of turbomachinery CFD." In: June 2010. DOI: ASMEGT2010-˜22540.

[20] Douglas L Dwoyer, M Yousuff Hussaini, and Robert G Voigt. *Theoretical approaches to turbulence*. Vol. 58. Springer Science & Business Media, 2012.

[21] A. Corsini. "Modeling (Understanding and Controlling) Turbulent Flows: the Heritage of Leonardo da Vinci in Modern Computational Fluid Dynamics". In: Sept. 2018.

[22] L. Daston and E. Lunbeck. *Histories of Scientific Observation*. University of Chicago Press, 2011. ISBN: 9780226136783.

[23] Xindong Wu et al. "Data mining with big data". In: *IEEE transactions on knowledge and data engineering* 26.1 (2013), pp. 97–107.

[24] Huang Fang. "Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem". In: *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*. IEEE. 2015, pp. 820–824.

[25] G.E.P. Box and N.R. Draper. *Empirical model-building and response surfaces*. Wiley series in probability and mathematical statistics: Applied probability and statistics. Wiley, 1987. ISBN: 9780471810339.

[26] R.L. Launer and G.N. Wilkinson. *Robustness in Statistics*. Academic Press Rapid Manuscript Reproduction. Elsevier Science, 2014. ISBN: 9781483263366.

[27] Jeffrey Slotnick et al. "Enabling the environmentally clean air transportation of the future: A vision of computational fluid dynamics in 2030". In: *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences* 372 (Aug. 2014). DOI: 10.1098/rsta.2013.0317.

[28] Karthik Duraisamy, Gianluca Iaccarino, and Heng Xiao. "Turbulence Modeling in the Age of Data". In: *Annual Review of Fluid Mechanics* 51.1 (2019), pp. 357–377. DOI: 10.1146/annurev-fluid-010518-040547.

[29] D.C. Wilcox. *Turbulence Modeling for CFD*. Turbulence Modeling for CFD v. 1. DCW Industries, 2006. ISBN: 9781928729082.

[30] B. E. Launder and D. B. Spalding. *Lectures in mathematical models of turbulence [by] B. E. Launder and D. B. Spalding*. English. Academic Press London, New York, 1972, 7, 169 p. ISBN: 0124380506.

[31] Heng Xiao and Paola Cinnella. "Quantification of Model Uncertainty in RANS Simulations: A Review". In: (Apr. 2019).

[32] É Turgeon et al. "Application of a sensitivity equation method to the K-epsilon model of turbulence". In: *Optimization and Engineering* 8 (Oct. 2007), pp. 341–372. DOI: `10.1007/s11081-007-9003-5`.

[33] Matthew CollessAndrew M. Dunn, Babak Bahram Shotorban, and Abdelkader Frendi. "Uncertainty Quantification of Turbulence Model Coefficients via Latin Hypercube Sampling Method". In: 2010.

[34] P.D.A. Platteeuw, G.J.A. Loeven, and H. Bijl. "Uncertainty Quantification Applied to the k-epsilon Model of Turbulence Using the Probabilistic Collocation Method". In: *49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. DOI: `10.2514/6.2008-2150`.

[35] L. Margheri et al. "Epistemic uncertainties in RANS model free coefficients". In: *Computers and Fluids* 102 (2014), pp. 315 –335. ISSN: 0045-7930. DOI: `https://doi.org/10.1016/j.compfluid.2014.06.029`.

[36] John Schaefer et al. "Uncertainty Quantification of Turbulence Model Closure Coefficients for Transonic Wall-Bounded Flows". In: *AIAA Journal* 55.1 (2017), pp. 195–213. DOI: `10.2514/1.J054902`.

[37] Michael Emory, Rene Pecnik, and Gianluca Iaccarino. "Modeling Structural Uncertainties in Reynolds-Averaged Computations of Shock/Boundary Layer Interactions". In: *49th AIAA Aerospace Sciences Meeting including the New Horizons*

Forum and Aerospace Exposition. DOI: 10.2514/6.2011-479.

[38] Michael Emory, Johan Larsson, and Gianluca Iaccarino. “Modeling of structural uncertainties in Reynolds-averaged Navier-Stokes closures”. In: *Physics of Fluids* 25.11 (2013), p. 110822. DOI: 10.1063/1.4824659.

[39] Gianluca Iaccarino, Aashwin Ananda Mishra, and Saman Ghili. “Eigenspace perturbations for uncertainty estimation of single-point turbulence closures”. In: *Phys. Rev. Fluids* 2 (2 2017), p. 024605. DOI: 10.1103/PhysRevFluids.2.024605.

[40] Aashwin Ananda Mishra and Gianluca Iaccarino. “Uncertainty Estimation for Reynolds-Averaged Navier–Stokes Predictions of High-Speed Aircraft Nozzle Jets”. In: *AIAA Journal* 55.11 (2017), pp. 3999–4004. DOI: 10.2514/1.J056059.

[41] Wouter Edeling, Gianluca Iaccarino, and Paola Cinnella. “Data-Free and Data-Driven RANS Predictions with Quantified Uncertainty”. In: *Flow Turbulence and Combustion* 100 (Nov. 2017), pp. 593–616. DOI: 10.1007/s10494-017-9870-6.

[42] W.N. Edeling, P. Cinnella, and R.P. Dwight. “Predictive RANS simulations via Bayesian Model-Scenario Averaging”. In: *Journal of Computational Physics* 275 (2014), pp. 65 –91. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2014.06.052.

[43] Wouter N. Edeling et al. “Bayesian Predictions of Reynolds-Averaged Navier–Stokes Uncertainties Using Maximum a Posteriori Estimates”. In: *AIAA Journal* 56.5 (2018), pp. 2018–2029. DOI: 10.2514/1.J056287.

[44] Sai Hung Cheung et al. “Bayesian uncertainty analysis with applications to turbulence modeling”. In: *Reliability Engineering and System Safety* 96.9 (2011). Quantification of Margins and Uncertainties, pp. 1137 –1149. ISSN: 0951-

8320. DOI: https://doi.org/10.1016/j.ress.2010.09.013.

[45] Hiroshi Kato and Shigeru Obayashi. "Approach for uncertainty of turbulence modeling based on data assimilation technique". In: *Computers and Fluids* 85 (2013). International Workshop on Future of CFD and Aerospace Sciences, pp. 2 –7. ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2012.09.002.

[46] Hiroshi Kato et al. "A data assimilation methodology for reconstructing turbulent flows around aircraft". In: *Journal of Computational Physics* 283 (2015), pp. 559 –581. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2014.12.013.

[47] Qiqi Wang and Eric Dow. "Quantification of Structural Uncertainties in the k -w Turbulence Model". In: (Apr. 2011). DOI: 10.2514/6.2011-1762.

[48] Anand Pratap Singh and Karthik Duraisamy. "Using field inversion to quantify functional errors in turbulence closures". In: *Physics of Fluids* 28.4 (2016), p. 045110. DOI: 10.1063/1.4947045.

[49] H. Xiao et al. "Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier–Stokes simulations: A data-driven, physics-informed Bayesian approach". In: *Journal of Computational Physics* 324 (2016), pp. 115 –136. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2016.07.038.

[50] Jinlong Wu, Jian-Xun Wang, and Heng Xiao. "A Bayesian Calibration-Prediction Method for Reducing Model-Form Uncertainties with Application in RANS Simulations". In: *Flow, Turbulence and Combustion* 97 (Oct. 2015). DOI: 10.1007/s10494-016-9725-6.

[51] Jian-Xun Wang, Jinlong Wu, and Heng Xiao. “Incorporating Prior Knowledge for Quantifying and Reducing Model-Form Uncertainty in RANS Simulations”. In: *International Journal for Uncertainty Quantification* 6 (June 2016), pp. 109–126. DOI: 10.1615/Int.J.UncertaintyQuantification.2016015984.

[52] W.P Jones and B.E Launder. “The prediction of laminarization with a two-equation model of turbulence”. In: *International Journal of Heat and Mass Transfer* 15.2 (1972), pp. 301 –314. ISSN: 0017-9310. DOI: https://doi.org/10.1016/0017-9310(72)90076-2.

[53] Svetlana V. Poroseva and Gianluca Iaccarino. “Simulating separated flows using the $k - \varepsilon$ model By”. In: 2002.

[54] Ichiro Kimura and Takashi Hosoda. “A non-linear k–ε model with reliability for prediction of flows around blunt bodies”. In: *International Journal for Numerical Methods in Fluids* 42 (July 2003). DOI: 10.1002/fld.540.

[55] P. A. Durbin. “Separated flow computations with the k-ε-v-squared model”. In: *AIAA Journal* 33.4 (1995), pp. 659–664. DOI: 10.2514/3.12628.

[56] Wouter Edeling et al. “Bayesian estimates of parameter variability in the $k - \varepsilon$ turbulence model”. In: *Journal of Computational Physics* 258 (Feb. 2014), pp. 73–94. DOI: 10.1016/j.jcp.2013.10.027.

[57] Jaideep Ray et al. “Bayesian Calibration of a RANS Model with a Complex Response Surface - A Case Study with Jet-in-Crossflow Configuration”. In: June 2015. DOI: 10.2514/6.2015-2784.

[58] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[59] HKarthik Duraisamy. “A Framework for Turbulence Modeling using Big Data”. In: *Summary of Research, 2015-2017* ().

[60] Svetlana Poroseva and Scott M. Murman. "Velocity/Pressure-Gradient Correlations in a FORANS Approach to Turbulence Modeling". In: June 2014. ISBN: 978-1-62410-289-9. DOI: 10.2514/6.2014-2207.

[61] B.B.C.H.T. Richard C. Aster et al. *Parameter Estimation and Inverse Problems*. International geophysics series. Elsevier Science, 2005. ISBN: 9780120656042.

[62] P.K. Kundu, I.M. Cohen, and D.R. Dowling. *Fluid Mechanics*. Science Direct e-books. Elsevier Science, 2012. ISBN: 9780123821003.

[63] S. B. Pope. "A more general effective-viscosity hypothesis". In: *Journal of Fluid Mechanics* 72.2 (1975), 331–340. DOI: 10.1017/S0022112075003382.

[64] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance". In: *Journal of Fluid Mechanics* 807 (2016), 155–166. DOI: 10.1017/jfm.2016.615.

[65] S. Yarlanki, B. Rajendran, and H. Hamann. "Estimation of turbulence closure coefficients for data centers using machine learning algorithms". In: *13th InterSociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*. 2012, pp. 38–42. DOI: 10.1109/ITHERM.2012.6231411.

[66] Eric J. Parish and Karthik Duraisamy. "A paradigm for data-driven predictive modeling using field inversion and machine learning". In: *Journal of Computational Physics* 305 (2016), pp. 758 –774. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2015.11.012.

[67] Karthikeyan Duraisamy, Ze J. Zhang, and Anand Pratap Singh. "New Approaches in Turbulence and Transition Modeling Using Data-driven Techniques". In: *53rd AIAA Aerospace Sciences Meeting*. DOI: 10.2514/6.2015-1284.

[68] Ze Jia Zhang and Karthik Duraisamy. “Machine Learning Methods for Data-Driven Turbulence Modeling”. In: June 2015. DOI: 10.2514/6.2015-2460.

[69] Brendan D. Tracey, Karthikeyan Duraisamy, and Juan J. Alonso. “A Machine Learning Strategy to Assist Turbulence Model Development”. In: *53rd AIAA Aerospace Sciences Meeting*. DOI: 10.2514/6.2015-1287.

[70] Anand Pratap Singh, Shivaji Medida, and Karthik Duraisamy. “Machine Learning-augmented Predictive Modeling of Turbulent Separated Flows over Airfoils”. In: *CoRR* abs/1608.03990 (2016). arXiv: 1608.03990. URL: http://arxiv.org/abs/1608.03990.

[71] Jian-Xun Wang and Heng Xiao. “Data-driven CFD modeling of turbulent flows through complex structures”. In: *International Journal of Heat and Fluid Flow* 62 (2016), pp. 138–149. ISSN: 0142-727X. DOI: https://doi.org/10.1016/j.ijheatfluidflow.2016.11.007.

[72] Julia Ling, Reese Jones, and Jeremy Templeton. “Machine learning strategies for systems with invariance properties”. In: *Journal of Computational Physics* 318 (2016), pp. 22 –35. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2016.05.003.

[73] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. “Reynolds averaged turbulence modelling using deep neural networks with embedded invariance”. In: *Journal of Fluid Mechanics* 807 (2016), pp. 155–166.

[74] Pedro M. Milani, Julia Ling, and John K. Eaton. “Generalization of Machine-Learned Turbulent Heat Flux Models Applied to Film Cooling”. In: June 2019.

[75] Pedro M. Milani et al. “A Machine Learning Approach for Determining the Turbulent Diffusivity in Film Cooling Flows”. In: *Journal of Turbomachinery* 140 (Oct. 2017). DOI: 10.1115/1.4038275.

[76] Harshal Akolekar et al. “Development and Use of Machine-Learnt Algebraic Reynolds Stress Models for Enhanced Prediction of Wake Mixing in LPTs”. In: May 2018. DOI: `10.1115/GT2018-75447`.

[77] Jack Weatheritt et al. “Machine Learning for Turbulence Model Development Using a High-Fidelity HPT Cascade Simulation”. In: June 2017, V02BT41A015. DOI: `10.1115/GT2017-63497`.

[78] Richard Sandberg et al. “Applying Machine Learnt Explicit Algebraic Stress and Scalar Flux Models to a Fundamental Trailing Edge Slot”. In: *Journal of Turbomachinery* 140 (Sept. 2018), p. 101008. DOI: `10.1115/1.4041268`.

[79] Jack Weatheritt et al. “A Comparative Study of Contrasting Machine Learning Frameworks Applied to RANS Modeling of Jets in Crossflow”. In: *Volume 2B: Turbomachinery*. Charlotte, US, 2017, pp. 1–10. DOI: `10.1115/GT2017-63403`.

[80] Jinlong Wu, Heng Xiao, and Eric G. Paterson. “Data-Driven Augmentation of Turbulence Models with Physics-Informed Machine Learning”. In: 2018.

[81] Jian-Xun Wang, Jinlong Wu, and Heng Xiao. “Physics informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data”. In: *Physical Review Fluids* 2 (Feb. 2017), pp. 1–22. DOI: `10.1103/PhysRevFluids.2.034603`.

[82] Tieghi Lorenzo et al. “Assessment Of A Machine-Learnt Adaptive Wall-Function in a Compressor Cascade With Sinusoidal Leading Edge.” In: (June 2019). DOI: `ASMEGT2019-91238`.

[83] Michele Milano and Petros Koumoutsakos. “Neural Network Modeling for Near Wall Turbulent Flow”. In: *Journal of Computational Physics* 182.1 (2002), pp. 1 –26. ISSN: 0021-9991. DOI: `https://doi.org/10.1006/jcph.2002.7146`.

[84] Corsini Alessandro et al. “Identification of poorly ventilated zones in gas-turbine enclosures with machine learning.” In: June 2019.

[85] Pedro Domingos. *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*. New York, NY, USA: Basic Books, Inc., 2018. ISBN: 0465094279, 9780465094271.

[86] Edward Lee Thorndike. “Animal intelligence: An experimental study of the associate processes in animals.” In: *American Psychologist* 53.10 (1998), p. 1125.

[87] Burrhus F Skinner. “Operant conditioning”. In: *The encyclopedia of education* 7 (1971), pp. 29–33.

[88] J.O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer Series in Statistics. Springer New York, 2013. ISBN: 9781475742862.

[89] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford University Press, Inc., 1995. ISBN: 0198538642.

[90] Diederik Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *International Conference on Learning Representations* (Dec. 2014).

[91] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *CoRR* abs/1609.04747 (2016).

[92] Timothy Craft et al. “Wall-function strategies for use in turbulent flow CFD”. In: Jan. 2002. DOI: `10.1615/IHTC12.3100`.

[93] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[94] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015.

[95] “OpenFoam User Guide v5”. In:

[96] J. Nathan Kutz. “Deep learning in fluid dynamics”. In: *Journal of Fluid Mechanics* 814 (2017), 1–4. DOI: 10.1017/jfm.2016.803.

[97] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. 10. Springer series in statistics New York, 2001.

[98] Jochen Fröhlich et al. “Highly resolved large-eddy simulation of separated flow in a channel with streamwise periodic constriction”. In: *Journal of Fluid Mechanics* 526 (Mar. 2005), pp. 19 –66. DOI: 10.1017/S0022112004002812.

[99] HUNG LE, PARVIZ MOIN, and JOHN KIM. “Direct numerical simulation of turbulent flow over a backward-facing step”. In: *Journal of Fluid Mechanics* 330 (1997), 349–374. DOI: 10.1017/S0022112096003941.

[100] Matteo Bernardini, Sergio Pirozzoli, and Paolo Orlandi. “Velocity statistics in turbulent channel flow up to Ret =4000”. In: *Journal of Fluid Mechanics* 742 (Feb. 2014). DOI: 10.1017/jfm.2013.674.

[101] Ahmad Sohankar, Lars Davidson, and Christoffer Norberg. “Large eddy simulation of flow past a square cylinder: comparison of different subgrid scale models”. In: *Journal of Fluids Engineering* 122.1 (2000), pp. 39–47.

[102] Junfei Qiu et al. “A survey of machine learning for big data processing”. In: *EURASIP Journal on Advances in Signal Processing* 2016.1 (2016), p. 67. ISSN: 1687-6180. DOI: 10.1186/s13634-016-0355-x.

[103] Liu F. “A Thorough Description Of How Wall Functions Are Implemented In OpenFOAM”. In: *n Proceedings of CFD with OpenSource Software, 2016* ().

[104] M. Breuer et al. “Flow over periodic hills – Numerical and experimental study in a wide range of Reynolds numbers”. In: *Computers and Fluids* 38.2 (2009), pp. 433 –457. ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2008.05.002.

[105] Alessandro Corsini, Giovanni Delibra, and Anthony G. Sheard. "On the Role of Leading-Edge Bumps in the Control of Stall Onset in Axial Fan Blades". In: *Journal of Fluids Engineering* 135.8 (June 2013). 081104. ISSN: 0098-2202. DOI: 10.1115/1.4024115.

[106] Tieghi Lorenzo et al. "Exploration of axial fan design space using a metamodel for aerodynamic properties of NACA 4-digit profiles." In: (June 2019). DOI: ASMEGT2019-91588.

[107] François Chollet. *keras*. https://github.com/fchollet/keras. 2015.

[108] Jr. Dean Robert C. and Yasutoshi Senoo. "Rotating Wakes in Vaneless Diffusers". In: *Journal of Basic Engineering* 82.3 (Sept. 1960), pp. 563–570. DOI: 10.1115/1.3662659.

[109] John D. Stanitz and Gaylord O Ellis. "Two-dimensional compressible flow in centrifugal compressors with straight blades". In: *NACA Annual Report 36* (Jan. 1950), pp. 141–163.

[110] S. Menon and W.-W. Kim. "High Reynolds number flow simulations using the localized dynamic subgrid-scale model". In: 1996.

[111] Brian Launder and B.I. Sharma. "Application of the energy-dissipation model of flow near a spinning disc". In: *Lett. Heat Mass Transfer* (Jan. 1974), pp. 131–138.

[112] P. Davidson. "Turbulence: An Introduction for Scientists and Engineers". In: Jan. 2004. ISBN: 019852949X. DOI: 10.1063/1.2138427.

[113] Balakrishnan Ponnuraj et al. "3D CFD Analysis of an Industrial Gas Turbine Compartment Ventilation System". In: Jan. 2003. DOI: 10.1115/IMECE2003-41672.

[114] *Numerical Assessment of Fan-Ducting Coupling for Gas Turbine Ventilation Systems*. Vol. Volume 1: Aircraft Engine; Fans and Blowers; Marine. Turbo Expo: Power for Land, Sea, and Air. June 2015. DOI: 10.1115/GT2015-42449.

[115] Alessandro Corsini et al. "CFD Analysis of Ventilation Systems for Gas Turbine Enclosures". In: 2015.

[116] ISO Central Secretary. *Gas turbine applications – Safety*. Standard. Geneva, CH: International Organization for Standardization, 2009.

[117] *Experimental and Numerical Investigation on Gas Turbine Package Scale Model*. Vol. Volume 9: Oil and Gas Applications; Supercritical CO2 Power Cycles; Wind Energy. Turbo Expo: Power for Land, Sea, and Air. June 2018. DOI: 10.1115/GT2018-75694.

[118] Roger Santon, Jasper Kidger, and Chris Lea. "Safety Developments in Gas Turbine Power Applications". In: Jan. 2002. DOI: 10.1115/GT2002-30469.

[119] Roger Santon, Mat Ivings, and David Pritchard. "A New Gas Turbine Enclosure Ventilation Design Criterion". In: Jan. 2005. DOI: 10.1115/GT2005-68725.

[120] Matteo Bernardini, Sergio Pirozzoli, and Paolo Orlandi. "Velocity statistics in turbulent channel flow up to $Re_\tau = 4000$". In: *Journal of Fluid Mechanics* 742 (2014), 171–191. DOI: 10.1017/jfm.2013.674.

[121] HUNG LE, PARVIZ MOIN, and JOHN KIM. "Direct numerical simulation of turbulent flow over a backward-facing step". In: *Journal of Fluid Mechanics* 330 (1997), 349–374. DOI: 10.1017/S0022112096003941.

[122] Siva Thangam. "Analysis of Two-Equation Turbulence Models for Recirculating Flows." In: 1991.

[123] Y Jang, Lionel Temmerman, and M Leschziner. "Investigation of anisotropy-resolving turbulence models by reference to highly-resolved LES data for separated flow". In: *European Community on Computational Methods in Applied Sciences ECCOMAS Computational Fluid Dynamics Conference* (Oct. 2001), pp. 4–7.

[124] Bruno Chaouat and Roland Schiestel. “Hybrid RANS/LES simulations of the turbulent flow over periodic hills at high Reynolds number using the PITM method”. In: *Computers & Fluids* 84 (2013), pp. 279 –300. ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2013.06.012.

[125] Jony Castagna, Yufeng Yao, and Jun Yao. “Direct numerical simulation of a turbulent flow over an axisymmetric hill”. In: *Computers & Fluids* 95 (2014), pp. 116 –126. ISSN: 0045-7930. DOI: https://doi.org/10.1016/j.compfluid.2014.02.014.

[126] Michael Breuer. “A challenging test case for large eddy simulation: high Reynolds number circular cylinder flow”. In: *International Journal of Heat and Fluid Flow* 21.5 (2000). Turbulence and Shear Flow Phenomena 1, pp. 648 –654. ISSN: 0142-727X. DOI: https://doi.org/10.1016/S0142-727X(00)00056-4.

[127] Z.F. Tian et al. “Numerical studies of indoor airflow and particle dispersion by large Eddy simulation”. In: *Building and Environment* 42.10 (2007), pp. 3483 –3492. ISSN: 0360-1323. DOI: https://doi.org/10.1016/j.buildenv.2006.10.047.