The University Of Sheffield.

Aerospace Engineering.

Towards statistical inference to improve turbulence RANS closures for multi-element aerofoils

by

Omid Bidar

May, 2020

Supervisor: Professor Ning Qin

Dissertation submitted to the University of Sheffield
in partial fulfilment of the requirements for the
degree of Master of Engineering

Abstract

While remaining an indispensable tool in design, predictions of turbulent flows based on computational fluid dynamics analyses continue to face challenges. High-fidelity approaches based on resolving the Navier-Stokes equations in one form or other are predicted to remain computationally intractable for the near future. This means the Reynolds-averaged Navier-Stokes models remain the workhorse of industrial CFD analyses. RANS closures are developed based on various modelling assumptions which are plagued with inaccuracies in complex physical flows such as flow separation over high-lift multi-element aerofoils, essential for takeoff and landing phases of an aircraft flight envelope. This thesis explores the application of a new paradigm in turbulence modelling which leverages high-fidelity data such as those from experiments, and machine learning algorithms to augment RANS closures. Through statistical inference a discrepancy field is generated by formulating a highly dimensional, deterministic, gradient-based optimisation problem where the goal is to minimise the error between a base RANS model and experimental data in every cell of the computational grid.

The main contribution of the present work is to implement a discrete adjoint method for efficient derivative computations using open-source software. The errors in the adjoint derivative computations are about 2%. Additionally, the data-driven turbulence modelling paradigm is explored through an extensive literature survey; three multi-element aerofoils and the associated experimental data are identified for the task of statistical inference; the capabilities of two popular RANS models (Spalart-Allmaras and $k-\omega$ shear stress transport) in predicting flows around a representative three-element high-lift aerofoil are evaluated against experimental data; and to conclude, key questions and possible ways of investigating these are highlighted for future work.

"Even these things will one day be a pleasure to recall, perhaps."

Virgil, The *Aeneid*

Acknowledgements

I would like to thank my supervisor, Prof. Ning Qin for agreeing to advise me on this self-proposed project. Thanks is also due to Dr. Ping He at the University of Michigan for his generous guidance with DAFoam, and Dr. Spiridon Siouris for his help with the HPC at Sheffield. I look forward to future collaboration with all three.

I am also infinitely grateful to my family for everything, and dedicate this thesis to رهجو؛ پریما؛ پویا؛ آبی؛ آتی.

Quote from the Aeneid translated by Llewelyn Morgan.

Contents

List of Figures

List of Tables

Nomenclature

General

$(\cdot)_\infty$	freestream value of a quantity
$(\cdot)_{\text{expt}}$	experimental value of a quantity
$(\cdot)_{\text{perturb}}$	perturbed value of a quantity
$(\cdot)_{\text{ref}}$	reference value of a quantity
$[\boldsymbol{M}]_{PC}$	pre-conditioner of a matrix, $\boldsymbol{M}$
$\mathfrak{D}$	data set for field inversion
$\frac{Df}{Dt}$	total derivative of a function f, $\frac{Df}{dt} = \frac{\partial f}{\partial t} + \boldsymbol{U} \cdot \nabla f$
$(\cdot)'$	fluctuating component, e.g. $u_i^{'}$
$\mathbb{R}$	the set of real numbers
$\mathcal{D}$	destruction term
$\mathcal{J}$	objective or cost function
$\mathcal{K}$	Krylov subspace, $\mathcal{K}_i = \text{span}(r_0, Ar_0, A^2r_0, \ldots, A^{i-1}r_0)$
$\mathcal{P}$	production term
$\mathcal{T}$	transport term
$\overline{(\cdot)}$	ensemble averaging or space filtering
$\boldsymbol{\mathcal{R}}$	residual vector, $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{n_w}$, $\boldsymbol{\mathcal{R}} = [\mathcal{R}_u, \mathcal{R}_v, \mathcal{R}_w, \mathcal{R}_p, \mathcal{R}_{\nu_t}, \mathcal{R}_\phi]^T$
$\boldsymbol{A}$	transpose of the state Jacobian, $\boldsymbol{A} = [\partial \boldsymbol{\mathcal{R}} / \partial \boldsymbol{w}]^T$
$\boldsymbol{C}_{obs}$	observational covariance matrix
$\boldsymbol{C}_{prior}$	prior covariance matrix
$\boldsymbol{r}_0$	initial residual, $\boldsymbol{r}_0 = [\partial \mathcal{J} / \partial \boldsymbol{w}]^T - [\partial \boldsymbol{\mathcal{R}} / \partial \boldsymbol{w}]^T \boldsymbol{\psi}_0$
$\boldsymbol{U}$	velocity vector, $\boldsymbol{U} = [u, v, w]^T$
$\boldsymbol{w}$	state vector, $\boldsymbol{w} \in \mathbb{R}^{n_w}$, $\boldsymbol{w} = [u, v, w, p, \nu_t, \phi]^T$
$\boldsymbol{x}$	coordinates, $\boldsymbol{x} = [x, y, z]^T$
a_{ij}	Reynolds stress anisotropy tensor
c	chord length
C_d	two-dimensional drag coefficient, $C_d = D' / \frac{1}{2}\rho U_\infty^2 c$
C_l	two-dimensional lift coefficient, $C_l = L' / \frac{1}{2}\rho U_\infty^2 c$
C_p	pressure coefficient, $C_p = (p - p_\infty) / \frac{1}{2}\rho U_\infty^2 c$
$D^{'}$	two-dimensional drag force
k	turbulent kinetic energy
l	length
$L^{'}$	two-dimensional lift force
M	mach number
N_m	number of mesh cells
$n_{(\cdot)}$	number of elements of a vector, e.g. n_w
P	mean pressure
p	instantaneous pressure
Re	Reynolds number, $Re = \rho l_{\text{ref}} U_{\text{ref}} / \mu$
t	time
U	mean velocity
u_i	velocity component, index notation

x_i	Cartesian coordinates, index notation

Greek Symbols

α	angle of attack
$\boldsymbol{\beta}$	corrective term accounting for model discrepancy
$\boldsymbol{\eta}$	set of machine learning features
$\boldsymbol{\psi}$	adjoint vector, $\boldsymbol{\psi} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{w}} \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{w}}^{-1}$
χ	ratio of surrogate to freestream viscosity, $\chi = \tilde{\nu}/\nu$
δ	deflection angle of the high-lift aerofoil elements
ϵ	rate of dissipation of turbulent kinetic energy
λ	regularisation constant
μ	dynamic viscosity
∇	partial derivative operator, for $\mathbb{R}^3$ system, $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)$
ν_t	turbulent kinematic viscosity
Ω	vorticity magnitude
ω	specific turbulent dissipation rate
ϕ	surface flux
ρ	density
τ_{ij}	Reynolds stress, or second moment, $\tau_{ij} = -\overline{u_i^{'} u_j^{'}}$
$\tilde{\nu}$	surrogate viscosity for Spalart-Allmaras model

Abbreviations

CFD	computational fluids dynamics
DA	discrete adjoint
DES	detached eddy simulations
DLR	German Aerospace Centre (Deutsches Zentrum für Luft- und Raumfahrt)
DNS	direct numerical simulations
FIML	field inversion and machine learning
LES	large eddy simulations
MAP	maximum a posteriori
MDA	McDonnell Douglas Aerospace
ML	machine learning
N-S	Navier-Stokes equations
NHLP	National High-Lift Programme (UK)
OpenFOAM	open field operation and manipulation
RANS	Reynolds-averaged Navier-Stokes equations
S-A	Spalart-Allmaras turbulence model
SST	shear stress transport

"When I die and go to heaven there are two matters on which I hope for enlightenment. One is quantum electrodynamics, and the other is the turbulent motion of fluids. And about the former I am rather optimistic."

Horace Lamb

1 Introduction

1.1 Motivations

This section motivates the core reasons behind the choice of the current project by putting it into context in the field of fluid mechanics in general and computational fluid dynamics and turbulence modelling in particular.

1.1.1 The need for predicting turbulent flows

Turbulence is a ubiquitous flow phenomenon, extending from ocean, to atmosphere, to astrophysical processes such as solar flares. Engineering applications include fluid processing in pipelines, pumps, compressors; flows around sea, land, and air vehicles; or mixing of various fluids in engines, chemical reactors, and many more. Therefore, understanding turbulent flow behaviour is crucial in the design and optimisation of the listed applications.

Figure 1-1: Graphical representation of how "CFD moved from an exploratory tool to a full flight physics production" in Airbus aircraft design cycle, image for CFD application in 2010, and quote from the company's head of aerodynamic research and technology, [1].

Within the aerospace industry, wind-tunnel testing and computational fluid dynamics (CFD) are regularly performed to understand turbulent flow behaviour. Due to long turnaround times involved in wind-tunnel testing and associated expenses, CFD is seen as

an attractive tool throughout the design cycle since it has the potential for reducing the expense, complexity, and technical limitation of a purely experimental design approach; providing insight into flow physics; and enabling design optimisation [2].

Despite decades of research, however, CFD approaches for simulating turbulent flow behaviour still face significant challenges which will be briefly outlined in the following subsections.

1.1.2 High-fidelity approaches and their limitations

Mathematical representation of flow phenomenon are based on centuries old conservation laws of mass, momentum and energy, incorporated in the Navier-Stokes (N-S) equations[1], which remain the cornerstones of fluid mechanics. For an incompressible, Newtonian fluid, with constant-property flow, the N-S equations are;

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{1.1}$$

$$\frac{\partial u_i}{\partial t} + \frac{\partial u_i u_j}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{1}{Re}\frac{\partial^2 u_i}{\partial x_j \partial x_j}, \tag{1.2}$$

where u_i is the velocity, x_i is the spatial coordinates, p is pressure, $Re = \rho l_{\text{ref}} U_{\text{ref}}/\mu$ is the Reynolds number based on density, reference length and velocity, and kinematic viscosity respectively.

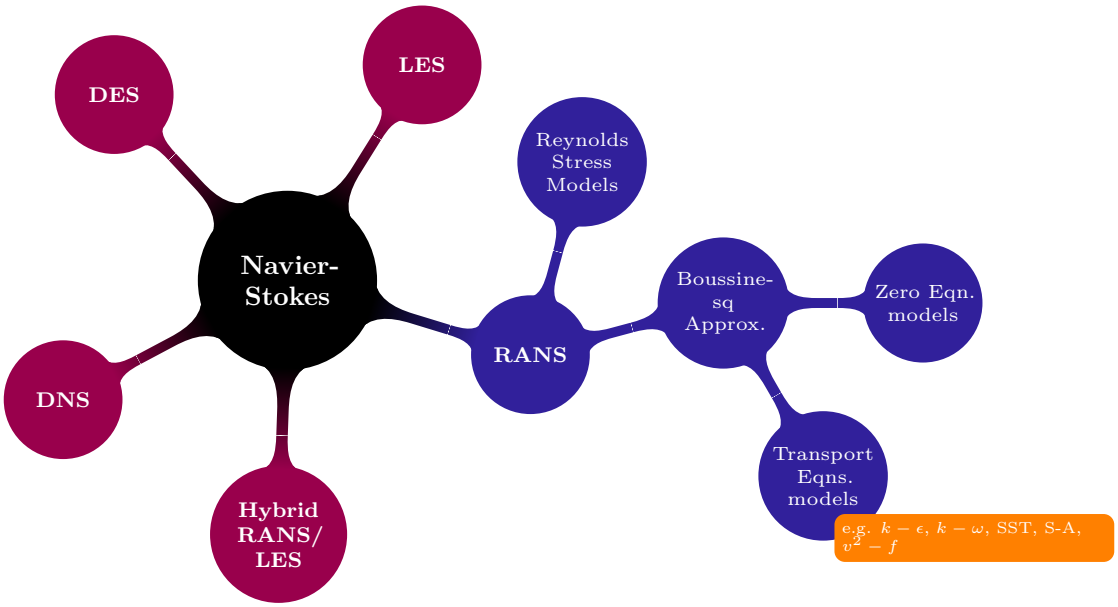


Figure 1-2: Classification of the main turbulence simulation and modelling approaches, diagram inspired from [3]. Abbreviations: DNS, direct numerical simulations; DES, detached eddy simulations; LES, large eddy simulations; RANS, Reynolds-averaged Navier-Stokes; SST, shear stress transport; S-A, Spalart-Allmaras.

The main approaches to the use of N-S equations is illustrated in Figure 1-2. Highest-fidelity can be achieved through direct numerical simulation (DNS) which *fully resolves*

[1] One of the seven most important open problems in mathematics according to the Clay Mathematics Institute

the N-S equations in space and time [4]. However, DNS approaches remains intractable due to the incredible amount of computational resources required for the resolution of complex industrial flows, as by one estimate the number of mesh cells required for DNS simulations grow at approximately Re^3 [5], whereas a typical Re for aerospace applications is $\mathcal{O}(10^6)$ and higher.

An alternative is large-eddy simulations (LES) [6] which involves resolving the large-scale turbulence effects while a filtering process is applied to scales below a threshold. The filtering process gives rise to the Smagorinsky stress tensor term, τ_{ij}, in the momentum equation (Eqn. 1.2), which leads to the need for models to close the equation. Despite reducing the computational costs compared to the DNS approach, LES unfortunately still remains prohibitively expensive for wall-bounded high Reynolds number flows as the energetic scales - even if small - dominate the dynamics in the near-wall region [7].

1.1.3 RANS modelling and associated inaccuracies

Turbulence models based on Reynolds-averaged Navier-Stokes (RANS) equations [8] remain the workhorse of most CFD analyses due to their relative simplicity, ease of implementation, and cosiderably lower computational cost. The RANS equations are so called because the instantaneous velocity and pressure (u_i and p) in the N-S equations (Eqn. 1.1, and 1.2) are decomposed into a mean (U_i and P) and fluctuating ($u_i^{'}$ and $p^{'}$) component through statistical ensemble averaging [9], leading to:

$$\frac{\partial U_i}{\partial x_i} = 0, \tag{1.3}$$

$$\frac{\partial U_i}{\partial t} + \frac{\partial (U_i U_j)}{dx_j} = -\frac{\partial P}{dx_i} + \frac{1}{Re}\frac{\partial^2 U_i}{\partial x_j \partial x_j} - \frac{\partial \overline{u_i' u_j'}}{\partial x_j}. \tag{1.4}$$

The rationale behind the averaging process is that for most of the engineering applications where turbulent flow behaviour simulations are required, the quantities of interest only depend on mean flow (e.g. force coefficients for aerodynamic design optimisation), thus the instantaneous fields are not of concern [10].

These models are formulated based on a combination of intuition, empiricism, and theoretical constraints. The most widely used models – $k-\omega$ [11], $k-\epsilon$ [12], Spalart-Allmaras (S-A) [13], etc. – assume a linear relation between the eddy viscosity, ν_t, and Reynolds stresses, and then use surrogate variable(s) to model ν_t. However, these models have well-documented deficiencies in complex flows involving strong flow curvatures, separations, and strong unsteadiness (e.g. [14, 15]), which arise due to ensemble-averaging, the various assumptions during closure formulation, and calibration of model parameters (i.e. model constants) based on a limited number of canonical flow cases.

Figure 1-3 illustrates the rise in the use of popular RANS models against wind tunnel tests in the design cycle of commercial aircraft. It is important to note that the linear eddy-viscosity models developed by the 1990s are still in use. While a plethora of elegant ideas have spurred in the turbulence research community such as non-linear eddy viscosity models [17], elliptic relaxation [18], and hybrid RANS-LES [19] to address the deficiencies, their complexity and lack of generalisation have limited their applications, leading to the conclusion that traditional RANS modelling approaches may have reached a plateau [20].

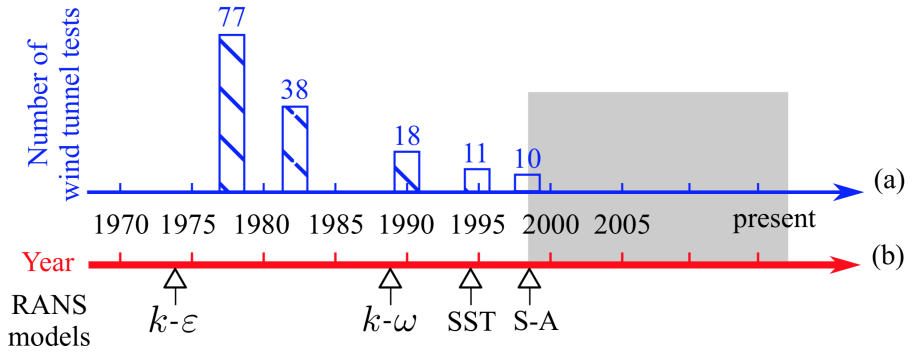


Figure 1-3: Shaded region representative of stagnation in turbulence models, (a) timeline of experiments required in the design cycle of commercial aircraft in the past five decade, (b) the years of development of popular RANS turbulence models. Adapted from [16] based on data from [10].

1.1.4 The case for recent data-driven turbulence modelling

Arguably traditional RANS model is inherently one of data-assimilation in that calibrating them involve the use of high-fidelity data (experimental or DNS) for determining model coefficients and functions empirically.

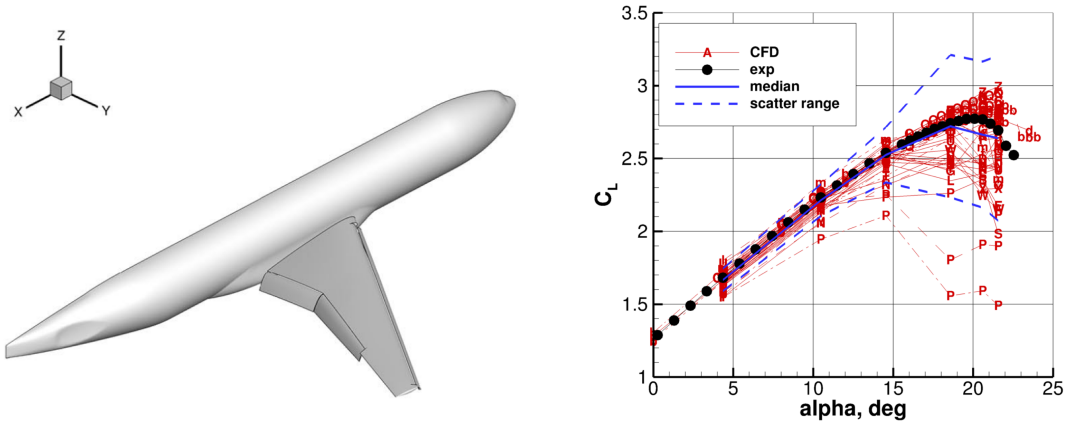
While data-driven techniques such as machine learning are decades old, the recent rise in their application in various disciplines, including fluid mechanics and turbulence modelling are driven by, [21]: 1) abundance of large datasets (e.g. particle image velocimetry (PIV) data [22], and high-fidelity DNS and LES data); 2) the rapid growth in computational hardware capabilities; 3) rapid advances in algorithms; and 4) the availability of open-source algorithms and benchmark cases. Based on these, the CFD community is beginning to tap into machine learning and other data-driven techniques that can potentially lead to a paradigm shift in the field. Various proposed data-driven techniques are discussed in the following chapter.

1.1.5 High-lift multi-element systems

Multi-element high-lift configurations are employed for improved aircraft performance in the take-off, landing, and some manoeuvring phases of an aircraft's flight envelope. In this work, three-element aerofoils consisting of a slat (fore element), main element, and a flap (aft element) are investigated. The addition of the slat and flap elements is to allow the reshaping of the wing, and hence changing the lift characteristics. Generally, the role of the flap is to increase the lift by increasing the effective camber of the aerofoil, hence increasing the amount of air being deflected. The slat is employed to prevent very large adverse pressure gradients on the main aerofoil leading to flow separation. An extensive review of the modelling techniques for predicting high-lift flows is provided in [23].

Predicting high-lift multi-element aerofoil flows has received considerable attention within the CFD community, e.g. UK's National High-Lift Programme in 1970s, NASA workshops in 1990s [24], American Institute of Aeronautics and Astronautics (AIAA) High Lift Prediction Workshops (latest in 2017) [25] etc. As an example, Figure 1-4 shows various RANS predictions of lift coefficient C_L for the NASA High-Lift Common Research Model (HL-CRM) at a Mach number of 0.2, and freestream Reynolds number (based on mean aerodynamic chord) of 3.26 million. Although predictions at lower angles of

attack α have reasonable agreement with wind-tunnel experimental data, the prediction capabilities at higher angles show high variability.



(a) NASA HL-CRM **(b)** RANS predictions

Figure 1-4: Comparative RANS simulation lift coefficient results against experimental data from the 3rd AIAA High Lift Prediction Workshop, for details refer to the source: [25].

1.2 Goals of present work

This thesis aims to develop an understanding of the new frontier in turbulence prediction research: the use of data-driven techniques to augment RANS-based turbulence modelling. One framework is field inversion and machine learning (FIML), which is based on the following three steps: 1) Minimise the difference between baseline turbulence model and data by solving inverse problems based on statistical inference; 2) construct a model of the inferred discrepancy from previous step using machine learning algorithms; and 3) embed the model in a RANS solver for improved predictions. Key advantages of this framework are: a relatively low number of inferred fields can be used to train the machine learning algorithm on [26]; relatively spare experimental data such as lift coefficients can be used to achieve substantial improvements [27]; and the field inversion phase can provide valuable insight to the modeller [28].

The field inversion phase involves solving a gradient-based optimisation problem where the number of design variables is equal to the number of mesh cells. The two-dimensional mesh for the three-element aerofoils considered in this work are $\mathcal{O}(10^5)$. Due to the extremely high-dimensionality, it is crucial to compute the gradients efficiently.

Thus, the specific objectives of this project are as follows:

- Explore the short-comings of two popular RANS models: Spalart-Allmaras and $k-\omega$ shear stress transport (SST) in predicting flows around multi-element high-lift aerofoils.
- Develop an understanding of the state-of-the-art in data-driven turbulence modelling through an extensive literature survey.
- Identify multi-element aerofoils and experimental data for statistical inference.

- Implement and verify an efficient adjoint-based gradient computation for the baseline Spalart-Allmaras RANS model as a key step towards field inversion.

1.3 Thesis layout

Chapter 1 has served as an introduction to the state of turbulence simulation and modelling approaches and the rationale behind data-driven approaches in general, followed by formulation of the aims of the present work. The remainder of the thesis is structured as follows:

- Chapter 2 is concerned with a literature survey of the various ways data-driven turbulence modelling has been proposed in the research community; a detailed discussion of the field inversion and machine learning framework; and a brief discussion of some associated points of contention.
- Chapter 3 deals with the theory behind the field inversion and the adjoint approach.
- In Chapter 4, the implementational aspects are outlined: these include implementing the adjoint method, details of experimental data to be used for the inverse problems, and various aspects of the CFD simulation setup such as the meshes.
- Chapter 5 presents the results for the baseline RANS simulations, with an evaluation of the adjoint implementation.
- Finally, in Chapter 6 some conclusions are drawn, with plans for future work.

"The potential impact is high so long as outcomes are held to the long-standing critical standards that should guide studies of flow physics."

Perspective on machine learning for advancing fluid mechanics, Brenner et al.

2 Data-driven turbulence modelling

The use of machine learning in turbulence modelling can be broadly categorised into approaches where machine learning is used to calibrate model parameters, identify and quantify traditional turbulence modelling uncertainties, and techniques to improve overall prediction performance by using data. Detailed discussions are also available in the review papers [29, 21, 30].

2.1 Model parameter calibration

In RANS models uncertainties arise in ensemble-averaging, the assumptions when formulating model closures, and calibrating model parameters (i.e. model constants). Some machine learning based techniques to address these are discussed below.

In [31] the optimum model coefficients in the $k-\epsilon$ model is found using artificial neural networks. This is achieved by minimising the error between the experimental and simulation data. The optimum coefficients reduce the absolute average error by 35%. A similar, but a more rigorous approach is used in [32]. Bayesian data analysis techniques are applied for a number of flows with different pressure gradients to estimate the $k-\epsilon$ model coefficients based on boundary-layer velocity profiles. The limitation of these approaches come from the fact that functional forms of the RANS models remain unchanged, i.e. only coefficients are calibrated, hence the uncertainties arising from the functional forms remain unaddressed.

2.2 Uncertainty quantification

To characterise structural errors, [33] uses statistical inference to quantify the errors in the turbulence viscosity field in $k-\omega$ model. The discrepancy between the standard model and DNS results is computed using inverse modelling, then modelled as a Gaussian random field, and the uncertainty is propagated to the quantities of interest. Bayesian techniques are similarly applied for the S-A model to quantify functional form uncertainties in [34].

Another approach, employed in [35], is to identify regions of high RANS uncertainty by utilising machine learning techniques on high-fidelity data. More detailed overview of data-driven uncertainty quantification can be found in [10].

2.3 Data-driven predictive modelling

This section gives an overview of data-driven techniques employed to improve the overall performance of traditional turbulence models. The majority of data-driven machine

learning attempts involve directly predicting, or applying corrections to the Reynolds stress anisotropy tensor, a_{ij}.

Evolutionary algorithms are employed in [36] to obtain analytical expressions for a_{ij} resulting in an interpretable model which can be used to provide insight into turbulence models, however are limited in application for complex geometries [37]. Evolutionary algorithms are also used in [38] to construct a non-linear stress-strain relationship to account for the extra anisotropy neglected in the Boussinesq assumption.

In [39] neural networks are used to learn the functional form of the discrepancy in a_{ij} between LES and baseline linear and non-linear eddy-viscosity models. This is then embedded in a predictive simulation resulting in significantly improved results. DNS data is used in [40] to reconstruct discrepancies in a_{ij}.

One of the key limitations of the techniques above is the need for detailed high-fidelity data such as Reynolds stress anisotropy tensor for training the machine learning algorithms. A more general technique that can overcome this limitation is the field inversion and machine learning framework described in the following section.

2.4 Field inversion and machine learning

The FIML framework, the subject of [5, 41, 26, 42, 27, 43, 44, 45], is based on the following three steps: Minimise the difference between baseline turbulence model and data by solving inverse problems; construct a model of the inferred discrepancy from previous step using machine learning algorithms; and finally embed the model in a RANS solver for improved predictions, as illustrated in Figure 2-1.

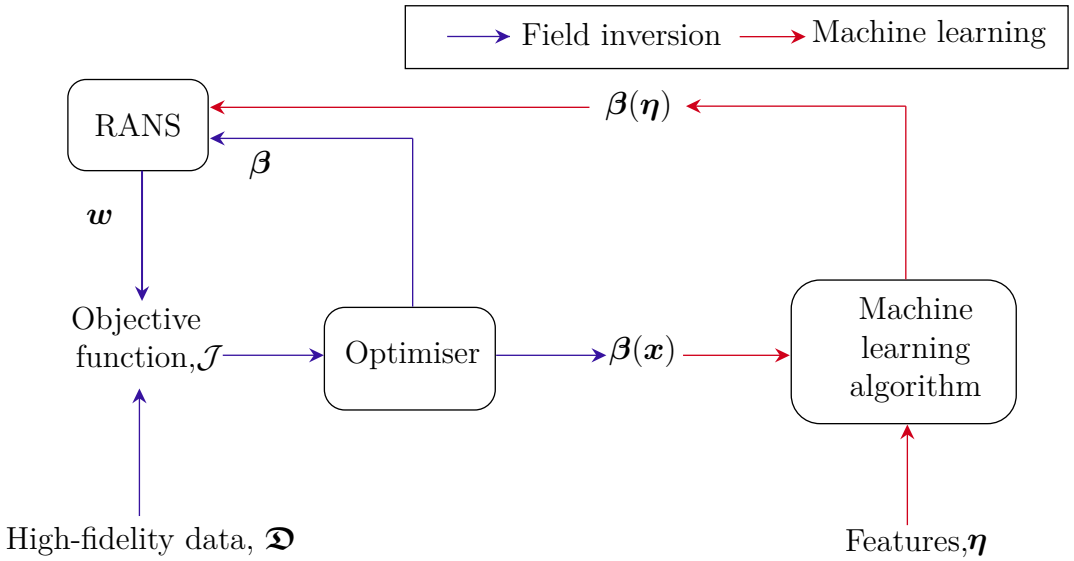


Figure 2-1: Processes involved in the field inversion and machine learning framework, where $\boldsymbol{\beta}$ is the discrepancy field, $\boldsymbol{w}$ is a vector of state variables in the RANS model, and $\boldsymbol{x}$ is the spatial coordinates.

The field inversion phase is the essence of the FIML framework. The goal is to address model-form errors in physical models (i.e. RANS equations) by seeking *field* distributions of variables involved. This is done by solution of inverse problems over a number of scenarios which are sufficiently representative of deficiencies in the physical model. The

inverse problem can be posed as one of optimisation, the mathematical basis of which is the subject of Chapter 3.2. To generalise the knowledge gained from the field inversion for improving predictive capabilities of RANS models, machine learning algorithms are employed to extract a functional relationship for the inferred discrepancies based on a set of flow features available from the RANS model.

The framework was first proposed in [41] where a demonstrative example, emulated since in [5, 43], is provided by deliberately introducing deficiency in an ordinary differential equation (ODE) modelling 1D heat conduction with conductive and radiative terms, and then using FIML to produce results closer to the original ODE. The framework is more thoroughly developed in the PhD project in [5], with applications to more realistic flows involving curvature, adverse pressure gradients, and separation where the S-A and Wilcox $k-\omega$ models are augmented using FIML. Building on this work, [43] investigates the use of continuous adjoints in the field inversion phase comparing it to the discrete adjoint approach in previous works. [42, 27, 44, 45] compare the performance of neural networks and Gaussian processes in the machine learning phase.

Key advantages of this framework are: a relatively low number of inferred fields can be used to train the machine learning algorithm on [26]; relatively spare experimental data such as pressure coefficients can be used to achieve substantial improvements [27]; and the field inversion phase can provide valuable insight to the modeller [28].

2.5 Some contentious issues

Data-driven turbulence modelling is a controversial topic in the research community. Two important metrics for evaluating turbulence models are *universality* – the ability of the model to robustly predict a wide-range of flows without flow-specific tuning – and *interpretability* – the possibility of expressing the model in explicit analytical forms [46].

Regarding the universality criterion, most of the notable data-driven frameworks in literature thus far have only achieved improved predictive capability for flow cases similar to the data set used for training, i.e. same class of flows with different Reynolds numbers or slightly different geometries. At this point it is useful to differentiate between frameworks proposed to augment RANS models, such as FIML, and one where attempts are made to discover RANS closures such as the use of gene expression programming (GEP) to develop algebraic Reynolds stress models [36]. For the former, the universality criterion becomes less critical in that, the FIML framework has the goal of augmenting RANS models for similar flow cases from the outset, [37] being another example of a similar approach.

Concerning the interpretability criterion, only the framework based on GEP has been able to achieve interpretable closures [38, 36] for one class of problems. It is debatable whether the GEP approach can preserve the interpretability requirement if the same model was trained on a range of flow classes without the analytical expression becoming too large for human comprehension, for instance due to the number of mathematical terms involved becoming too large. On a similar note, in [47] (a quote also used to begin this chapter) the authors express 'discomfort' regarding the complexity of mapping involved between inputs and outputs in machine learning algorithms.

The importance of the two criteria in data-driven turbulence modelling is summarised well in [46]: "Ultimately, both universality and interpretability requirements are intimately related to the fundamental question in turbulence modeling: does there exist a universal turbulent constitutive relation? Generations of researchers have labored for many decades

on dozens of turbulence models, yet none of them achieved predictive generality, which seems to indicate that the answer is "no". If so, then flow-specific tuning and fudge factors would be inevitable if good predictive performances are desirable. The machine learning based turbulence models can be considered automatic, flow-specific tuning schemes based on the flow regime to be predicted and the flow regimes that present in the training database."

Finally, another limitation of some data-driven approaches is the amount and fidelity of data required for training the models. Approaches in [37] and [36] require detailed high-fidelity data for training which can be limiting for complex flow cases where such data may not exist, is expensive or even computationally intractable to acquire. The FIML approach although not able to achieve interpretable models yet, does not suffer the high-fidelity data requirement.[1]

2.6 Summary

The contents of the chapter are summarised as follows:

- The state-of-the-art in data-driven approaches to turbulence modelling were briefly outlined.
- It was highlighted that proposed frameworks that address the functional errors in RANS closure seem to have more potential for augmenting turbulence models compared to approaches for model parameter calibration.
- Some contentious issues in the turbulence modelling community regarding machine-learning based model interpretability, universality and the amount of high-fidelity data required for some frameworks were highlighted.
- Overall, the jury is still out on the potential for data-driven modelling introducing a paradigm-shift in turbulence modelling.

[1] Based on personal correspondence, research work is ongoing to overcome the high-fidelity data requirement in the GEP approach in [36], by looking into the synergy of obtaining interpretable closures (original GEP approach) and using some form of statistical inference (FIML approach). I (refraining from using the phrase 'the author' self-consciously!) will also be investigating this for my upcoming PhD dissertation in collaboration with the original authors.

"Everything should be made as simple as possible, but no simpler."
Albert Einstein

3 Theory

The first part of this chapter introduces the baseline RANS model. Part two formulates the inverse problem based on two approaches: Bayesian, and deterministic, both of which can be ultimately posed as an optimisation problem requiring efficient derivative computations. To this end, a detailed description of the adjoint approach for gradient computation is provided. In part three the flow physics of multi-element aerofoils are discussed.

3.1 Spalart-Allmaras: baseline model

The baseline RANS model considered for this study is the one-equation Spalart-Allmaras (S-A) model [13]. As outlined in Chapter 1.1.3, the S-A model uses the linear Boussinesq approximation to close the RANS equations, and is one of the most popular models for aerodynamic applications due to its robustness. The model can be expressed as follows:

$$\nabla \cdot (\boldsymbol{U}\tilde{\nu}) = \underbrace{C_{b1}\tilde{\Omega}\tilde{\nu}}_{\text{production}} + \underbrace{\frac{1}{\sigma}\left\{\nabla \cdot \left[(\nu + \tilde{\nu})\nabla\tilde{\nu}\right] + C_{b2}|\nabla\tilde{\nu}|^2\right\}}_{\text{transport}} - \underbrace{C_{w1}f_w\left(\frac{\tilde{\nu}}{d}\right)^2}_{\text{destruction}}, \tag{3.1}$$

where the turbulent viscosity ν_t can be related to the surrogate viscosity $\tilde{\nu}$ via, $\nu_t = \tilde{\nu}\chi^3/(\chi^3 + C_{v1}^3)$, and $\chi = \tilde{\nu}/\nu$. For details of the parameters and constants refer to [13].

The choice of S-A model is based on its ease of implementation, and the possibility of using it without the need for wall functions given appropriate y^+ values [48]. For comparison, a limited number of baseline RANS simulations are also performed using the $k-\omega$ shear stress transport (SST) model, which provides similar predictions as will be discussed in Chapter 5.

3.2 Field inversion

As outlined in Chapter 2.4 the first phase of FIML framework is to solve a number of inverse problems (i.e. field inversion) to construct the discrepancy field based on the baseline RANS model and experimental data. This section explores two approaches to formulating the field inversion problem. Note that although the following considers S-A as the baseline model, the theory is equally application to other RANS models (see [5, 43] for example).

Consider the baseline S-A model (Eqn. 3.1) expressed in a general form as,

$$\frac{D\tilde{\nu}}{Dt} = \mathcal{P}\left(\tilde{\nu}, \boldsymbol{w}\right) + \mathcal{T}\left(\tilde{\nu}, \boldsymbol{w}\right) - \mathcal{D}\left(\tilde{\nu}, \boldsymbol{w}\right), \tag{3.2}$$

where $\tilde{\nu}$ is the surrogate viscosity term in the S-A model, $\boldsymbol{w}$ can be any of the Reynolds-average conserved flow variable, and the three right-hand side terms represent the production, transport, and destruction terms respectively.

The goal of the field inversion phase is to construct a functional correction to the baseline model in Eqn. 3.2, which is done by introducing a spatially varying term $\boldsymbol{\beta}(\boldsymbol{x}) \in \mathbb{R}^{n_\beta}$ with n_β representing the number of mesh cells, as a multiplier to the production term, where $\boldsymbol{x}$ represent the spatial variables, as follows,

$$\frac{D\tilde{\nu}}{Dt} = \boldsymbol{\beta}(\boldsymbol{x})\mathcal{P}\left(\tilde{\nu}, \boldsymbol{w}\right) + \mathcal{T}\left(\tilde{\nu}, \boldsymbol{w}\right) - \mathcal{D}\left(\tilde{\nu}, \boldsymbol{w}\right). \tag{3.3}$$

Note: i) that the corrective term can be spatio-temporal, however is simplified in this study as only a function of space since all flows will be steady; ii) $\boldsymbol{\beta}(\boldsymbol{x})$ changes the balance of Eqn. 3.3 entirely, and does not simply modify the production term [5].

The discrepancy field can be used to gain physical insight given the right setting. The caveat of the right setting can be most straightforwardly explained when the opposite is the case, i.e. when it is *not* possible to gain physical insight. An example is the work in this study, i.e. two-dimensional flow over an aerofoil where experimental data is to be used in order to construct a model that can predict outputs which agree with the data. In physical terms however, turbulence is famously a three-dimensional phenomenon, reducing the discrepancy field to a mere mathematical construct whose purpose is to improve predictive capabilities of a RANS model [5] and potentially identify case-specific flow regions where the baseline RANS model presents highest uncertainty [28].

3.2.1 Bayesian formulation

One approach to formulate the inverse problem where the objective is to infer the corrective field $\boldsymbol{\beta}(\boldsymbol{x})$ is the Bayesian inference. This allows for a rigorous problem formulation in which all the quantities are treated as random variables which are statistically incorporated in their probability distributions. In this setting, the inverse problem solution is the application of Bayes' theorem to find the probability distribution of the random quantity of interest, i.e. $\boldsymbol{\beta}(\boldsymbol{x})$ in our case, which is formally called the *posterior* distribution [49].

On an implementational level, solving the Bayesian inverse problem involves the use of sampling algorithms, Markov chain Monte Carlo (MCMC) methods being a popular category [50]. A typical sampling process involves i) drawing samples from the prior distribution, ii) evaluating the likelihood, and iii) using the likelihood to update the posterior distribution. High dimensionality of the problems involved for CFD analyses generally means that seeking solutions to a Bayesian inverse problem can be computationally prohibitive [51], and simplifying assumptions must be made to reduce the computational cost.

If it is assumed that the prior and likelihood distributions are Gaussian, then the posterior distribution will also be Gaussian. In this case the solution of the Bayesian problem reduces to one of approximating the *maximum a posteriori* (MAP) solution [52], which can be formulated as an optimisation problem with the following objective function,

$$\boldsymbol{\beta}_{MAP} = \arg\min_{\boldsymbol{\beta}} \frac{1}{2}\left[\left(\mathfrak{D} - \boldsymbol{h}\left(\boldsymbol{\beta}\right)\right)^T \boldsymbol{C}_{obs}^{-1}\left(\mathfrak{D} - \boldsymbol{h}\left(\boldsymbol{\beta}\right)\right) + \left(\boldsymbol{\beta} - \boldsymbol{\beta}_{prior}\right)^T \boldsymbol{C}_{prior}^{-1}\left(\boldsymbol{\beta} - \boldsymbol{\beta}_{prior}\right)\right], \tag{3.4}$$

where $\boldsymbol{\beta}$ is quantity of interest being optimised at every mesh point in the flow domain, $\mathfrak{D}$ is the data set, $\boldsymbol{h}(\boldsymbol{\beta})$ is the likelihood, $\boldsymbol{\beta}_{prior}$ is the prior distribution of $\boldsymbol{\beta}$, and $\boldsymbol{C}_{obs}$ and

$\boldsymbol{C}_{prior}$ are the covariance matrices for the observations and prior, respectively. Detailed derivations of Eqn. 3.4 can be found in [53], and the solution process in the context of FIML in [41].

3.2.2 Deterministic formulation

Assumptions of Gaussian distributions may be rather simplistic, as turbulence models involves highly non-linear systems. Thus, a deterministic inverse problem formulation can be adopted where the goal is to minimise some discrepancy between the baseline RANS output and experimental data. The optimisation problem in this setting is formulated as follows [5]:

$$\begin{aligned} \boldsymbol{\beta}_{inverse} &= \arg\min_{\boldsymbol{\beta}} \frac{1}{2} \left(\mathcal{J}_1 \left(\boldsymbol{\beta} \right) + \lambda \mathcal{J}_2 \left(\boldsymbol{\beta} \right) \right), \\ &= \arg\min_{\boldsymbol{\beta}} \frac{1}{2} \left[\left(\mathfrak{D} - \boldsymbol{h} \left(\boldsymbol{\beta} \right) \right)^T \left(\mathfrak{D} - \boldsymbol{h} \left(\boldsymbol{\beta} \right) \right) + \lambda \left(\boldsymbol{\beta} - \boldsymbol{\beta}_{prior} \right)^T \left(\boldsymbol{\beta} - \boldsymbol{\beta}_{prior} \right) \right]. \end{aligned} \tag{3.5}$$

The second term in 3.5 is a regularisation term, where λ is the regularisation constant. The formulation would be ill-posed without the regularisation term because of the noisy data and the high degree of freedom in the model compared to the number of data points. The chosen regularisation term is an instance of the Tikhonov regularisation as used in [5, 27], which biases the solution to lie close to the initial solution (prior).

The regularisation constant can be chosen from estimating the confidence in the observational data and the model. Without such data, [5] suggests choosing the regularisation constant λ through the use of an L-curve, Figure 3-1.

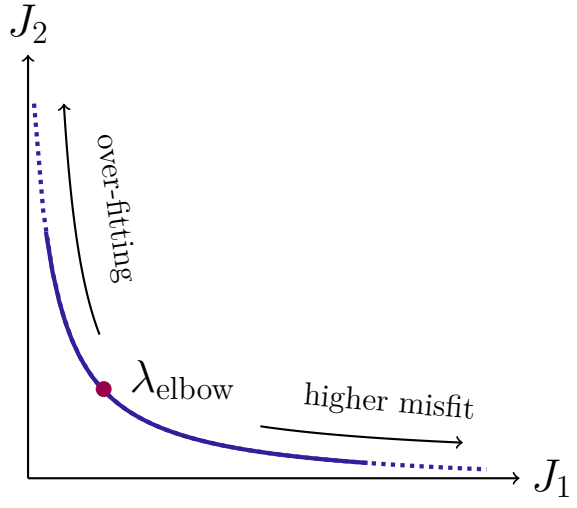


Figure 3-1: L-curve for choosing the 'right' regularisation constant λ for the inverse problems.

Plotting the L-curve involves the solution of a number of inverse problems with different λ values and plotting the two parts in Eqn. 3.5. Lower values of λ will lead to over-fitting, while higher values results in higher misfit. Thus, the λ_{elbow} value can be selected to strike a balance between over-fitting and the discrepancy.

Finally, solving the optimisation problem with the experimental data available in the form of lift coefficients, the objective function reduces to,

$$\mathcal{J} = \left[C_{l,\text{expt}} - C_l\left(\boldsymbol{\beta}\right)\right]^2 + \lambda \sum_{j=1}^{n_\beta} \left[\boldsymbol{\beta}_j - \boldsymbol{\beta}_{j,\text{prior}}\right]^2, \tag{3.6}$$

where $C_{l,\text{expt}}$ and $C_l(\boldsymbol{\beta})$ are the experimental and baseline RANS lift coefficient respectively, and $\boldsymbol{\beta}$ is the corrective field (design variable) with an initial value of unity at every mesh point. For all the simulations the prior $\boldsymbol{\beta}_{\text{prior}}$ is assumed to be unity so that it gives the baseline model. In other word, $\boldsymbol{\beta}$ has an initial value of 1 that will change during the optimisation, while $\boldsymbol{\beta}_{\text{prior}}$ will always remain 1.

3.3 Optimisation Problem

Broadly there are two types of optimisation methods: gradient-free and gradient-based. Gradient-free methods borrow analogues from natural mechanisms, or heuristics. Genetic algorithms are one popular class of gradient-free methods inspired from Darwin's theory of evolution, where the optimisation problem is parameterized into a set of *genes*, and *fitness functions* are used to evaluate the solution domain. Advantages include the capability to work with continuous and discrete design variables, and handling noisy objective functions with multiple local minima. However, a limitation that makes it less appealing for the problem at hand is the significant computational cost of fitness functions evaluations due to the large number of design variables [54].

As the name suggests, gradient-based optimisation methods employ the gradient (usually an approximation) of the objective function(s) with respect to design variable(s) to determine a search direction in a bid to find the minimum. The gradient computation requires the objective function(s) to be sufficiently smooth. Convergence is only guaranteed if a single global minimum exists, or the initial guess is adequately close to the global minimum with an appropriate step size [55]. A key advantage of the gradient-based optimisation algorithms is that the computational cost can be virtually independent of the number of design variables provided that the gradients are computed efficiently, which leads to the adjoint method.

The adjoint method leverages the chain rule to decouple the computation of the derivative of the objective function from the sensitives of the primal variables [56], derived in Section 3.3.1. In this way the derivatives can be evaluated at a cost significantly lower than a more naive direct computation such as the finite difference. Two main options are available in the adjoint method: continuous and discrete adjoint. In the former the adjoint equations for the governing equations are derived in a continuous form, then discretised. In the latter, the adjoint equations are derived for the discretised governing equations from the outset.

Advantages of the continuous adjoint approach include simpler code and less computational memory requirement, however, derivations of the adjoint equations are complex and error-prone. Discrete adjoint approach has the advantage of giving an exact gradient for the discretised objective function, which means that when optimising the derivative information is consistent with the objective function evaluations. Also, unlike the continuous approach, the adjoint equations for the discrete method can be derived using automatic differentiation. Ultimately, in the context of field inversion it has been shown that the choice between discrete and continuous adjoint is implementational, and the end result

from both are similar [43]. To this end, a discrete approach is chosen for this work and implemented using the DAFoam module especially developed for the OpenFOAM CFD package, discussed in detail in Chapter 4.

3.3.1 Discrete adjoint equations

The derivations here are based on the theory presented in [56]. The goal is to compute the derivative $\mathrm{d}\mathcal{J}/\mathrm{d}\boldsymbol{\beta}$ efficiently, where the scalar $\mathcal{J}$ is the objective function, and $\boldsymbol{\beta} \in \mathbb{R}^{n_\beta}$ is the vector of design variables. As mentioned earlier it is assumed that the governing equations are available in a discretised form, and the discrete residual equations $\boldsymbol{\mathcal{R}}(\boldsymbol{\beta}, \boldsymbol{w}) = 0$, where $\boldsymbol{w} \in \mathbb{R}^{n_w}$ is the vector of state variables and $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{n_w}$ is the residual vector, is satisfied. The cost function is now a function of both the design variables, and the state variables, i.e. $\mathcal{J} = f(\boldsymbol{\beta}, \boldsymbol{w})$.

Using the chain rule the derivative $\mathrm{d}\mathcal{J}/\mathrm{d}\boldsymbol{\beta}$ can be expressed as,

$$\underbrace{\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\boldsymbol{\beta}}}_{1 \times n_\beta} = \underbrace{\frac{\partial \mathcal{J}}{\partial \boldsymbol{\beta}}}_{1 \times n_\beta} + \underbrace{\frac{\partial \mathcal{J}}{\partial \boldsymbol{w}}}_{1 \times n_w} \underbrace{\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\boldsymbol{\beta}}}_{n_w \times n_\beta}, \tag{3.7}$$

where the computational cost of computing the partial derivatives $\partial\mathcal{J}/\partial\boldsymbol{\beta}$ and $\partial\mathcal{J}/\partial\boldsymbol{w}$ is relatively cheap as these only involve explicit computations. However, the total derivative matrix $\mathrm{d}\boldsymbol{w}/\mathrm{d}\boldsymbol{\beta}$ must be implicitly determined by the residual equations $\boldsymbol{\mathcal{R}}(\boldsymbol{\beta}, \boldsymbol{w}) = 0$, and is thus computationally expensive. To compute the total derivative $\mathrm{d}\boldsymbol{w}/\mathrm{d}\boldsymbol{\beta}$, the chain rule can be applied for $\boldsymbol{\mathcal{R}}$, noting that $\mathrm{d}\boldsymbol{\mathcal{R}}/\mathrm{d}\boldsymbol{\beta}$ must equal zero in order for the governing equations to satisfy $\boldsymbol{\mathcal{R}}(\boldsymbol{\beta}, \boldsymbol{w}) = 0$:

$$\frac{\mathrm{d}\boldsymbol{\mathcal{R}}}{\mathrm{d}\boldsymbol{\beta}} = \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{\beta}} + \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{w}} \frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\boldsymbol{\beta}} = 0, \tag{3.8}$$

which can be expressed as the following linear system

$$\frac{\mathrm{d}\boldsymbol{w}}{\mathrm{d}\boldsymbol{\beta}} = -\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{w}}^{-1} \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{\beta}}. \tag{3.9}$$

The expression for $\mathrm{d}\boldsymbol{w}/\mathrm{d}\boldsymbol{\beta}$ is now substituted in Eqn. 3.8 to get the following:

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\boldsymbol{\beta}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\beta}} - \underbrace{\frac{\partial \mathcal{J}}{\partial \boldsymbol{w}} \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{w}}^{-1}}_{\boldsymbol{\psi}^T} \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{\beta}}, \tag{3.10}$$

where $\boldsymbol{\psi}$ is the adjoint vector. Transposing the state Jacobian matrix $\partial\boldsymbol{\mathcal{R}}/\partial\boldsymbol{w}$ and solving with $[\partial\mathcal{J}/\partial\boldsymbol{w}]^T$ as the right-hand side of Eqn. 3.10 yield the adjoint equation,

$$\frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{w}} \boldsymbol{\psi} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{w}}^T. \tag{3.11}$$

Having solved Eqn. 3.11, the total derivative $\mathrm{d}\mathcal{J}/\mathrm{d}\boldsymbol{\beta}$ can now be computed by substituting the adjoint vector $\boldsymbol{\psi}$ into Eqn. 3.10 as follows:

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\boldsymbol{\beta}} = \frac{\partial \mathcal{J}}{\partial \boldsymbol{\beta}} - \boldsymbol{\psi}^T \frac{\partial \boldsymbol{\mathcal{R}}}{\partial \boldsymbol{\beta}}. \tag{3.12}$$

The efficiency of the adjoint approach can now be illustrated using the fact that for each computation of the objective function, the adjoint equation is only solved once since the design variable is not explicitly present in Eqn. 3.11. In other words, the computational cost is proportional to the number of objective functions, e.g. one for our purposes, and is independent of the number of design variables, e.g. the number of mesh cells, $\mathcal{O}(10^5)$ for our purposes.

3.4 Multi-element configurations flow physics

The flow field around a multi-element configuration involves complex flow phenomena as illustrated in Figure 3-2. Some of the key aspects of the flow are wakes of each element, recirculation regions, strongly curved boundary layers, relaminarisation, local compressible flows even at low Mach number regimes, and possible separation.

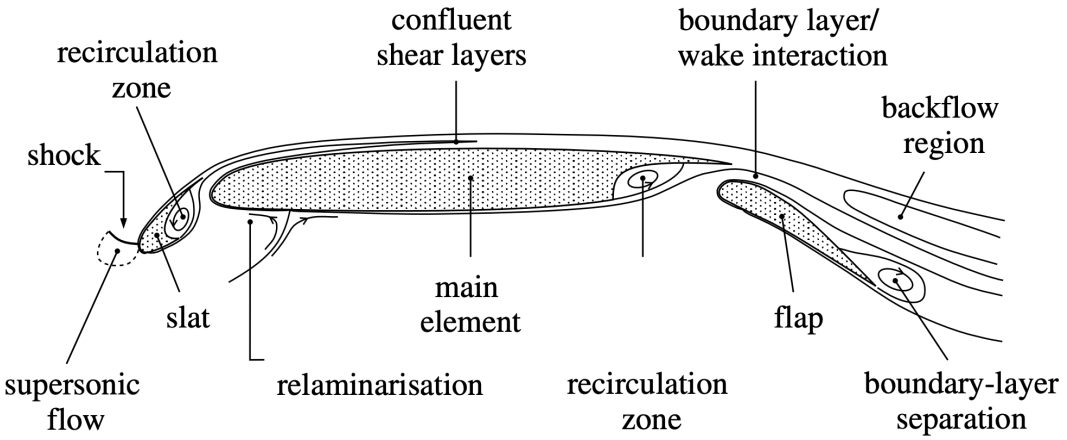


Figure 3-2: Some of the complex flow phenomena around a typical multi-element high-lift configuration, from [57].

One of the classic[1] descriptions of multi-element aerofoil flow physics is in the seminal work by A. M. O. Smith, [58], where the effects of the gaps between the elements are attributed to five main phenomena.

I. Slat effect: Consider two elements, e.g. the slat and main element in Figure 3-2. The circulation at the trailing edge (TE) of the slat induces a velocity at the leading edge (LE) of the main element. This induced velocity is counter to the natural acceleration the LE of the main aerofoil would see without the presence of the slat. The effect of this induced velocity is to reduce the suction peak at the LE of the main element, thus reducing the requirement for pressure recovery, hence delaying flow separation.

II. Circulation effect: The main element in turn causes a circulation at the TE of the slat. This increases the loading on the slat, thus increasing the lift. However, this also increases the pressure recovery demands.

III. Dumping effect: The increased pressure recovery demand is alleviated by the high velocity flow on the upper surface of the main element, allowing the flow at the slat to leave at a higher velocity.

[1] based on how often the paper is referred to in papers on this topic

IV. Off-the-surface pressure recovery: this idea is based on the wake of an aerofoil element, where it has been experimentally shown that when a boundary layer has left the surface, i.e. the boundary layer has become a wake, it can sustain higher adverse pressure gradients. The phenomenon allows faster pressure recovery without the adverse effects of separation of the boundary layer on the surface.

V. Fresh-boundary-layer effect: a boundary layer grows as it move over a surface. The introduction of multiple elements means that the surfaces on which the boundary layer will grow is 'broken up' allowing 'fresh' boundary layers to form on each surface. This allows each fresh boundary layer to remain relatively thin, hence, reducing adverse pressure gradients which can lead to flow separation.

3.5 Summary

The contents of the chapter are summarised as follows:

- It was shown that using data to infer spatial distribution of the discrepancy in RANS models by solving numerous inverse problems is the essence of field inversion and machine learning framework.
- The statistical inference problem was formulated in a Bayesian and deterministic framework. While the former is more systematic, it requires specific data that may not be available and more importantly, requires excessive computational power. Both are ultimately posed as an optimisation problem.
- The choice of gradient-based optimisation was justified due to the high dimensionality of the optimisation problem, and to this end the discrete adjoint equations were derived for efficient gradient computations.
- Finally, the flow physics for multi-element configurations were discussed.

"It does not matter how slowly you go as long as you do not stop."
Confucius

4 Implementation

The first part of the chapter briefly introduces OpenFOAM for solving the S-A RANS model. The second part introduces DAFoam for solving the adjoint equations, also detailing the additions to implement the FIML case. Finally, part three details the aerofoil configurations identified to perform the inverse problems, the meshes for one demonstrative case, and details of the experimental data identified for all three.

4.1 OpenFOAM

Open field operation and manipulation (OpenFOAM) is a popular open-source object-oriented CFD software. Written in C++, it is capable of solving various complex flows with more than 80 solvers [59]. For this work one of the primal solvers, `simpleFoam` is used, which is an incompressible solver with the following governing equations,

$$\nabla \cdot \boldsymbol{U} = 0, \tag{4.1}$$

$$\nabla \cdot (\boldsymbol{U}\boldsymbol{U}) + \nabla p - \nu_{\text{eff}} \nabla \cdot \left(\nabla \boldsymbol{U} + \nabla \boldsymbol{U}^T\right) = 0, \tag{4.2}$$

where U is the velocity vector, p is the pressure, $\nu_{\text{eff}} = \nu + \nu_t$ is the effective viscosity with ν and ν_t representing the molecular and turbulent kinematic viscosity, respectively. `simpleFoam` is so called as the continuity and momentum equations are coupled using the semi-implicit method for pressure-linked equations (SIMPLE) algorithm [60].

The variant of the S-A model in OpenFOAM ignores the f_{t2} term, and the f_{v3} function is added to enhance the stability by ensuring non-negative $\tilde{\Omega}$ term.

4.2 DAFoam

DAFoam[1] is an open-source library specifically designed to couple with OpenFOAM for discrete adjoint development of its steady-state primal solvers [61, 62]. It computes the partial derivatives using a finite difference method using the graph-colouring technique for efficiency, and solves the adjoint equations using a Krylov method. Proposed with multidisciplinary design optimisation in mind, DAFoam offers the capability of shape optimisation using a high-level Python interface by employing other external libraries and modules. These modules are used for controlling the geometric shape, mesh deformation, and a Python module to link with optimisation packages. In our context, although geometry control and mesh deformation are not required, the optimisation interface could prove useful for the optimisation phase of the field inversion.

[1] https://dafoam.readthedocs.io

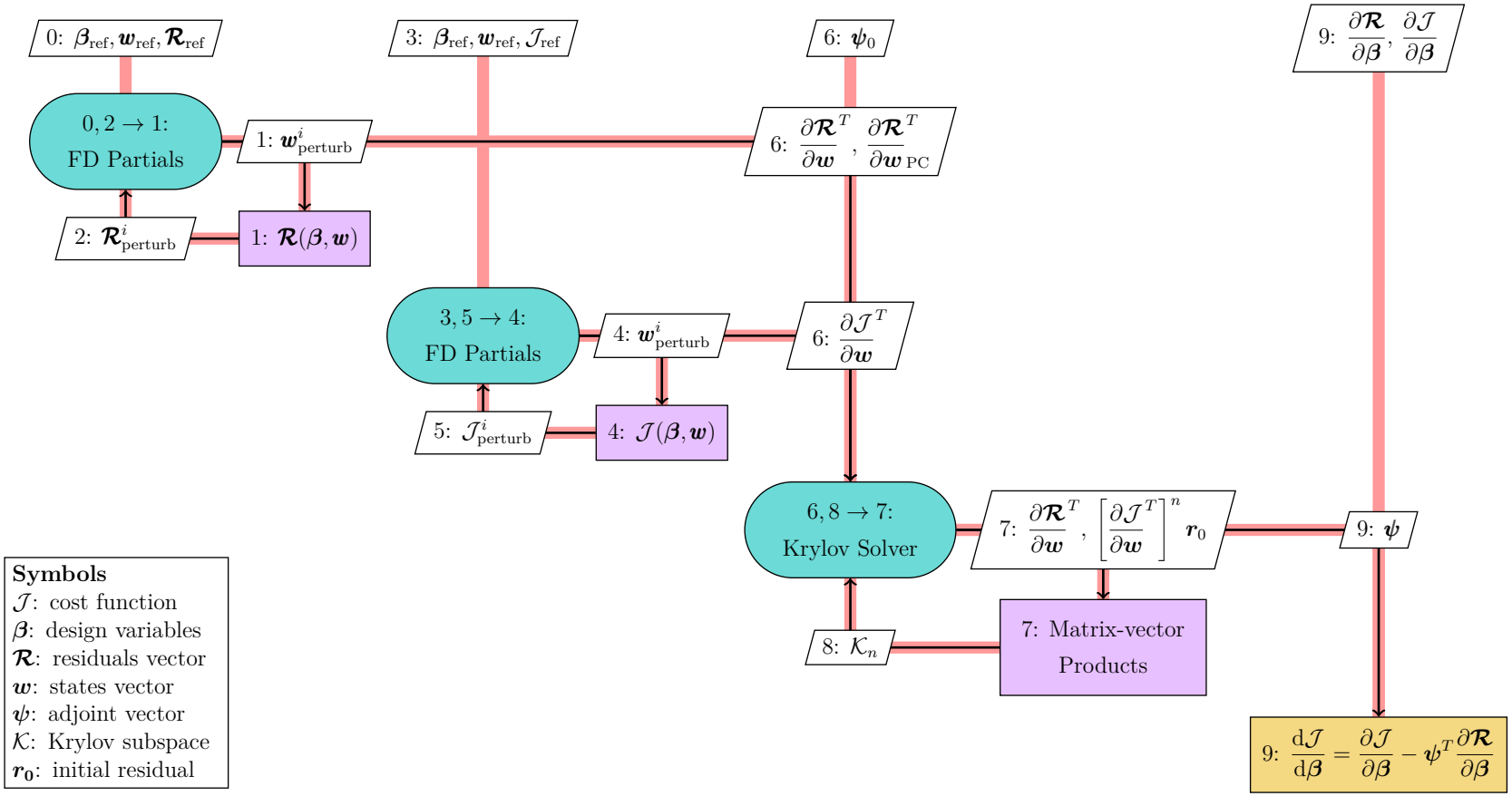


Figure 4-1: Process and data flow for the adjoint approach in DAFoam using the extended design structure matrix (XDSM) proposed in [63]. Modules are represented by the diagonal nodes, while data is represented by the off-diagonal. Black line represent the process flow, while the thick red lines represent the data flow. The execution order is represented by the number in each node, and the superscripts i and n represent the ith colour, and nth iteration respectively. Generated using `pyXDSM`: https://github.com/mdolab/pyXDSM.

4.2.1 Overview of the adjoint computation process

The discrete adjoint implementation in DAFoam will be described using the extended design structure matrix (XDSM) proposed in [63], Figure 4-1, and the descriptions follow [62, 61]. Modules are represented by the diagonal nodes, while data is represented by the off-diagonal. Data is input in horizontal direction for each module, and output in vertical direction. The thick red lines represent the data flow, while the black lines represent the process flow. The execution order is represented by the number in each node.

Finite-difference approximates of the partial derivatives

The following partial derivatives need to be solved for the adjoint implementation as derived in Chapter 3.3.1: $\partial \mathcal{J}/\partial \boldsymbol{\beta}$, $\partial \boldsymbol{\mathcal{R}}/\partial \boldsymbol{\beta}$, $\partial \boldsymbol{\mathcal{R}}/\partial \boldsymbol{w}$, and $\partial \mathcal{J}/\partial \boldsymbol{w}$. In DAFoam the partial derivatives are approximated using the finite differences (FD) method. Given a Jacobian matrix $\partial \boldsymbol{Y}/\partial \boldsymbol{X}$, the finite-difference approximation is as follows [62]:

$$\left(\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{X}}\right)_{i,j} \approx \frac{Y_i(\boldsymbol{X} + \epsilon \boldsymbol{e}_j) - Y_i(\boldsymbol{X})}{\epsilon}, \tag{4.3}$$

where i and j are for the row and column indices of the matrix, respectively, ϵ is the step size, and $\boldsymbol{e}_j$ is a unit vector with a value of 1 in row j and 0 in all other rows.

In order to accelerate the FD Jacobian computations, DAFoam employs a graph-colouring method [64] which exploits the sparsity of the matrices. The graph-colouring techniques is as follows: i) making different structurally orthogonal subgroups (colours) by partitioning the columns of a Jacobian matrix such that, in each subgroup, no two columns have a non-zero entry in a common row; ii) simultaneously perturbing multiple columns with the same colour to compute the derivative. The point is that in this way no two columns will affect the same row, thus the computations involve calling the routines for computing the residual and objective only once.

The graph-colouring method is only applied for the expensive Jacobian computations, $[\partial \boldsymbol{\mathcal{R}}/\partial \boldsymbol{w}]^T$ (process 0-1-2-0 in Figure 4.2), and $[\partial \mathcal{J}/\partial \boldsymbol{w}]^T$ (process 3-4-5-3 in Figure 4.2). For details refer to [61]. The partial derivatives $\partial \boldsymbol{\mathcal{R}}/\partial \boldsymbol{\beta}$ and $\partial \mathcal{J}/\partial \boldsymbol{\beta}$ are computed using a brute-force finite-difference approach as the computations for these are deemed computationally less intensive.

A limitation of the FD approach is that it is prone to truncation and cancellation errors, and for functions exhibiting strong non-linearity the values of the derivatives are sensitive to the step size ϵ. However, due to its ease of implementation and the minimal modifications required to the original OpenFOAM code, it is still used. To alleviate the finite-difference errors it is proposed to choose an appropriate step size by experimenting with different values of ϵ. In [62], it is shown that the FD approach can produce reasonable results with satisfactory computational resource requirements. The alternatives to the FD approach are analytical methods, the complex-step method, and automatic differentiation which are discussed in [56].

Solving the adjoint equations using the Krylov method

Once the Jacobian computations for $[\partial \boldsymbol{\mathcal{R}}/\partial \boldsymbol{w}]^T$ and $[\partial \mathcal{J}/\partial \boldsymbol{w}]^T$ are complete, the adjoint vector $\boldsymbol{\psi}$ must be calculated using Eqn. 3.11, repeated here for clarity: $[\partial \boldsymbol{\mathcal{R}}/\partial \boldsymbol{w}]^T \boldsymbol{\psi} = [\partial \mathcal{J}/\partial \boldsymbol{w}]^T$. This is the 6-7-8-6 process in Figure 4.2.

DAFoam uses the portable, extensible toolkit for scientific computing (PETSc) [65] library to solve the equation, using the generalised minimal residual (GMRES) iterative linear equation solver [66]. The GMRES algorithm is based on the Krylov subspace, $\mathcal{K}_i = \text{span}(r_0, Ar_0, \ldots, A^{i-1}r_0)$, where $\boldsymbol{A} = [\partial \boldsymbol{\mathcal{R}}/\partial \boldsymbol{w}]^T$ is the transpose of the state Jacobian matrix, and $\boldsymbol{r}_0 = [\partial \mathcal{J}/\partial \boldsymbol{w}]^T - [\partial \boldsymbol{\mathcal{R}}/\partial \boldsymbol{w}]^T \boldsymbol{\psi}_0$ is the initial residual. The preconditioner matrix $[\partial \boldsymbol{\mathcal{R}}/\partial \boldsymbol{w}]^T_{PC}$ is used to improve the convergence via the approximation of the residuals and their linearisations. For details see [62].

Having computed $\boldsymbol{\psi}$, $\partial \boldsymbol{\mathcal{R}}/\partial \boldsymbol{\beta}$ and $\partial \mathcal{J}/\partial \boldsymbol{\beta}$, the total derivative is finally computed using Eqn. 3.12. This is process 9 in Figure 4.2.

4.2.2 New objective function and adjoint computation setup

For the field inversion, additional routines need to be added to the original DAFoam code. The additional routines are to compute the objective function (Eqn. 3.11), and the partial derivatives $\partial \boldsymbol{\mathcal{R}}/\partial \boldsymbol{\beta}$ and $\partial \mathcal{J}/\partial \boldsymbol{\beta}$. As explained earlier, the two partial derivatives mentioned are computed using the brute-force FD method. Figure 4-2 shows the scripts in DAFoam source directory (`src`) where additional codes are added for this study.

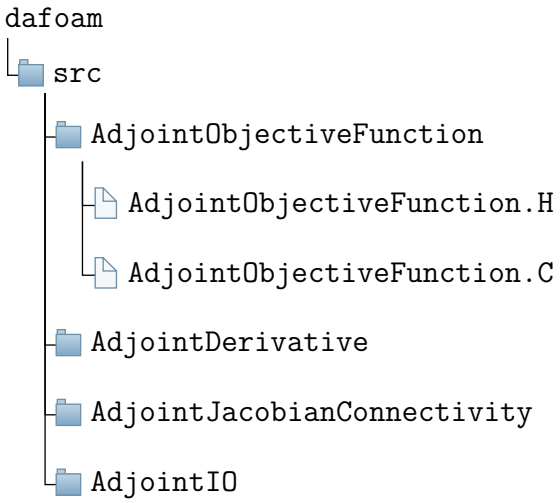


Figure 4-2: Additional codes added to the DAFoam source code to implement the FIML objective function, and compute its adjoint derivatives.

Each child directory under `src` is intuitively named and contains at least two scripts with `.H` and `.C` extensions of the same name as the directory, as illustrated for the `AdjointObjectiveFunction` in Figure 4-2. The files with `.H` extensions are header scripts used to declare classes as a means of checking errors when compiling the codes. This is to ensure the classes used and the operations performed with them actually exist. The classes defined through a set of instruction such as object construction, data storage and class member functions are defined in scripts with `.C` extensions. The additional codes are briefly described below[2]:

- `AdjointObjectiveFunction`: used for computing the objective function (Eqn. 3.11). User-specified inputs include parameters for computing the lift coefficient, the data (i.e. experimental lift coefficient), and the regularisation constant, main code shown in Listing A.1.

[2]A detailed documentation will be written in the near future.

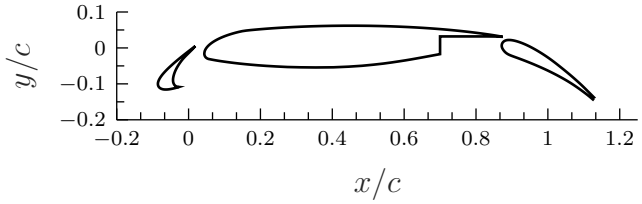
- `AdjointDerivative`: used for computing the partial derivatives $\partial \boldsymbol{\mathcal{R}}/\partial \boldsymbol{\beta}$ and $\partial \mathcal{J}/\partial \boldsymbol{\beta}$, and the total derivative $\mathrm{d}\mathcal{J}/\mathrm{d}\boldsymbol{\beta}$. The partial derivatives are initialised for faster computations, and a separate routine is added to perturb these derivatives for finite-difference approximations, see Listing A.2. Additional functions to compute the partial derivatives $\partial \mathcal{J}/\partial \boldsymbol{w}$ and $\partial \boldsymbol{\mathcal{R}}/\partial \boldsymbol{w}$ are not required as these are computed by the original DAFoam codes.

- `AdjointJacobianConnectivity`: this directory is essential as it used for graph-colouring to accelerate finite difference computations. Here a list is used to specify how many levels of surrounding states vector ($\boldsymbol{w}$) are connected to the residual vector ($\boldsymbol{\mathcal{R}}$) which will be used by the graph-colouring scheme to compute the colours.

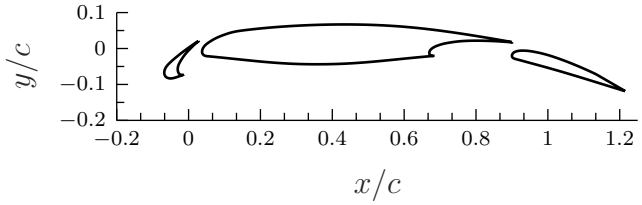
- `AdjointIO`: The additions in this directory deal with the user-specified input-output throughout the code, such as supplying the experimental lift coefficient and the regularisation constant for the objective function computation.

4.3 Aerofoils and experimental data

Recall that the essence of field inversion is to solve inverse problems for a range of geometric configurations and flow conditions in order to then use machine learning algorithms for modelling the discrepancy, and hence augmenting a baseline RANS model. In that spirit this section describes three aerofoils and associated experimental data for the inverse problems.



(a) MDA 30P-30N, $\delta_{\text{slat}} = 30°$, $\delta_{\text{flap}} = 30°$



(b) NHLP L1T2, $\delta_{\text{slat}} = 25°$, $\delta_{\text{flap}} = 20°$

(c) DLR F-15, $\delta_{\text{slat}} = 28°$, $\delta_{\text{flap}} = 35°$ [redacted due to license terms]

Figure 4-3: Multi-element aerofoils identified for this study, where δ_{slat} and δ_{flap} represent the slat and flap deflection, respectively.

Table 4-1: Three-element high-lift aerofoil experimental data identified for solving the inverse problems. M_∞ and Re_c are the freestream Mach number and the Reynolds number based on the chord (typically retracted aerofoil), respectively.

Aerofoil	Flow conditions	Angle of attack	Available data type	Geometry	Ref.
MDA 30P-30N	$M_\infty = 0.2$ $Re_c = \{5 \times 10^6, 9 \times 10^6\}$	$\{4.07°, 8.12°, 16.21°,$ $21.29°, 23.28°\}$	lift and drag coefficients; pressure coefficients distribution; limited skin friction coefficients; velocity profiles	UIUC Applied Aerodynamics Group[a]	[67, 24]
	M_∞: not given $Re_c = 1.71 \times 10^6$	$\{3°, 5.5°, 8.5°\}$	steady and unsteady pressure coefficient distributions; stereoscopic PIV velocity field measurements for slat cove	see above	[68]
NHLP-L1T2	$M_\infty = 0.197$ $Re_c = 3.52 \times 10^6$	$\{4.01°, 20.18°\}$	lift and drag coefficients; pressure coefficients distribution; velocity profiles	ERCOFTAC[b]	[69, 70]
DLR-F15	$M_\infty = \{0.13, 0.15, 0.2\}$ $Re_c = \{4 \times 10^6, 7 \times 10^6\}$	$\{19°, 25°\}$	lift coefficients; pressure coefficients distribution; stall lift coefficients at various Reynolds numbers and temperatures	data acquired under license	[71]

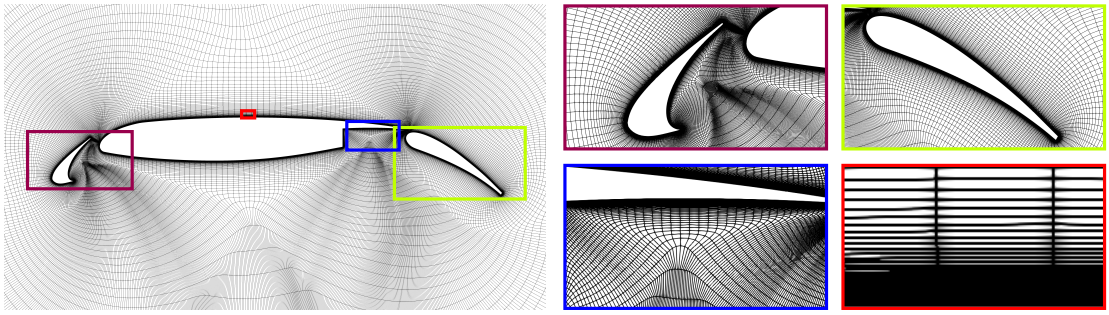
[a] https://m-selig.ae.illinois.edu/ads/coord_updates/30P-30N.dat
[b] http://qnet-ercoftac.cfms.org.uk/w/index.php/Description_AC1-08

Figure 4-3 illustrates the three normalised aerofoils identified, each with different slat and flap deflection angles, different gap sizes between the elements, and different element sizes and profiles.[3] The details of experimental data are presented in Table 4-1.

The McDonnell-Douglas Aerospace (MDA) 30P-30N (Figure 4-3a) has the richest experimental data and is one of the most widely numerically studied aerofoils. This is the aerofoil used for the baseline simulations for the present work. The L1T2 aerofoil (Figure 4-3b) was designed and experimentally studied under the UK's National High-Lift Programme (NHLP) in the 1970s and the data, though publicly available, is relatively sparse. Finally, the third aerofoil F-15 (Figure 4-3c) is from the German Aerospace Centre (DLR). The available data for this is relatively rich. For this work, a three-year data and geometry license (with possibility for extension) has been acquired from the owners.

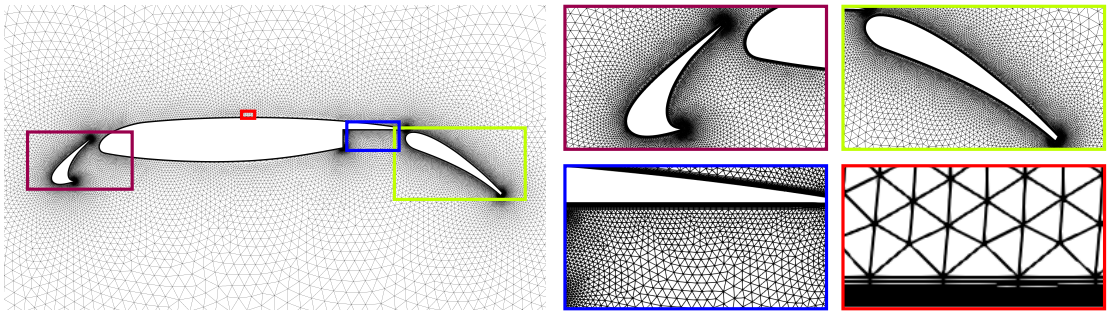
4.4 Baseline CFD setup

The aforementioned OpenFOAM software is used to perform the simulations. As a representative case, the MDA 30P-30N aerofoil is used in the present study with the flow conditions $M_\infty = 0.2$, and $Re_c = 5 \times 10^6$ based on the experimental data from [67]. The simulations assume steady, incompressible flow, and the main turbulence models used is the Spalart-Allmaras, however, a number of simulations are also performed using the $k - \omega$ SST model, results compared in the next chapter.



(a) **(b)**

Figure 4-4: Finest structured mesh used for the 30P-30N aerofoil.



(a) **(b)**

Figure 4-5: Hybrid mesh used for the 30P-30N aerofoil.

[3] To the best of my knowledge, these are the only three-element aerofoils available in literature with substantial enough experimental data for our purposes.

Three structured, Figure 4-4, and a hybrid, Figure 4-5, O-grid type meshes were used, with freestream extending over a hundred chord lengths away from the airfoil surface. The structured meshes were generated using the largely automated hyperbolic extrusion feature in the commercial mesh generation tool, Pointwise. The three levels of mesh density in the structured case were achieved by varying the size of the first element from the walls and the growth rate. The hybrid mesh–structured in the wall regions and unstructured away from the walls–was found online[4] and used for comparison purposes.

Although due to lack of time no mesh sensitivity analysis was undertaken, an attempt was made to generate meshes based on best practices advice, including meshes with y^+ values of approximately 1.0 near solid walls, placing 20 to 30 mesh points up to $y^+ = 100$, and farfield sufficiently faraway from the walls. The y^+ values are presented in Table 4-2. The OpenFOAM boundary conditions applied are listed in Appendix B. Finally, the simulation were run until the residuals for all the variables reduced by at least five orders of magnitude.

Table 4-2: y^+ values of the various meshes used.

Mesh	y^+ min.	y^+ avg.	y^+ max.
Hybrid	0.0052	0.2294	1.1619
Structured - fine	0.0026	0.0422	0.3074
Structured - medium	0.0042	0.2715	1.5288
Structured - coarse	0.1518	2.4502	11.9881

4.5 Summary

The contents of the chapter are summarised as follows:

- The `simpleFOAM` solver in OpenFOAM was introduced, with the additional module DAFoam for implementing the discrete adjoint derivative computations.
- The processes for adjoint computation in DAFoam were described, which employs a graph-colouring technique to accelerate finite-difference based gradient computations.
- Additional codes written to implement the field inversion were described which mainly comprised of routines to compute the objective function, and the various partial derivatives.
- Aerofoils and associated experimental data were identified for inverse problems.
- Finally, the CFD setup for baseline simulations were discussed. Details discussed included flow conditions, structured and hybrid grids used, and boundary conditions.

[4] http://www.pointwise.com/webinars/2014-09/SU2-Pointwise-Joint-Workshop.html

"The test of science is its ability to predict. Had you never visited the earth, would you predict the thunderstorms, the volcanoes, the ocean waves, the auroras, and the colourful sunset?"

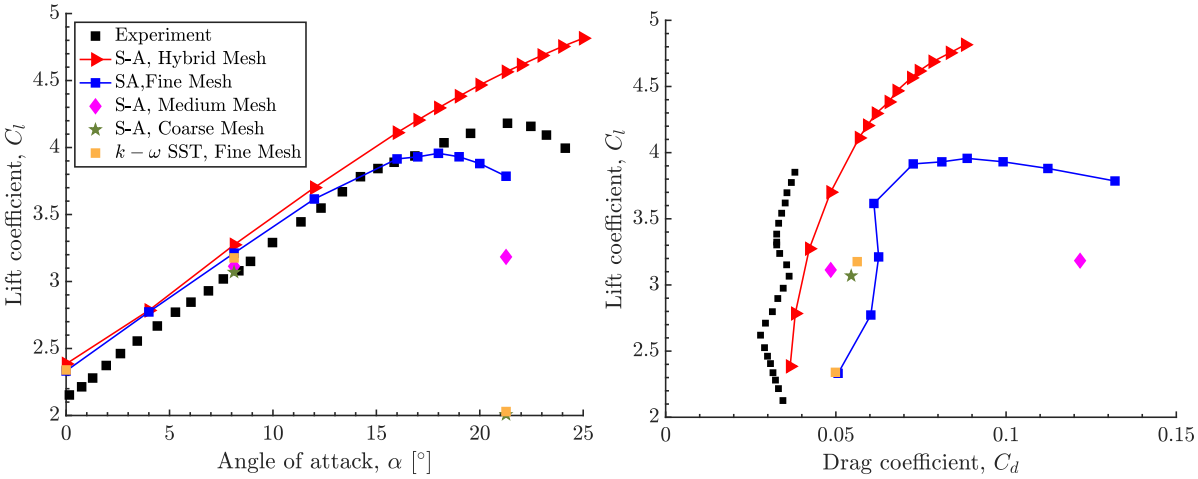
Richard Feynman

5 Results and discussion

5.1 Baseline RANS predictions

Key results illustrating the predictive capabilities of the S-A and $k-\omega$ SST models are presented in this section. The experimental data were digitised from [24] using a MATLAB function, thus may involve uncertainties. For uncertainties due to the experimental setup and limitations refer to [67].

The lift coefficients, Figure 5-1a, are over-predicted by the models for the different meshes in the linear region. Considerable discrepancy exits between the experimental data for the S-A model for different meshes. The hybrid mesh vastly over-predicts the onset of stall, while the structured meshes predict stall at a lower angle of attack, where closest results compared to the experimental data achieved are by the finest structured mesh. The $k-\omega$ SST and S-A models produce similar results for the same mesh at lower angles of attack, however the SST model vastly under-predicts the lift coefficient at the experimental stall angle.



(a) lift coefficient

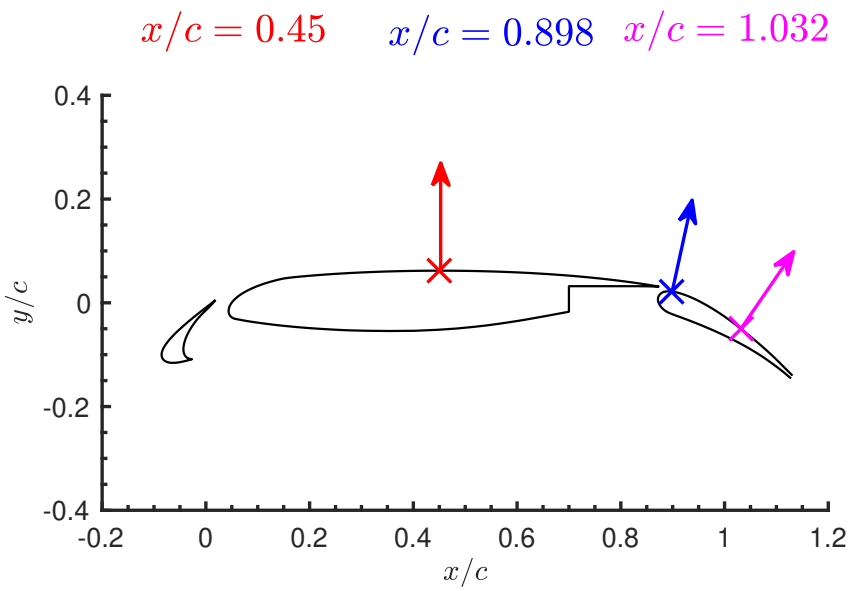
(b) drag coefficient, legend same as **(a)**

Figure 5-1: Aerodynamic force coefficients against angles of attack for the MDA 30P-30N aerofoil at $Re_c = 5 \times 10^6$ and $M_\infty = 0.2$.

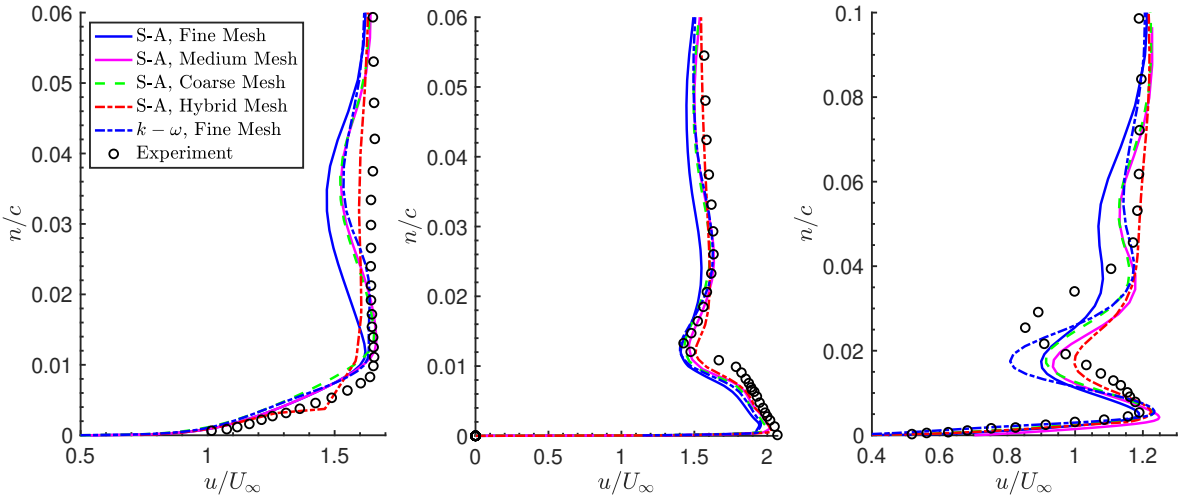
Poorest prediction accuracy is seen in the drag coefficients, Figure 5-1b, where C_d is over-predicted for both turbulence models on all the meshes. The closest agreement to experimental data is achieved by the hybrid mesh with the S-A model.

The velocity profiles give important insight about the turbulent flow phenomenon in boundary layer, shown in Figure 5-2. Figure 5-2a can be used to visualise the positions along the chord where the velocity profiles are plotted. The structured meshes over-predict the boundary layer and wake interaction from the slat on the main element as illustrated in

Figure 5-2b. The overall prediction quality is reasonably good towards the leading edge of the flap, Figure 5-2c. The interaction between the boundary layer and wakes from the slat and main element on the flap is shown in Figure 5-2d, where both the models and meshes show high variability in the velocity magnitude. The position where wake interaction dominates the velocity profile is similar for all the simulations, but under-predicated compared to the experimental data.



(a) Velocity profiles locations



(b) $x/c = 0.45$ **(c)** $x/c = 0.898$ **(d)** $x/c = 1.032$

Figure 5-2: Comparison of the velocity profiles at various points along the airfoil upper surface, at $\alpha = 8.12°$.

The pressure distribution on the airfoil at $\alpha = 8.12°$ is shown in Figure 5-3. Firstly, it is clear that the simulation results show good agreement to experimental data. Secondly, in terms of flow behaviour it shows that the distribution on the pressure side is relatively uniform, while the suction side has variability where the pressures are high at the LE of each element, decreasing and reaching relatively uniform levels as the flow approaches the trailing edges for the slat and main element.

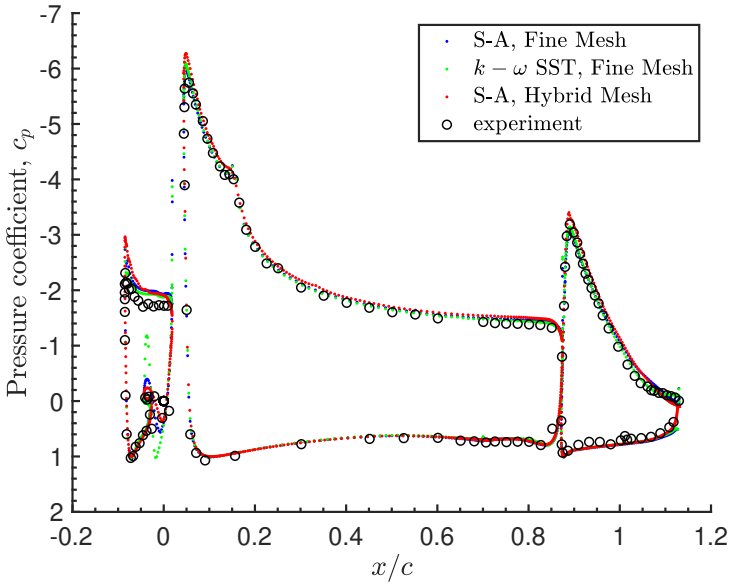
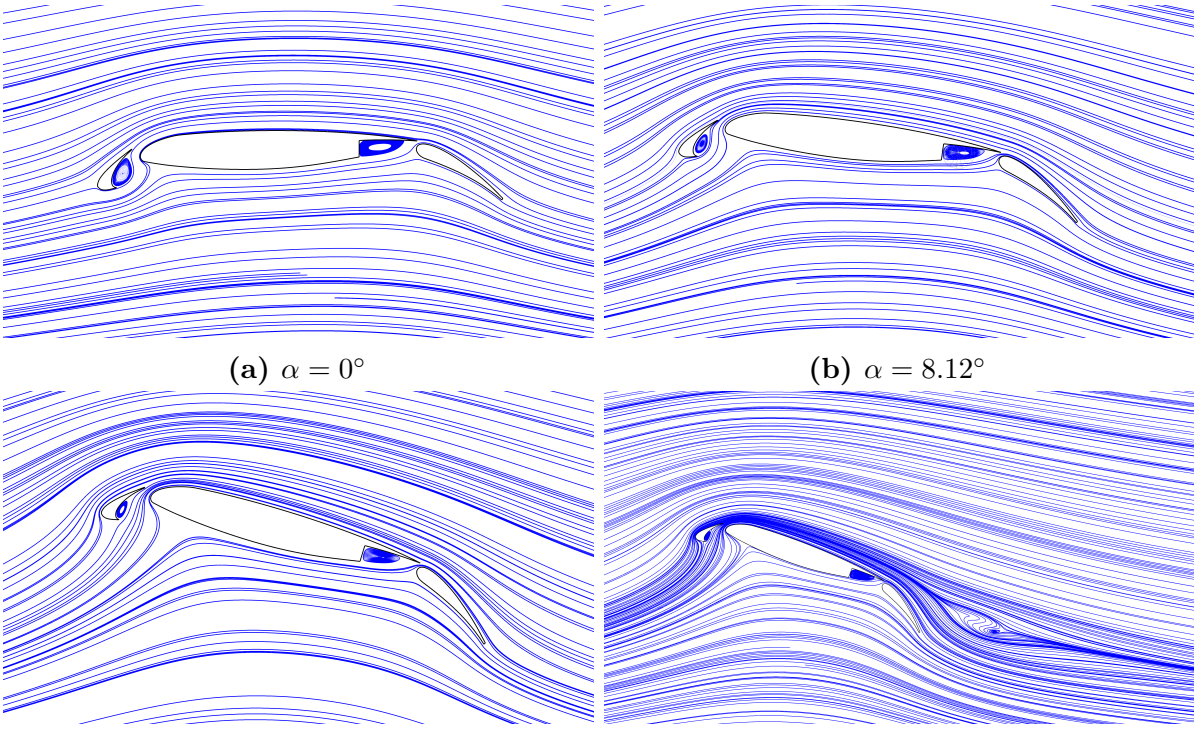


Figure 5-3: Pressure coefficient distribution at $\alpha = 8.12°$.

The streamlines at various angles of attack, Figure 5-4, illustrate how at even high angles of attack the flow from the gaps between the slat and main element, or main element and flap, can accelerate the velocity on the element surfaces reducing the possibility of flow reversal due to adverse pressure gradients. The downstream streamlines shown in Figure 5-4d shows what appears to be a recirculation pocket or vortices. This could be an exaggeration of the flow behaviour, since based on the lift curve in Figure 5-1a, the airfoil has already reached stall at approximately $\alpha = 18°$.

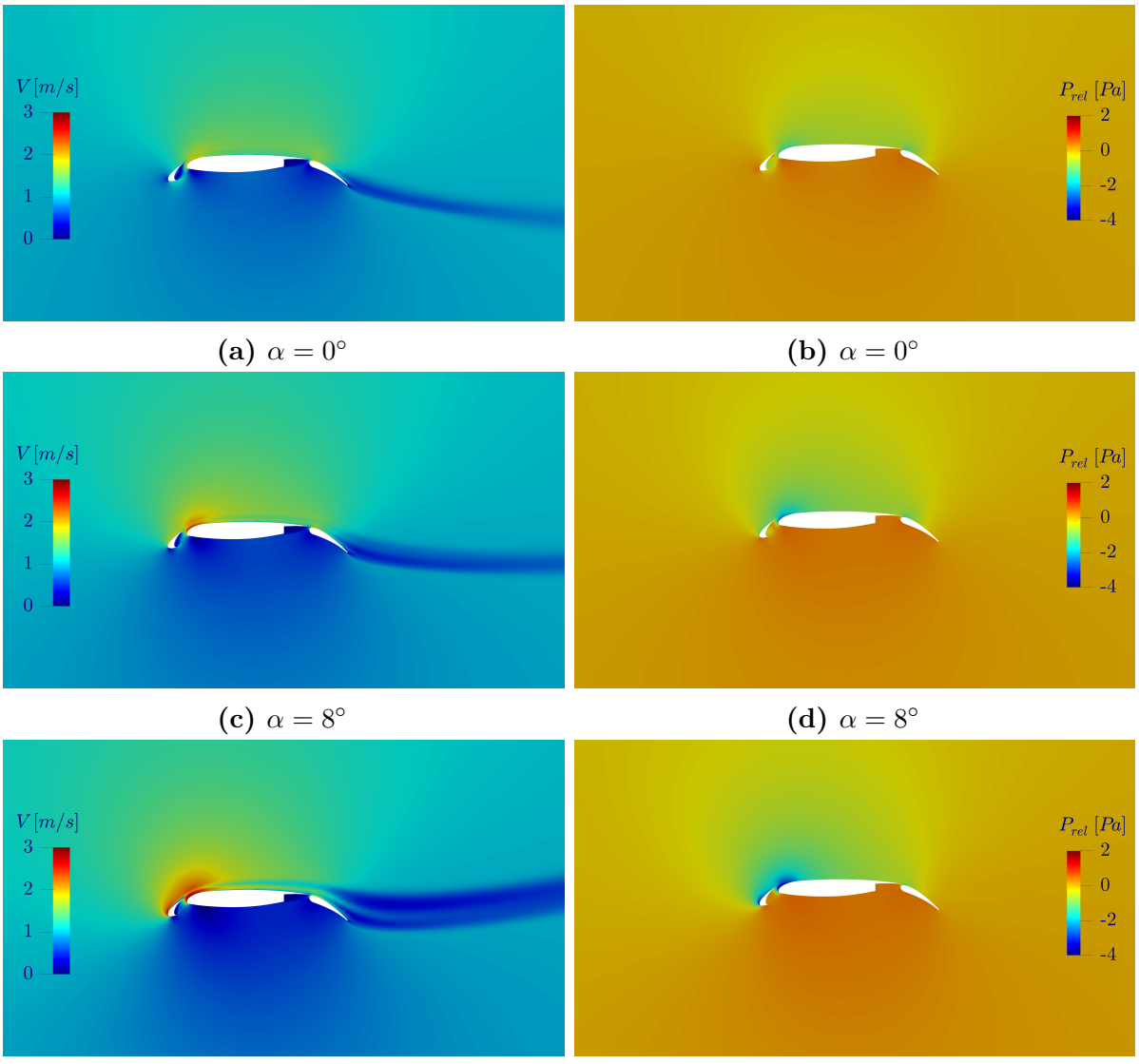


(a) $\alpha = 0°$ **(b)** $\alpha = 8.12°$

(c) $\alpha = 17°$ **(d)** $\alpha = 21.27°$

Figure 5-4: Streamlines based on the simulations at different angles of attack.

The velocity and relative pressure contours are shown in Figure 5-5. The velocity contours illustrate the following: the areas of accelerated flow (considerably higher than the freestream velocity of 1 m/s) especially at the LE of the main element which increases considerably at higher angles of attack leading to the high lift; the wake effects are shown to become more and more prevalent at higher angles of attack, for instance distinct wake effects from each element shown in Figure 5-5e, where the complex phenomena briefly discussed in Section 3.4 play increasingly critical roles to keep the flow attached; and it is shown that the upper surface sees highest flow variability in terms of velocity distribution while the majority of regions in the lower surface sees velocity magnitudes lower than freestream velocity. Note that the freestream velocity is chosen to be 1 m/s for convenience, and the other variables in the Reynolds number capture the Mach number used in the experimental data. The pressure contours obviously goes hand in hand with the velocity



(a) $\alpha = 0°$ **(b)** $\alpha = 0°$

(c) $\alpha = 8°$ **(d)** $\alpha = 8°$

(e) $\alpha = 18°$ **(f)** $\alpha = 18°$

Figure 5-5: Velocity (left column) and relative pressure (right column) contours from S-A model at various angles of attack.

contours. The suction side sees low static pressures, while interestingly, at high angles of attack the stagnation region on the LE of the slat and main element grows considerably.

5.2 Evaluating the discrete adjoint implementation

The adjoint implementation is verified in this section by comparing the adjoint derivatives with a brute-force finite difference approximation. The verification is done on the two-dimensional NACA 0012 aerofoil case from the DAFoam tutorials[1].

The finite-difference derivative approximation is given by the following formula,

$$\frac{\mathrm{d}\mathcal{J}}{\mathrm{d}\beta} \approx \frac{\beta(\boldsymbol{x}+\epsilon)-\beta(\boldsymbol{x})}{\epsilon}, \tag{5.1}$$

where the step size, ϵ, considered are from 5×10^{-3} to 5×10^{-9}. The brute-force computations are achieved by converging the two simulations, one with the original $\boldsymbol{\beta}$ field (unity at every cell), and the other perturbed by ϵ. The total number of flow iterations for both the brute-force and adjoint approach are kept the same.

For consistency the same step size is specified for the various partial derivative computations involved in the adjoint solutions. The relative residual tolerance for the adjoint linear equations is set to 10^{-6} following the cases in [61]. As pointed in that paper, it is not necessary to converge the solutions any further since the adjoint total derivatives will be accurate only up to four significant digits as a result of the errors in the finite-difference partial derivative approximations.

The regularisation constant λ in the objective function (Eqn. 3.6) is somewhat arbitrarily chosen to be 10^{-4} based on the separated aerofoil flows in [27]. As discussed in Chapter 3.2.2 an L-curve should be plotted by solving multiple inverse problems and experimenting with various values of λ. Since the project did not reach the optimisation stage, this is not possible, i.e. to solve the inverse problems the optimisation problems must be solved. However, this should not impact the verification process, which is the point of the present analysis.

The results for the adjoint solutions and the brute-force computations show reasonable agreement, as shown in Figure 5-6. As a sanity check, using Eqns. 3.6 and 5.1 it can be easily shown that the absolute value of the total derivative should converge to zero as the step sizes become smaller simply because the original $\boldsymbol{\beta}$ is unity everywhere, as explained in Chapter 3.2.2.

Figure 5-7 shows the error between the adjoint and brute-force solutions. The error is defined as $[(\delta_{\text{adjoint}} - \delta_{\text{FD}})/\delta_{\text{FD}}] \times 100$ where δ_{adjoint} and δ_{FD} are the total derivatives from the adjoint and brute-force solutions, respectively. It is shown that, on average, there is an error of approximately 2% between the two solutions. Although relatively small, it is still an order of magnitude higher than errors (around 0.1%) reported in the various simulation cases in [61]. It is not clear why this is the case, and time constraints have not allowed closer investigations.[2]

As an illustration of the computational time for the various partial derivatives involved in the adjoint solutions, a runtime breakdown is shown in Table 5-1. The times are based on running the simulation in parallel using four cores on a 2.7 GHz Intel "Core i5" MacBook Pro. As a reference the flow simulation runtime is 93 seconds, and the adjoint computation time is around 7 times the flow simulation runtime.

[1] Due to technical issues it was not possible to verify the adjoint implementation on the 30P-30N aerofoil, and although some time was spent investigating the reasons behind this, time constraint would not allow finding the causes for the issues. For verification purposes though, it is possible to use the existing cases in DAFoam, so the results shown in this section are based on the case described here: https://dafoam.readthedocs.io/en/latest/Tutorial_Aerodynamics_NACA0012_Incompressible.html

[2] Of course this will be investigated beyond the thesis submission deadline, in the near future.

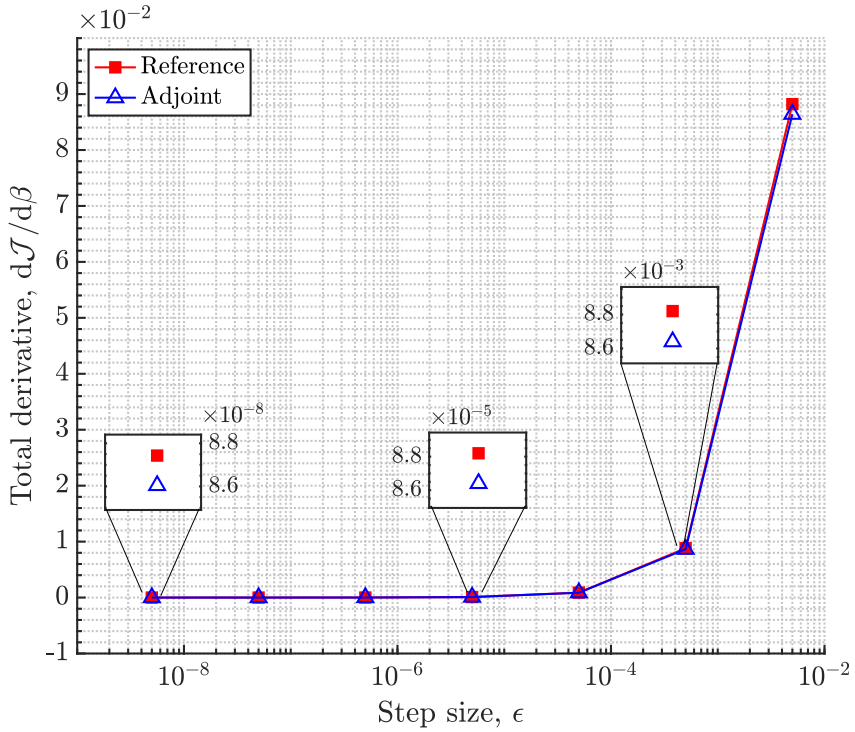


Figure 5-6: Total adjoint computation using a brute-force vs. the adjoint approach.

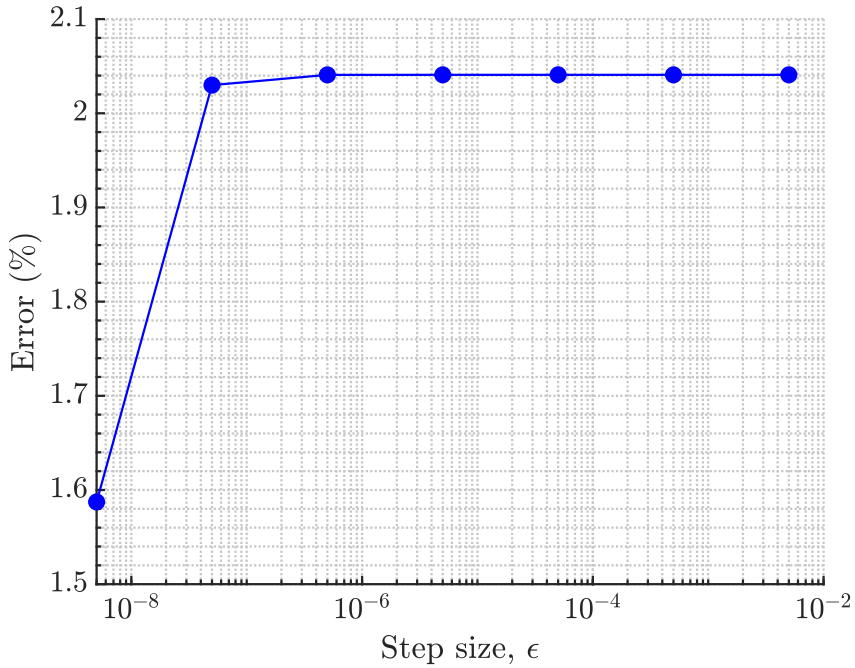


Figure 5-7: Error between reference derivatives and the adjoint implementation.

It is clear that the most expensive computations are for the partial derivatives $\partial \boldsymbol{\mathcal{R}}/\partial \boldsymbol{w}$ and its preconditioning matrix computations (PC) (refer to Chapter 4.2.1 for details), and $\partial \mathcal{J}/\partial \boldsymbol{\beta}$. In case of the former partial derivatives the runtimes are as expected since these are large matrices and runtime proportions are similar to those reported in [61].

However, the runtime involved in computing $\partial \mathcal{J}/\partial \boldsymbol{\beta}$ (partial derivative of the objective function with respect to the design variables) is far too high compared to less than 1% reported in [62, 61]. This is due to the fact that, in DAFoam this derivative is computed

using brute-force finite-difference method (as explained in Chapter 4.2.1). DAFoam is mainly written for shape optimisation where the design variables for the cases in [62, 61] are at most $\mathcal{O}(10^2)$, and thus evaluating $\partial\mathcal{J}/\partial\boldsymbol{\beta}$ is relatively cheap. However, for proposes of this work the design variables are equal to the number of mesh cells, which as explained earlier, are of $\mathcal{O}(10^5)$ and higher for the three-element aerofoils. Accelerating this computation using the Jacobian matrix colouring scheme will be investigated in the future.

Table 5-1: Runtime breakdown for the adjoint computations.

	Runtime (s)	Percent
$\partial\boldsymbol{\mathcal{R}}/\partial\boldsymbol{w}$	233	33.4
$\partial\boldsymbol{\mathcal{R}}/\partial\boldsymbol{w}$ (PC)	198	28.4
$\partial\mathcal{J}/\partial\boldsymbol{w}$	25	3.6
$\partial\mathcal{J}/\partial\boldsymbol{\beta}$	188	27.0
Adjoint equation	52	7.5

Finally, the scalability of adjoint implementation using higher number of CPU cores was not tested. This was due to technical issues in compiling DAFoam on the University high-performance computing cluster, most likely due to software incompatibilities between the various external modules employed in DAFoam. But it is expected that the adjoint computations should scale up to 10 million cells and 1536 CPU cores as reported in [61].

5.3 Summary of results

The main results are summarised as follows:

- At lower angles of attack, with reasonable meshes, both the S-A and $k-\omega$ SST models produce largely similar results for lift coefficients. However, there is a large discrepancy at high angles of attack, especially in the stall region where complex flow physics such as flow reversibility dominate.
- Both the turbulence models and all the meshes perform poorly in predicting drag coefficients irrespective of the angle of attack.
- Best agreement compared to experimental data is achieved by both turbulence models for the pressure coefficient distribution apart from some discrepancies in the slat cove region - which is region of largely recirculating flow phenomenon.
- The wake effects from the aerofoil elements, represented by the velocity profiles at three locations on the aerofoil, show variable results. Finer meshes tend to over-predict the wake effect in regions where the experimental data does not show such trend. The velocity profiles at the leading edge of the flap, where there is complex flow phenomenon in terms of confluent boundary layer flows and wake interactions of the slat and main element, show surprisingly good agreement to data. Towards the centre of the flap, while wake of the main element is reasonably captured (in terms of its location, though some variability exists in magnitude), there is a higher disagreement in terms of both magnitude and location for wake from the slat.

- The bespoke parallelisable adjoint implementation for the field inversion phase in DAFoam computes derivatives with an average error of around 2% compared to a brute-force finite difference method. This is an order of magnitude higher compared to average errors of around 0.1% for various flow cases used for testing DAFoam capabilities in [62, 61].
- The brute-force approach to computing the Jacobian matrix $\partial \mathcal{J}/\partial \boldsymbol{\beta}$, inherited from the original DAFoam approach, constitutes close to a third of the adjoint computation runtime compared to less than 1% reported in [62]. This is because for the shape optimisation applications in DAFoam cases the design variables are not greater than $\mathcal{O}(10^2)$ whereas for the current application the number of design variables are equal to the number of mesh cells.

"Research is what I'm doing when I don't know what I'm doing."
Wernher von Braun

6 Conclusions and future work

This dissertation takes the first steps towards statistical inference, termed field inversion in this context, for improving turbulence Reynolds-averaged Navier-Stokes closures for high-lift multi-element aerofoils. Starting with an overview of the turbulence simulations and modelling landscape, the recent rise in data-driven techniques for augmenting RANS closures has been outlined.

The field inversion is formulated as a deterministic optimisation problem. Due to the highly dimensional nature of this problem, a gradient-based optimisation approach is adopted. In this work an efficient, fully parallelisable, discrete adjoint gradient computation was implemented in the open-source CFD software OpenFOAM, using the DAFoam package. The results for the Spalart-Allmaras turbulence model illustrate that the adjoint computations are accurate to within 2%.

Baseline simulations using two popular RANS models, Spalart-Allmaras and $k-\omega$ shear stress transport with a range of meshes (average y^+ values of 2.5 at most) show that these models perform poorly at high angles of attack where complex flow phenomenon such as flow separation, wake interactions of the aerofoil elements, and confluent boundary layers dominate.

A summary of the adjoint evaluations and simulations are provided in Chapter 5.3, hence not included here for brevity's sake.

6.1 Short-term future plans

Having implemented the discrete adjoint method, the next step is to use optimisation algorithms to solve the inverse problems using the aerofoils and experimental data identified in Chapter 4.3. For this the `pyOptSparse` external module employed in DAFoam will be scoped. pyOpt is an object-oriented framework for solving nonlinear optimisation problems by allowing access to a range of optimisation algorithms through a common interface [72].

The next step will be the use of machine learning algorithms to use the outputs from the field inversion phase to construct a model of the discrepancy between RANS models and experimental data. As a first step neural networks will be used to train the models using the features identified in [27]. Finally, the discrepancy model will be injected in the baseline turbulence models for improved predictions.

6.2 Broader outlook

The FIML framework is a research topic in its infancy, as is the entire data-driven approach to turbulence modelling. Some key questions to be explored in the future and potential approaches are highlighted below:

Is it possible to systematically select features for machine learning model training and testing? Features are the inputs for the machine learning algorithms to train a model on. Key requirements are for the the features to be defined as a function of the mean flow properties of the RANS model, and should be Galilean invariant and independent of geometry in order to be useful for prediction in a generalised setting. For the limited cases the FIML framework has been applied, the feature selection has relied on intuition and understanding of turbulence modelling. But for the approach to be generalised, a systematic, and possibly automated approach, is essential.

One technique to address this question is the use of hill-climbing feature selection originally proposed in [73], and applied for two limited cases in the FIML context in [26]. In this method features are appended to a usable feature set until appending to it any further does not lead to improvements in the performance when fitting the output. To improve the machine learning capabilities embedding Galilean invariance into the framework will also be explored based on previous demonstration of its successful implementation outside the FIML framework [74].

What are the impacts of the various assumptions when formulating and optimising the objective function? The aim of the field inversion phase of the FIML framework is to find a corrective field which can be applied to a conventional RANS model such that the prediction output best approximates the high-fidelity data. This is ultimately posed as an optimisation problem with an objective function derived from arguments based on Bayesian statistics, and solved approximately by defining the posterior covariance matrix as the inverse of its Hessian [43]. The derivations involve massive assumptions about the Gaussian nature of the probability density functions (PDFs) for the prior probability, observational data, likelihood and conditional probability.

To address this question sampling techniques such as the Markov chain Monte Carlo sampling can be utilised to determine whether the posterior distribution from the sampling method agrees reasonably with the maximum a posteriori (MAP) estimations in the original field inversion phase, as demonstrated for a simplified case in [41].

Can more sophisticated RANS models achieve significant enough prediction improvements given the additional complexity involved? The FIML framework has so far been applied to RANS models with closures formulated based on the linear eddy-viscosity assumptions. As previously mentioned, for complex flows this assumption introduces significant errors. It is possible to apply the FIML framework to more sophisticated RANS closures (such as non-linear eddy viscosity and algebraic stress models), which can produce more accurate predictions, however, these models are more complex to implement and can suffer from convergence issues [75].

Addressing this question will depend on whether the non-linear eddy viscosity models or algebraic stress models are implemented in CFD solver platforms this project will choose to use, which will be scoped through an extensive literature search.

How does the formulation of the corrective term impact predictions? In the FIML framework, the corrective term is the spatial field that is injected in a conventional RANS model to improve predictions. In most of the cases applied so far, this corrective term has been in the form of a scalar field multiplied with the production term of a surrogate variable (e.g. the turbulent viscosity, $\tilde{\nu}_t$, in the Spalart-Allmaras model, or the specific rate of dissipation, ω, in the Wilcox $k-\omega$ model). [43] investigated defining two scalar corrective functions applied to the eigenvalues of the anisotropy tensor in the turbulence models, which resulted in prediction improvements.

To address this question inferring the full description of the Reynolds stress tensor

by inferring corrections to the eigenvectors along with the shape and magnitude will be explored through implementation based on the preliminary derivations in [43].

How does FIML perform across multiple classes of problems? For the turbulence models augmented using FIML previously, the flows cases have had similar properties to the one used for constructing the augmentation. In order to generalise the framework it is important to test the extent of its capabilities on multiple classes of flow cases.

Potential broadening of flow cases can include flows involving buoyancy effects, transient behaviour, mixing of multiple fluids, etc. However, this will be limited to the availability of high-fidelity data which will be explored through an extensive search of the available turbulence databases.

Is it possible to standardise the approach for reproduciblity and open science in the turbulence modelling community? One of the key achievements of the ML community has been the open sharing of data and software to encourage collaborative research. Although, similar frameworks exists in the turbulence modelling community for traditional approaches, i.e. a plethora of open-source platforms that implement the popular RANS models, such as OpenFOAM [59] and SU2 [76], implementation of these models in a data-driven context is nearly non-existent. Addition of a module to platforms such as OpenFOAM and SU2 for implementing the FIML framework will go a long way in accessibility within the wide research community, encouraging further investigations to explore the use of data-driven techniques in turbulence modelling.

Inspirations maybe taken from the SU2 implementation of using machine learning for uncertainty quantification in turbulence modelling [77]. This will also be scoped by direct contact with researchers originally proposing the FIML framework and their unsuccessful attempt at introducing the *Turbulence Modeling Gateway*[1] that aimed to address some aspects of this approach [5].

[1] http://turbgate.engin.umich.edu/

Bibliography

[1] Adel Abbas-Bayoumi and Klaus Becker. "An industrial view on numerical simulation for aircraft aerodynamic design". In: *Journal of Mathematics in Industry* 1.1 (2011), p. 10. DOI: 10.1186/2190-5983-1-10.

[2] P. R. Spalart and V. Venkatakrishnan. "On the role and challenges of CFD in the aerospace industry". In: *The Aeronautical Journal* 120.1223 (Jan. 2016), pp. 209–232. DOI: 10.1017/aer.2015.10.

[3] Xiangdong Li and Jiyuan Tu. "Evaluation of the eddy viscosity turbulence models for the simulation of convection–radiation coupled heat transfer in indoor environment". In: *Energy and Buildings* 184 (Feb. 2019), pp. 8–18. DOI: 10.1016/j.enbuild.2018.11.043.

[4] Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, Aug. 2000. DOI: 10.1017/cbo9780511840531.

[5] Anand Pratap Singh. "A framework to improve turbulence models using full-field inversion and machine learning". PhD thesis. 2018.

[6] PJ Mason. "Large-eddy simulation: A critical review of the technique". In: *Quarterly Journal of the Royal Meteorological Society* 120.515 (Jan. 1994), pp. 1–26. DOI: 10.1256/smsqj.51501.

[7] Philippe R. Spalart. "Detached-Eddy Simulation". In: *Annual Review of Fluid Mechanics* 41.1 (Jan. 2009), pp. 181–202. DOI: 10.1146/annurev.fluid.010908.165130.

[8] Giancarlo Alfonsi. "Reynolds-Averaged Navier–Stokes Equations for Turbulence Modeling". In: *Applied Mechanics Reviews* 62.4 (June 2009). DOI: 10.1115/1.3124648.

[9] Osborne Reynolds. "IV. On the dynamical theory of incompressible viscous fluids and the determination of the criterion". In: *Philosophical Transactions of the Royal Society of London. (A.)* 186 (Dec. 1895), pp. 123–164. DOI: 10.1098/rsta.1895.0004.

[10] Heng Xiao and Paola Cinnella. "Quantification of model uncertainty in RANS simulations: A review". In: *Progress in Aerospace Sciences* 108 (July 2019), pp. 1–31. DOI: 10.1016/j.paerosci.2018.10.001.

[11] David Wilcox. "A half century historical review of the k-omega model". In: *29th Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, Jan. 1991. DOI: 10.2514/6.1991-615.

[12] Bijan Mohammadi and Olivier Pironneau. "Analysis of the k-epsilon turbulence model". In: *Wiley–Masson Series Research in Applied Mathematics* (1993).

[13] P. Spalart and S. Allmaras. "A one-equation turbulence model for aerodynamic flows". In: *30th Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, Jan. 1992. DOI: 10.2514/6.1992-439.

[14] W.-D. Su et al. "A diagnosis of linear eddy-viscosity in turbulence modeling". In: *Physics of Fluids* 14.3 (Mar. 2002), pp. 1284–1287. DOI: 10.1063/1.1436495.

[15] Alan Celic and Ernst H. Hirschel. "Comparison of Eddy-Viscosity Turbulence Models in Flows with Adverse Pressure Gradient". In: *AIAA Journal* 44.10 (Oct. 2006), pp. 2156–2169. DOI: 10.2514/1.14902.

[16] Forrester T. Johnson, Edward N. Tinoco, and N. Jong Yu. "Thirty years of development and application of CFD at Boeing Commercial Airplanes, Seattle". In: *Computers & Fluids* 34.10 (Dec. 2005), pp. 1115–1151. DOI: 10.1016/j.compfluid.2004.06.005.

[17] T.B. Gatski and T. Jongen. "Nonlinear eddy viscosity and algebraic stress models for solving complex turbulent flows". In: *Progress in Aerospace Sciences* 36.8 (Nov. 2000), pp. 655–682. DOI: 10.1016/s0376-0421(00)00012-9.

[18] K. Hanjalić, M. Popovac, and M. Hadžiabdić. "A robust near-wall elliptic-relaxation eddy-viscosity turbulence model for CFD". In: *International Journal of Heat and Fluid Flow* 25.6 (Dec. 2004), pp. 1047–1051. DOI: 10.1016/j.ijheatfluidflow.2004.07.005.

[19] R. Nichols. "Applications of RANS/LES Turbulence Models". In: *41st Aerospace Sciences Meeting and Exhibit*. American Institute of Aeronautics and Astronautics, Jan. 2003. DOI: 10.2514/6.2003-83.

[20] Jeffrey Slotnick et al. "CFD vision 2030 study: a path to revolutionary computational aerosciences". In: *NASA Langley Research Centre* (2014). NASA/CR–2014-218178.

[21] Steven L. Brunton, Bernd R. Noack, and Petros Koumoutsakos. "Machine Learning for Fluid Mechanics". In: *Annual Review of Fluid Mechanics* 52.1 (Jan. 2020), pp. 477–508. DOI: 10.1146/annurev-fluid-010719-060214.

[22] Jerry Westerweel, Gerrit E. Elsinga, and Ronald J. Adrian. "Particle Image Velocimetry for Complex and Turbulent Flows". In: *Annual Review of Fluid Mechanics* 45.1 (Jan. 2013), pp. 409–436. DOI: 10.1146/annurev-fluid-120710-101204.

[23] Christopher L. Rumsey and Susan X. Ying. "Prediction of high lift: review of present CFD capability". In: *Progress in Aerospace Sciences* 38.2 (Feb. 2002), pp. 145–180. DOI: 10.1016/s0376-0421(02)00003-9.

[24] Steven M Klausmeyer and John C Lin. "Comparative results from a CFD challenge over a 2D three-element high-lift airfoil". In: (1997).

[25] Christopher L. Rumsey, Jeffrey P. Slotnick, and Anthony J. Sclafani. "Overview and Summary of the Third AIAA High Lift Prediction Workshop". In: *2018 AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, Jan. 2018. DOI: 10.2514/6.2018-1258.

[26] 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*. American Institute of Aeronautics and Astronautics, Jan. 2015. DOI: 10.2514/6.2015-1284.

[27] Anand Pratap Singh, Shivaji Medida, and Karthik Duraisamy. "Machine-Learning-Augmented Predictive Modeling of Turbulent Separated Flows over Airfoils". In: *AIAA Journal* 55.7 (July 2017), pp. 2215–2227. DOI: 10.2514/1.j055595.

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

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

[30] Andrew Pollard. "Whither Turbulence and Big Data for the Twenty-First Century". In: *Whither Turbulence and Big Data in the 21st Century?* Springer International Publishing, Aug. 2016, pp. 531–547. DOI: 10.1007/978-3-319-41217-7_30.

[31] 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*. IEEE, May 2012. DOI: 10.1109/itherm.2012.6231411.

[32] W.N. Edeling et al. "Bayesian estimates of parameter variability in the k–ϵ turbulence model". In: *Journal of Computational Physics* 258 (Feb. 2014), pp. 73–94. DOI: 10.1016/j.jcp.2013.10.027.

[33] Eric Dow and Qiqi Wang. "Uncertainty Quantification of Structural Uncertainties in RANS Simulations of Complex Flows". In: *20th AIAA Computational Fluid Dynamics Conference*. American Institute of Aeronautics and Astronautics, June 2011. DOI: 10.2514/6.2011-3865.

[34] Sai Hung Cheung et al. "Bayesian uncertainty analysis with applications to turbulence modeling". In: *Reliability Engineering & System Safety* 96.9 (Sept. 2011), pp. 1137–1149. DOI: 10.1016/j.ress.2010.09.013.

[35] J. Ling and J. Templeton. "Evaluation of machine learning algorithms for prediction of regions of high Reynolds averaged Navier Stokes uncertainty". In: *Physics of Fluids* 27.8 (Aug. 2015), p. 085103. DOI: 10.1063/1.4927765.

[36] J. Weatheritt and R.D. Sandberg. "The development of algebraic stress models using a novel evolutionary algorithm". In: *International Journal of Heat and Fluid Flow* 68 (Dec. 2017), pp. 298–318. DOI: 10.1016/j.ijheatfluidflow.2017.09.017.

[37] Jin-Long Wu, Heng Xiao, and Eric Paterson. "Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework". In: *Physical Review Fluids* 3.7 (July 2018). DOI: 10.1103/physrevfluids.3.074602.

[38] Jack Weatheritt. "The development of data driven approaches to further turbulence closures". PhD thesis. University of Southampton, 2015.

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

[40] Jian-Xun Wang, Jin-Long Wu, and Heng Xiao. "Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data". In: *Physical Review Fluids* 2.3 (Mar. 2017). DOI: 10.1103/physrevfluids.2.034603.

[41] 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 (Jan. 2016), pp. 758–774. DOI: 10.1016/j.jcp.2015.11.012.

[42] Anand Pratap Singh, Karthikeyan Duraisamy, and Ze Jia Zhang. "Augmentation of Turbulence Models Using Field Inversion and Machine Learning". In: *55th AIAA Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, Jan. 2017. DOI: 10.2514/6.2017-0993.

[43] Arent van Korlaar. "Field inversion and machine learning in turbulence modeling". In: *Delft University of Technology Thesis Repository* (2019). MSc Thesis.

[44] Jonathan R. Holland, James D. Baeder, and Karthikeyan Duraisamy. "Field Inversion and Machine Learning With Embedded Neural Networks: Physics-Consistent Neural Network Training". In: *AIAA Aviation 2019 Forum*. American Institute of Aeronautics and Astronautics, June 2019. DOI: 10.2514/6.2019-3200.

[45] Jonathan R. Holland, James D. Baeder, and Karthik Duraisamy. "Towards Integrated Field Inversion and Machine Learning With Embedded Neural Networks for RANS Modeling". In: *AIAA Scitech 2019 Forum*. American Institute of Aeronautics and Astronautics, Jan. 2019. DOI: 10.2514/6.2019-1884.

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

[47] M. P. Brenner, J. D. Eldredge, and J. B. Freund. "Perspective on machine learning for advancing fluid mechanics". In: *Physical Review Fluids* 4.10 (Oct. 2019). DOI: 10.1103/physrevfluids.4.100501.

[48] Georgi Kalitzin et al. "Near-wall behavior of RANS turbulence models and implications for wall functions". In: *Journal of Computational Physics* 204.1 (Mar. 2005), pp. 265–291. DOI: 10.1016/j.jcp.2004.10.018.

[49] Jari Kaipio and Erkki Somersalo. *Statistical and computational inverse problems*. Vol. 160. Springer Science & Business Media, 2006.

[50] Jeffrey S. Rosenthal. "Markov Chain Monte Carlo Algorithms: Theory and Practice". In: *Monte Carlo and Quasi-Monte Carlo Methods 2008*. Springer Berlin Heidelberg, 2009, pp. 157–169. DOI: 10.1007/978-3-642-04107-5_9.

[51] Mike Cullen et al., eds. *Large Scale Inverse Problems*. DE GRUYTER, Jan. 2013. DOI: 10.1515/9783110282269.

[52] Peter F Craigmile. "All of Statistics: A Concise Course in Statistical Inference". In: *The American Statistician* 59.2 (May 2005), pp. 203–204. DOI: 10.1198/tas.2005.s30.

[53] Richard C. Aster, Brian Borchers, and Clifford H. Thurber. *Parameter Estimation and Inverse Problems*. Elsevier, 2019. DOI: 10.1016/c2015-0-02458-3.

[54] Andrew R. Conn, Katya Scheinberg, and Luis Nunes Vicente. "Introduction to Derivative-Free Optimization". In: Society for Industrial and Applied Mathematics, Jan. 2009. DOI: 10.1137/1.9780898718768.ch1.

[55] Ashok D. Belegundu and Tirupathi R Chandrupatla. "Optimization Concepts and Applications in Engineering". In: Cambridge University Press. DOI: 10.1017/cbo9780511975905.002.

[56] Gaetan K.W. Kenway et al. "Effective adjoint approaches for computational fluid dynamics". In: *Progress in Aerospace Sciences* 110 (Oct. 2019), p. 100542. DOI: 10.1016/j.paerosci.2019.05.002.

[57] M.J. Tummers et al. "Computations of a turbulent wake in a strong adverse pressure gradient". In: *International Journal of Heat and Fluid Flow* 28.3 (June 2007), pp. 418–428. DOI: 10.1016/j.ijheatfluidflow.2006.07.002.

[58] A. Smith. "High-lift aerodynamics /37th Wright Brothers Lecture/". In: *6th Aircraft Design,Flight Test and Operations Meeting*. American Institute of Aeronautics and Astronautics, Aug. 1974. DOI: 10.2514/6.1974-939.

[59] Hrvoje Jasak, Aleksandar Jemcov, Zeljko Tukovic, et al. "OpenFOAM: A C++ library for complex physics simulations". In: *International workshop on coupled methods in numerical dynamics*. Vol. 1000. IUC Dubrovnik Croatia. 2007, pp. 1–20.

[60] S.V Patankar and D.B Spalding. "A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows". In: *International Journal of Heat and Mass Transfer* 15.10 (Oct. 1972), pp. 1787–1806. DOI: 10.1016/0017-9310(72)90054-3.

[61] Ping He et al. "DAFoam: An Open-Source Adjoint Framework for Multidisciplinary Design Optimization with OpenFOAM". In: *AIAA Journal* 58.3 (Mar. 2020), pp. 1304–1319. DOI: 10.2514/1.j058853.

[62] Ping He et al. "An aerodynamic design optimization framework using a discrete adjoint approach with OpenFOAM". In: *Computers & Fluids* 168 (May 2018), pp. 285–303. DOI: 10.1016/j.compfluid.2018.04.012.

[63] Andrew B. Lambe and Joaquim R. R. A. Martins. "Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes". In: *Structural and Multidisciplinary Optimization* 46.2 (Jan. 2012), pp. 273–284. DOI: 10.1007/s00158-012-0763-y.

[64] Assefaw Hadish Gebremedhin, Fredrik Manne, and Alex Pothen. "What Color Is Your Jacobian? Graph Coloring for Computing Derivatives". In: *SIAM Review* 47.4 (Jan. 2005), pp. 629–705. DOI: 10.1137/s0036144504444711.

[65] Nadezhda M. Afanasyeva and Maria V. Vasilieva. "9. PETSc for parallel solving of linear and nonlinear equations". In: *Computational Technologies*. DE GRUYTER. DOI: 10.1515/9783110359961.153.

[66] Youcef Saad and Martin H. Schultz. "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems". In: *SIAM Journal on Scientific and Statistical Computing* 7.3 (July 1986), pp. 856–869. DOI: 10.1137/0907058.

[67] Vincent Chin et al. "Flowfield measurements about a multi-element airfoil at high Reynolds numbers". In: *23rd Fluid Dynamics, Plasmadynamics, and Lasers Conference*. AIAA, July 1993. DOI: 10.2514/6.1993-3137.

[68] Kyle Pascioni, Louis N. Cattafesta, and Meelan M. Choudhari. "An Experimental Investigation of the 30P30N Multi-Element High-Lift Airfoil". In: *20th AIAA/CEAS Aeroacoustics Conference*. American Institute of Aeronautics and Astronautics, June 2014. DOI: 10.2514/6.2014-3062.

[69] G Redeker. "A selection of experimental test cases for the validation of CFD codes". In: *AGARD AR* 303 (1994).

[70] Christopher L. Rumsey et al. "Prediction of high-lift flows using turbulent closure models". In: *AIAA Journal* 36 (Jan. 1998), pp. 765–774. DOI: 10.2514/3.13890.

[71] J. Wild. "Mach and Reynolds Number Dependencies of the Stall Behavior of High-Lift Wing-Sections". In: *Journal of Aircraft* 50.4 (July 2013), pp. 1202–1216. DOI: 10.2514/1.c032138.

[72] Ruben E. Perez, Peter W. Jansen, and Joaquim R. R. A. Martins. "pyOpt: A Python-Based Object-Oriented Framework for Nonlinear Constrained Optimization". In: *Structural and Multidisciplinary Optimization* 45.1 (Jan. 2012), pp. 101–118. DOI: 10.1007/s00158-011-0666-3.

[73] Ron Kohavi and George H. John. "Wrappers for feature subset selection". In: *Artificial Intelligence* 97.1-2 (Dec. 1997), pp. 273–324. DOI: 10.1016/s0004-3702(97)00043-x.

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

[75] T. B. Gatski and C. G. Speziale. "On explicit algebraic stress models for complex turbulent flows". In: *Journal of Fluid Mechanics* 254 (Sept. 1993), pp. 59–78. DOI: 10.1017/s0022112093002034.

[76] Thomas D. Economon et al. "SU2: An Open-Source Suite for Multiphysics Simulation and Design". In: *AIAA Journal* 54.3 (Mar. 2016), pp. 828–846. DOI: 10.2514/1.j053813.

[77] Aashwin Ananda Mishra et al. "Uncertainty Estimation Module for Turbulence Model Predictions in SU2". In: *AIAA Journal* 57.3 (Mar. 2019), pp. 1066–1077. DOI: 10.2514/1.j057187.

A C++ codes

Although the actual codes written were a lot more, some key parts are presented in this chapter. A full documentation will be compiled in the near future.

Listing A.1: Main code to compute the objective function added to `AdjointObjectiveFunction.C` in the DAFoam source code.

```
void AdjointObjectiveFunction::calcBeta(scalar& objVal, scalarList& objDiscreteVal)
{
    // calculate the objective function in two stages, 1) c_l term, 2) beta term
    // calculating term 1
    wordList betaPatches;
    forAll(objFuncs_,idxI)
    {
        if(objFuncs_[idxI]=="beta") betaPatches=objFuncGeoInfo_[idxI];
    }

    const objectRegistry& db = mesh_.thisDb();
    const volScalarField& beta = db.lookupObject<volScalarField>("beta");
    const scalar& magURef = adjIO_.referenceValues["magURef"];
    const scalar& rhoRef = adjIO_.referenceValues["rhoRef"];
    const scalar& ARef = adjIO_.referenceValues["ARef"];
    const scalar& CLRef = adjIO_.referenceValues["CLRef"];
    const scalar& CLRef = adjIO_.referenceValues["lambda"];
    scalar pDyn = 0.5*rhoRef*magURef*magURef;


    vector fSum(this->calcForces());

    scalar CLsim = fSum & adjIO_.liftDir / (ARef*pDyn); // lift coefficient from the
        simulation


    // calculate beta term
    label cfsurfCounter = 0;
    forAll(betaPatches, cI)
    {
        // get the patch id label
        label patchI = mesh_.boundaryMesh().findPatchID(betaPatches[cI]);

        // create a shorter handle for the boundary patch
        const fvPatch& patch = mesh_.boundary()[patchI];

        forAll(patch, faceI)
        {
            const scalar& betaCell = beta.boundaryField()[patchI][faceI]; // extract
                beta value for the cell

```

```
40              objDiscreteVal[cfsurfCounter] = (betaCell - 1.0)*(betaCell - 1.0); // part
                    of term 2 of objective function
41
42              betaField_.boundaryFieldRef()[patchI][faceI] = betaCell;
43
44              cfsurfCounter += 1;
45
46          }
47      }
48
49      objVal = 0.0;
50      forAll(objDiscreteVal, idxI)
51      {
52          objVal += objDiscreteVal[idxI];
53      }
54
55      objVal = (CLsim - CLRef)*(CLsim - CLRef) + lambda*objVal;
56
57      forAll(surfForces_,idxI)
58      {
59          objDiscreteVal[idxI] += surfForces_[idxI] &adjIO_.liftDir;
60          objDiscreteVal[idxI] /= ARef*pDyn;
61      }
62
63      reduce(objVal, sumOp<scalar>() );
64
65      return;
66
67  }
```

Listing A.2: Main codes to compute the derivatives using the discrete adjoint method, added to `AdjointDerivative.C` in the DAFoam source code.

```
void AdjointDerivative::initializedRdBeta()
{
        label localSize = adjIdx_.nLocalAdjointStates;
    const label& nCells = mesh_.nCells();

        MatCreate(PETSC_COMM_WORLD,&dRdBeta_);
        MatSetSizes(dRdBeta_,localSize,PETSC_DECIDE,PETSC_DETERMINE,nCells);
        MatSetFromOptions(dRdBeta_);
        MatMPIAIJSetPreallocation(dRdBeta_,nCells,NULL,nCells,NULL);
        MatSetUp(dRdBeta_);
        Info<<"dRdBeta created!"<<endl;
        return;

}

void AdjointDerivative::initializedFdBeta()
{
        const label& nCells = mesh_.nCells();
        VecCreate(PETSC_COMM_WORLD,&dFdBeta_);
        VecSetSizes(dFdBeta_,PETSC_DETERMINE,nCells);
        VecSetFromOptions(dFdBeta_);
        Info<<"dFdBeta Created!"<<endl;

        return;

}

void AdjointDerivative::perturbBeta(scalar eps)
{
        volScalarField& beta
        (
                const_cast<volScalarField&>
                (
                        db_.lookupObject<volScalarField>("beta")
                )
        );

        forAll(beta.internalField(),cellI)
        {
                beta[cellI] += eps;

        }

        beta.correctBoundaryConditions();

}

void AdjointDerivative::calcdRdBeta()
{
        Info<<"Calculating dRdBeta"<<endl;
        const label& nCells = mesh_.nCells();

        MatZeroEntries(dRdBeta_);

        label isRef=1,isPC=0,transposed=0;
        this->copyStates("Ref2Var");
```

```
57          this->calcResiduals(isRef,isPC);
58          adjRAS_.calcTurbResiduals(isRef,isPC);
59
60          for (label i=0; i<nCells; i++)
61          {
62                  scalar eps=adjIO_.epsDerivBeta;
63
64                  this->perturbBeta(eps);
65
66                  this->calcResPartDeriv(eps,isPC);
67
68                  this->perturbBeta(-eps);
69
70                  this->setJacobianMat(dRdBeta_,i,transposed);
71          }
72
73          MatAssemblyBegin(dRdBeta_,MAT_FINAL_ASSEMBLY);
74          MatAssemblyEnd(dRdBeta_,MAT_FINAL_ASSEMBLY);
75
76          if(adjIO_.writeMatrices)
77          {
78                  adjIO_.writeMatrixBinary(dRdBeta_,"dRdBeta");
79          }
80
81          this->calcFlowResidualStatistics("verify");
82
83  }
84
85  void AdjointDerivative::calcdFdBeta(const word objFunc)
86  {
87          Info<<"Calculating dFdBeta for "<<endl;
88
89          const label& nCells = mesh_.nCells();
90
91          VecZeroEntries(dFdBeta_);
92
93          if(adjIO_.inletPatches.size()>1)
94                  FatalErrorIn("")<<"Inletpatch size >1"<<abort(FatalError);
95
96          label isRef=1;
97          this->copyStates("Ref2Var");
98          adjObj_.calcObjFuncs(objFunc,isRef);
99
100         for (PetscInt i=0; i<nCells; i++)
101         {
102                 scalar eps=adjIO_.epsDerivBeta;
103
104                 this->perturbBeta(eps);
105
106                 adjObj_.calcObjFuncPartDerivs(eps,objFunc);
107
108                 // reset perturbation
109                 this->perturbBeta(-eps);
110
111                 PetscScalar val;
112
113                 val = adjObj_.getObjFuncPartDeriv(objFunc);
114
```

```
115                 if (fabs(val) > adjIO_.minTolJac && fabs(val) < adjIO_.maxTolJac)
116                 {
117                         PetscInt rowI = i;
118                         VecSetValue(dFdBeta_,rowI,val,INSERT_VALUES);
119                 }
120         }
121
122         this->calcFlowResidualStatistics("verify");
123
124         VecAssemblyBegin(dFdBeta_);
125         VecAssemblyEnd(dFdBeta_);
126
127         if(adjIO_.writeMatrices)
128         {
129                 adjIO_.writeVectorASCII(dFdBeta_,"dFdBeta_"+objFunc);
130         }
131
132         return;
133
134 }
135
136 void AdjointDerivative::calcTotalDerivBeta()
137 {
138         Info<<"Calculating Beta Total Derivatives. "<<this->getRunTime()<<" s"<<endl;
139
140         this->initializedRdBeta();
141         this->calcdRdBeta();
142
143         this->initializedFdBeta();
144
145         const label& nCells = mesh_.nCells();
146
147         VecCreate(PETSC_COMM_WORLD,&dFdBetaTotal_);
148         VecSetSizes(dFdBetaTotal_,PETSC_DETERMINE,nCells);
149         VecSetFromOptions(dFdBetaTotal_);
150
151         Vec psi;
152         VecDuplicate(psi_,&psi);
153
154         forAll(adjIO_.objFuncs,idxI)
155         {
156                 word objFunc = adjIO_.objFuncs[idxI];
157                 word psiName = "psi_"+objFunc;
158                 word sensName = "objFuncsSens_d"+objFunc+"dBeta.bin";
159                 word sensNameASCII = "objFuncsSens_d"+objFunc+"dBeta.dat";
160
161                 this->calcdFdBeta(objFunc);
162                 VecZeroEntries(psi);
163                 adjIO_.readVectorBinary(psi,psiName.c_str());
164
165                 VecZeroEntries(dFdBetaTotal_);
166                 MatMultTranspose(dRdBeta_,psi,dFdBetaTotal_);
167                 VecAXPY(dFdBetaTotal_,-1.0,dFdBeta_);
168                 VecScale(dFdBetaTotal_,-1.0);
169
170                 PetscViewer viewer;
171                 PetscViewerBinaryOpen(PETSC_COMM_WORLD,sensName.c_str(),
172                         FILE_MODE_WRITE,&viewer);
```

```
173                 VecView(dFdBetaTotal_,viewer);
174                 PetscViewerDestroy(&viewer);
175 
176                 PetscViewer viewer1;
177                 PetscViewerASCIIOpen(PETSC_COMM_WORLD,sensNameASCII.c_str(),&viewer1);
178                 PetscViewerPushFormat(viewer1,PETSC_VIEWER_ASCII_MATLAB);
179                 VecView(dFdBetaTotal_,viewer1);
180                 PetscViewerDestroy(&viewer1);
181         }
182 
183         Info<<"Calculating Beta Total Derivatives. Completed. "<<this->getRunTime()<<"
                s"<<endl;
184 
185         return;
186 
187 }
```

B OpenFOAM boundary conditions

Table B-1: Boundary conditions for key flow quantities. Descriptions of the various boundary conditions in OpenFOAM: https://www.openfoam.com/documentation/user-guide/standard-boundaryconditions.php

Boundary	U	P	ν_t
`farfield`	`freestream`	`freestreamPressure`	`freestream`
`airfoil`	`fixedValue`	`zeroGradient`	`nutUSpaldingWallFunction`
`internalField`	`uniform`	`uniform`	`uniform`

Table B-2: SA and $k-\omega$ SST specific boundary conditions.

Boundary	k	ω	$\tilde{\nu}$
`farfield`	`freestream`	`inletOutlet`	`freestream`
`airfoil`	`kqRWallFunction`	`omegaWallFunction`	`fixedValue`
`internalField`	`uniform`	`uniform`	`uniform`