APPLICATION AND EXTENSION OF A DATA-DRIVEN TURBULENCE MODELING METHOD USING MACHINE LEARNING

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of

Master of Science in Aerospace Engineering

by

Parv Khurana

November 2021

Student number: 4808827
Project Duration: February 2021- November 2021
Thesis Committee: Dr. Richard P. Dwight, TU Delft, Chair
Dr. Steven J. Hulshoff, TU Delft, Examiner
Dr E. van Kampen, TU Delft, Examiner
Florian Jäckel, DLR, Thesis supervisor

TU Delft

The work in this thesis was made in the:

Turbulence and Transition Modeling group
C²A²S²E - Center for Computer Applications in AeroSpace Science and Engineering
Institute for Aerodynamics and Flow Technology
German Aerospace Center (DLR)

Contract Duration:	February 2021- October 2021
Supervisors:	Dr. Cornelia Grabe, DLR
	Dr. Richard P. Dwight, TU Delft
	Dr. Andreas Krumbein, DLR
	Florian Jäckel, DLR

ABSTRACT

Numerical efforts to estimate turbulence in fluid flows are focused on developing turbulence models, with Reynolds Averaged Navier–Stokes (RANS) models being the most popular. RANS methods are practical to apply on complex geometries, and high Reynolds number flows, albeit at a loss of accuracy in difficult flow situations like separation and transition. In recent years, many data-driven approaches which leverage high-fidelity data have been developed to augment the performance of RANS models. The goal of this M.Sc. thesis is to apply and extend one such data-driven approach "*Field Inversion and Machine Learning (FIML)*" [Parish and Duraisamy, 2016] to improve the negative Spalart-Allmaras (SA-neg) turbulence model, with specific application to the shock-induced separation on a 2D airfoil profile. Inversion techniques involve formulating an optimisation problem aiming to provide an improved closure for the turbulence model at the point of inversion by minimising a measure of discrepancy between the baseline model and the high-fidelity data. This results in a corrective, spatially distributed discrepancy field and is referred to as β in this work. To incorporate a general β field for improved predictions in a RANS solver, machine learning algorithms (neural networks in this case) will be used to find a functional approximation. Machine learning (ML) algorithms will identify patterns in the training data, which is an appropriately chosen set of flow features (η_i) from the solutions of the inverse problem for multiple flow cases for the 2D airfoil over a range of Mach numbers (M), Reynolds Number (Re), and angle of attacks (AoA). This work's primary objectives are to identify flow features (η_i) relevant to shock-induced flow separation. The improved RANS model will be tested on unseen flow conditions to evaluate the generalisation capability of the machine learning augmentation.

ACKNOWLEDGEMENTS

Firstly, I would like to express my humble gratitude to Dr Cornelia Grabe and her team at the German Aerospace Center (DLR), Göttingen, for providing me with the incredible opportunity to work on this exciting project and making the whole experience in Germany a memorable one despite the challenging circumstances forced by the pandemic. Thank you, Frau Christiane Lenz, for the smooth recruitment process, facilitating my move to Germany and helping me settle in well in the city and DLR office. I would also like to thank Dr Andreas Krumbein and Dr Philipp Bekemeyer with their respective teams for being great people to work with. From their technical knowledge and willingness to help with project-related problems, I could not have been better placed to conduct my master thesis. I would also like to thank my supervisor Florian Jäckel for being a great mentor and daily supervisor. Right from being extremely patient with me while I was struggling to grasp the new tools at DLR, to providing great advice for the project; it never felt like I was your first student. Also, thank you being the ever ready döner buddy for lunch. Finally, I would like to thank Susan, Gideon and Lucas, my flatmates turned family in Göttingen, and Irene, my TU Delft buddy at DLR, for all the great experiences and fun times during the past 9 months of my thesis.

Secondly, I would like to thank all the teachers who taught me at TU Delft for making me fall in love with aerospace engineering and aerodynamics. My sincere gratitude goes to my thesis supervisor at TU Delft, Dr Richard Dwight, for accepting me as his thesis student. Your enthusiasm to support me in Germany, holding the thesis meetings and regularly interacting with me and the group at DLR was a massive factor in completing my thesis. I always looked forward to the thesis meetings for our mentally stimulating discussions and excellent technical advice. I always left with a bag full of thoughts and ideas after the meetings. Finally, I would like to thank my support system at TU Delft. My friends from aerodynamics, Uttam, Abhyuday and Muaaz, for motivating me to perform to my limits and being great people to hang out with at the same time. Rahul and Krishna, the first friends I made at Delft, for your constant support and tolerating all my antics.

This thesis project brings an end to one of the most exciting yet challenging periods of my life, staying away in a foreign land and being locked up for more than a year due to the pandemic. I would like to shoutout to the friends back home for always being a text away, keeping my morale high through a difficult personal time and supplying me the best memes from halfway around the globe; you know who you are. Last but not least, I would like to thank my big fat family of nine, each one with their own contribution to keep me sane and happy. Without their endless and unconditional love and support, I could not have made it through this course. All my success and happiness is best shared with you.

Parv Khurana
Göttingen, November 2021

CONTENTS

1 INTRODUCTION

1.1 BACKGROUND

Turbulence exists in all kinds of relevant fluid flows. Characterization of turbulent flow behaviour is a requirement for most practical fluid dynamic problems. The most straightforward option, in theory, is to directly solve the governing flow equations using direct numerical simulation (DNS); however, the unsteady behaviour of these flows and extensive range of spatial scales makes it extremely difficult to afford computationally. DNS has been used to solve canonical cases in industry and academia; however, this method cannot be implemented in practical problems like estimating the flow field around an aeroplane wing or a turbomachinery component for decades to come. Therefore, there always has been a need to develop reduced-fidelity methods to model turbulence in the governing equations addressed by Reynolds-Averaged Navier–Stokes (RANS) and Large Eddy Simulation (LES) turbulence models in the previous years. Both methods involve decomposing the flow variables ($\mathbf{U}$) into resolved ($\widetilde{\mathbf{U}}$) and unresolved ($\mathbf{U}'$), represented as $\mathbf{U} = \mathbf{U}' + \widetilde{\mathbf{U}}$. In LES, this decomposition is done using low-pass filtering of the physical scales of the flow variables to be resolved, and in RANS, it is done using ensemble averaging of the governing equations into mean and fluctuating parts of the flow.

LES methods provide higher fidelity results than RANS methods but are still computationally expensive for large-scale practical applications. RANS models are the most popular methods due to their capabilities to be applied to complex geometries, and high Reynolds number flows. The reason for this is the low computational cost, which arrives from the fact that the turbulent behaviour in RANS turbulent models depends entirely on the (empirical, mathematical, intuition-based, practical) modelling assumptions used to close the model. However, these models are still inadequate in terms of accuracy as information is always lost during the averaging process regardless of the turbulence modelling choice, and the modelling assumptions for the closure terms are often based on canonical flow cases like the flat plate flow, which makes it difficult to predict situations like separation and transition.

Turbulence modellers have been using experiments and DNS datasets for the calibration of these RANS models. However, due to the recent abundance of data and the development of sophisticated computational hardware and algorithms, the use of data-driven turbulence modelling has gained importance [Duraisamy et al., 2019]. Furthermore, these datasets can be used in today's accessible and scalable machine learning algorithms to provide even more improved turbulence models [Brunton et al., 2019]. In literature, there are various methods to model turbulence that are purely *data-driven*, i.e. are only dependent on data, but the focus of this work will be on methods that are *data-enabled* [Duraisamy, 2020]. Data-enabled means existing turbulence models are employed, and data is used to improve their performance or *augment* them. One such data-enabled approach is the "*Field Inversion and Machine Learning (FIML)*", formalized by Parish and Duraisamy [2016].

One of the more challenging problems to solve for CFD solvers is transonic flows with shock-wave boundary-layer interactions (SBLIs). For a strong enough shock, these interactions cause separation of the boundary layer from the airfoil at the

shock location leading to a "shock stall", characterized by a sudden loss of lift and increase in drag. The thesis applies the FIML approach to the test case of shock-induced separation on 2D airfoils operating in a transonic regime. The airfoil in consideration for this project is the RAE 2822 airfoil, which is a rear-loaded, subcritical, roof-top type pressure distribution at design conditions. The RAE 2822 airfoil has been used in literature to validate CFD codes and turbulence models as various flow characteristics like the shock location, the extent of separation, the pressure downstream of the shock need to be correctly estimated. Incorrect estimation of these phenomena can wrongly predict the lift, pitching, and hinge moments during win design.

This MSc thesis project aims to apply the FIML approach to improve the negative Spalart-Allmaras (SA-neg) turbulence model [Spalart and Johnson, 2012] with specific application to the shock-induced separation on the 2D profile of the RAE2822 airfoil. The student undertook this project at the DLR Institute for Aerodynamics and Flow Technology, Göttingen, Germany. The high-fidelity data is supplied to the student by DLR and was generated by Airbus. The Airbus RWC.01 database gathers experimental aerodynamic data acquired in 2016 using the pilot facility of the European Transonic Wind Tunnel (pETW) for a series of 2D airfoil sections.

1.2 RESEARCH OBJECTIVE & RESEARCH QUESTIONS

The main research objective for this M.Sc. thesis work is:

> *"To apply and extend the data-driven Field Inversion and Machine Learning (FIML) approach to augment the negative Spalart-Allmaras (SA-neg) turbulence model, with specific application to the shock-induced separation on a 2D transonic airfoil profile."*

The achievement of the above objective would signify that the FIML approach can be extended to the shock-induced separation case on a 2D transonic airfoil profile, which is a flow case that has not been tested extensively yet in literature. This study is the first one in the author's knowledge that uses the extensive Airbus RWC.01 database for a data-driven turbulence modelling activity. The aim of the study. The focus of this work was in particular on the applicability of the FIML approach to shock-induced flow separation and the feature selection and engineering process for the Machine Learning step. The generation of a general augmented RANS turbulence model is not addressed here.

From the understanding of the problem at the start of the project, the following research question that drives the project is identified:

> *"How can the Field Inversion and Machine Learning (FIML) approach be applied to augment the negative Spalart-Allmaras (SA-neg) turbulence model for shock-induced separation on a 2D transonic airfoil?"*

The following sub-questions have been identified:

- Is the experimental aerodynamic database of high-fidelity wind-tunnel measurements dataset relevant and representative of the flow cases on which the augmented turbulence model will be applied?
- Does a unique discrepancy field exists which can be inferred using a deterministic inversion approach? Does this unique field correspond to the global minimum of the discrepancy between the baseline model and high-fidelity data? Is it possible to reach this solution with the current tools and computational resources at hand?

- Will machine learning algorithms be able to find the patterns in the discrepancies obtained from inversion solutions? Are neural networks the best algorithm for the learning process in this case? What is the best architecture/ set of hyper-parameters for the neural networks to be used for learning?
- What is the appropriate set of features required for the learning process? How many features are necessary? Should the features specific to the test case (shock-induced separation) be introduced in the learning process?
- Are the results of this learning process interpretable? Is there a causal relationship between the data, the discrepancy, and the corresponding prediction?
- Does this ML-augmentation provide better results than the baseline model? Can it predict test cases with design points unrelated to those in the training database?

1.3 THESIS OUTLINE

Chapter 2 gives an overview of the theoretical background required for the topics discussed in this thesis work. This chapter includes topics like turbulence modelling focusing on RANS, transonic flows for a 2D airfoil, optimization techniques, machine learning and feature engineering. Chapter 3 gives a review of data-driven turbulence modelling activities with a focus on the usage of machine learning and then sets up the paradigm for the approach used in this thesis - the Field Inversion and Machine Learning (FIML) approach. A summary of the current FIML implementation is also provided in Chapter 3. Chapter 4 explores the scope of the Airbus RWC.01 experimental aerodynamic database provided to the author and provides a method of selecting the data points for the FIML activity considering the shock-induced separation case in mind. Chapter 5 provides a detailed description of how the Field Inversion part of the FIML approach was implemented for the selected data points, followed by the results and recommendations to improve the procedure further. Chapter 6 deals with the feature engineering pipeline used to process the input features for machine learning, providing explanations for the choices made considering the physical problem, data and the machine learning algorithm of neural networks. Chapter 7 focuses on the Machine Learning part of the FIML approach, motivating the design choices made behind the training procedure followed by results on various testing cases. Finally, Chapter 8 concludes the work by answering the research question stated in this chapter, commenting on the generality of this machine learning augmentation activity and providing recommendations for future work.

2 THEORETICAL BACKGROUND

This chapter gives a broad overview of all the pre-requisite theory to delve into this project. Section 2.1 is about turbulence modeling with special focus on the Spalart-Allmaras turbulence model, the model of choice for this study. Section 2.2 is about transonic flows around a 2D airfoil, which discusses the shock wave boundary layer interactions (SBLI) involved in this case. Section 2.3 focuses on theory optimization techniques which will be useful for solving the upcoming Field Inversion problem. Finally, Section 2.4 gives the theoretical framework for machine learning and feature engineering which is helpful for the ML part of the FIML procedure.

2.1 TURBULENCE MODELING

Instabilities, unsteadiness and randomness characterize turbulent behaviour in fluid flows. If a laminar flow is disturbed at low velocities, where viscous forces dominate, it tends to recover from the instabilities. The recovery does not take place at high velocities, and flow is said to become turbulent. The critical flow condition can be defined using a non-dimensional quantity called the Reynolds number Re, which describes the ability of the flow to *transition* from laminar to turbulent:

$$Re = \frac{\text{Inertia forces or Momentum of the flow}}{\text{Viscous forces of the flow}} = \frac{UL}{\nu}, \tag{2.1}$$

where U and L are characteristic velocity and length scales and ν is the kinematic viscosity of the fluid flow. A high Reynolds number implies domination of inertial forces of the flow, making it more susceptible to instabilities.

Turbulence is generated in the largest flow scales (integral scales - big vortices), and the turbulent energy is progressively transferred to medium-scale vortices (inertial sub-range) and dissipated in the small scale vortices. The energy spectrum of turbulence energy as a function of wavelength (or wavenumber) is governed by kinematic viscosity $\nu[m^2/s]$ and energy dissipation rate $\epsilon[m^2/s^3]$. Thus using this ansatz and dimensional arguments, the smallest associated characteristic scales of turbulent flows can be defined as:

$$\eta_K \equiv \left(\frac{\nu^3}{\epsilon}\right)^{1/4}, u_K \equiv (\epsilon\nu)^{1/4}, \tau_K \equiv \left(\frac{\nu}{\epsilon}\right)^{1/2}, \tag{2.2}$$

where, η_K, u_K and τ_K are the length, velocity and time scale, respectively. These are also known as Kolmogorov microscales as they come with the underlying assumption given by the Kolmogorov hypothesis, which states that at sufficiently high Reynolds numbers, small-scale turbulent motions are statistically isotropic.

To sufficiently resolve all the scales of the turbulent flows numerically, one needs to solve the governing equation for fluid flows (Navier-Stokes equations) on a computational mesh fine enough to resolve all turbulent scales without any assumptions. Simulating the flow in this manner is known as Direct Numerical Simulation (DNS) and can also be called the "exact" solution for a given flow case. Navier-Stokes equa-

tions for unsteady incompressible flow are presented below (in Einstein notation):

$$\begin{aligned} \frac{\partial u_i}{\partial x_i} &= 0 \\ \frac{\partial u_i}{\partial t} + \frac{\partial u_i u_j}{\partial x_j} &= -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \nu \frac{\partial^2 u_i}{\partial x_j \partial x_j} \quad i = 1,2,3 \end{aligned} \tag{2.3}$$

where u is the velocity vector, p and ρ are the pressure, and density, respectively. Navier-Stokes equations are non-linear partial differential equations that need to be solved computationally. Assuming that a mesh is used for discretization, the mesh should be fine enough to capture η_K, u_K and τ_K of the flow. Therefore, the number of grid points to capture an integral scale of length L with a mesh of smallest element η_K is:

$$N_L = \frac{L}{\eta_K} \sim Re^{3/4} \tag{2.4}$$

In a three dimensional problem, the number of grid points required is N_L^3. Therefore, the number of grid points will scale with the Reynolds number with $\left(Re^{3/4}\right)^3 \equiv Re^{9/4}$. Assuming the number of time steps required follows from the number of grids points, i.e. $N_T \sim N_L$, which is a stable time stepping condition; the estimated cost of DNS procedure is $N_T \times N_L^3 \equiv Re^3$. Therefore, the cost of DNS for aerospace applications with typically high-Reynolds number flows is almost always too high to be used for practical applications.

2.1.1 Need for modeling: RANS and LES

To reduce the computational cost of solving the Navier-Stokes equation numerically, there is a need for some form of modelling which captures only the most essential flow properties. This need is addressed by Reynolds-Averaged Navier–Stokes (RANS) and Large Eddy Simulation (LES) formulations. LES methods are generally more accurate and expensive than RANS methods. The focus of this project is RANS methods, which are the most popular method for aerospace applications because they provide an acceptable quality of solution with a quick turnaround time.

These methods involve decomposing the flow variables ($\mathbf{U}$) into resolved ($\widetilde{\mathbf{U}}$) and unresolved ($\mathbf{U}'$). Both these methods lead to an unclosed system of equations or a closure problem. Consider the following set of equations:

$$\begin{aligned} \frac{\partial \tilde{u}_i}{\partial x_i} &= 0 \\ \frac{\partial \tilde{u}_i}{\partial t} + \widetilde{u_j}\frac{\partial \tilde{u}_i}{\partial x_j} &= -\frac{1}{\rho}\frac{\partial \tilde{p}}{\partial x_i} + \nu \frac{\partial^2 \tilde{u}_i}{\partial x_j \partial x_j} - \frac{\partial \tau_{ij}}{\partial x_j} \end{aligned} \tag{2.5}$$

For LES, the decomposition is done in flow length scales, or $\widetilde{\mathbf{U}}$ represent the resolved scales of the flow and $\mathbf{U}'$ correspond to the scales filtered out (typically by mesh resolution). The momentum equation contains the term $\tau_{ij} = \widetilde{u_i u_j} - \tilde{u}_i \tilde{u}_j$ which is unclosed because it is a function of the unresolved scales. This term is referred to as subgrid-scale stress (SGS) tensor. In the case of LES Equation 2.5 is achieved by applying the filter operation on the Navier-Stokes equations.

For RANS, the decomposition is achieved by using ensemble averaging of the governing equations into a mean ($\widetilde{\mathbf{U}}$) and fluctuating ($\mathbf{U}'$) parts of the flow. The decomposition has the property that average of all the fluctuations is zero or $\widetilde{\mathbf{U}'} = 0$. The mean flow is resolved in space and time, and the effects of the fluctuations are modelled. The averaging operation, in this case, leads to the unclosed term $\tau_{ij} = \widetilde{u_i' u_j'}$

and is referred to as the Reynolds stress tensor. The Reynolds stress tensor can be further split into isotropic and deviatoric parts:

$$\widetilde{u_i'u_j'} = \frac{2}{3}\delta_{ij}k + a_{ij}, \tag{2.6}$$

where a_{ij} is the anisotropic part of the Reynolds stress tensor, δ_{ij} is the Kronecker delta fucntion and $k = \frac{1}{2}\widetilde{u_i'u_i'}$ is the turbulent kinetic energy. A non-dimensionless version of the anisotropic part, often referred as the anisotropy tensor especially in FIML literature, can be written as:

$$b_{ij} = \frac{a_{ij}}{2k}. \tag{2.7}$$

The addition of Reynolds stress tensor to the equations leads to an increase in diffusivity in the flow. Properties of the Reynolds stress tensor are similar to that of viscous stress. It is possible to formulate a transport equation for the Reynolds stresses using the Navier-Stokes equations, but the order of the unclosed terms keeps increasing, and they cannot be solved. Therefore the Reynolds stress needs to be modelled using empirical approximations. Considering the scope of this thesis, only one class of these models, namely Eddy Viscosity Models (EVM), is discussed in this chapter.

Eddy viscosity models

This class of turbulence models are the simplest form of turbulence models. These models have been based on the observation that turbulence leads to momentum exchange between fluid elements. The deviatoric part of the Reynolds stress tensor is proportional to the mean shear rate, and the proportionality factor is the eddy viscosity or turbulent viscosity. The deviatoric part of the Reynolds stress tensor is then defined according to the ansatz:

$$a_{ij} \approx -2\nu_t S_{ij} \tag{2.8}$$

where S_{ij} is the mean strain rate tensor and ν_t is the eddy-viscosity. This ansatz is also known as the Boussinesq hypothesis. Substituting the ansatz in the RANS equation transforms the problem only to estimate the scalara eddy-viscosity variable instead of six Reynolds stress components. The assumption is a strong one; however, it is confirmed by DNS to be valid for flows with a distinct mean flow direction. Eddy viscosity models have provided accuracy of acceptable levels for attached flows. However, the eddy viscosity models do not work well for situations like wall-bounded flows (the wall blocks one flow velocity component). Regardless, their ease of implementation and less complexity makes them the most important tool in industrial applications.

Eddy-viscosity models can be further sub-classified based on how the eddy viscosity is expressed and the number of additional transport equations needed to solve for their use:

1. Algebraic models (zero-equation): This class of models do not use any additional transport equations and are calculated using a *mixing length* assumption. The mixing length is analogous to the mean free path, which is approximately the distance from the wall (for wall bounded flows). They are simple to implement but only provide accurate results for simple flows the type of results they have been calibrated on. Examples are Prandtl's mixing length model, Baldwin-Lomax model [Baldwin and Lomax, 1978].

2. One-equation models: This class of models has one additional transport equation to solve from the eddy viscosity formulation. Examples include the

Prandtl one-equation model and Spalart-Allmaras model [Spalart and Allmaras, 1992]. These models include formulations for turbulence production, transport and dissipation, making them more accurate than algebraic models. Additionally, they remain computationally inexpensive and numerically stable because there is only one additional equation to solve.

For the Prandtl one-equation model, the eddy viscosity is modelled as $\nu_T = l_m\sqrt{k}$, where k is the turbulent kinetic energy. The transport equation for k comes from the exact turbulent kinetic energy equation ($k = \frac{1}{2}\widetilde{u_i' u_i'}$) and formulated using the Reynolds stress transport equation (setting $i = j$) with only the diffusion term that needs to be simplified and modelled. This model provides satisfying results for external flows and attached boundary layers, but the constant mixing length assumption is still limiting for internal and wall-bounded flows. The Spalart-Allmaras model will be discussed in detail in the upcoming section.

3. Two-equation models: These models have two transport equations, one of them typically being turbulent kinetic energy k and the other related to turbulent length or time scale. For the $k-\epsilon$ model [Jones and Launder, 1972], the eddy-viscosity is given by $\nu_t = C_\mu k^2/\epsilon$ where C_μ is a model parameter and ϵ is the turbulence dissipation rate. Unlike the Prandtl one-equation model where ϵ was modelled, a separate transport equation for ϵ means that it is a part of the solution. Similarly, the $k-\omega$ model [Wilcox, 1988] formulates the eddy viscosity as $\nu_t = C_\mu k/\omega$ or $\omega = \epsilon/k$, which was specifically done to achieve improved performance for boundary layer flows and pressure gradients. In both cases (ω or ϵ), the transport equation is formed using physical intuition and dimensional arguments.

2.1.2 Spalart-Allmaras turbulence model

The Spalart-Allmaras (SA) model [Spalart and Allmaras, 1992] is a one-equation model where the eddy viscosity ν_T is directly related to a SA working variable $\tilde{\nu}$. This model is classified as an eddy-viscosity model, and the Reynolds stresses are defined using the Boussinesq assumption in the following manner:

$$\nu_t = \frac{\mu_t}{\rho} = \tilde{\nu} f_{v1}, \qquad f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \qquad \chi \equiv \frac{\tilde{\nu}}{\nu}, \tag{2.9}$$

where ρ is the density, $\nu = \frac{\mu}{\rho}$ is the kinematic viscosity. This formulation of eddy viscosity provides damping in regions with low turbulent Reynolds numbers. The SA working variable $\tilde{\nu}$ is related to the eddy viscosity through a function, which follows the postulated transport equation:

$$\frac{D\tilde{\nu}}{Dt} = P - D + \frac{1}{\sigma}\left[\nabla.((\nu + \tilde{\nu})\nabla\tilde{\nu}) + c_{b2}\nabla(\tilde{\nu})^2\right], \tag{2.10}$$

where P and D are the production and destruction terms, respectively, details about these terms and the constants of this model are provided in Appendix A. The transport equation Equation 2.10 is empirically derived but has been developed explicitly for aeronautical boundary layer flows. The transport variable $\tilde{\nu}$ varies linearly in the near-wall region, making it highly robust. The model performs well for simply attached flows. For a parallel flow, the model predicts a constant eddy viscosity, and therefore it does not work well for free shear layers. The SA model also has a low sensitivity to separation.

The original model was formulated in the publication by [Spalart and Allmaras, 1992], but over the years, there have been many variants that provide corrections

like adding trip terms for laminar suppression (f_{t2}-term), rotation and curvature (SA-RC) correction. These variants have been summarized by the excellent online resource by NASA (here) described in the paper by Rumsey et al. [2010]. However, the negative Spalart-Allmaras (SA-neg) model is of specific interest because the framework of this project is based on this turbulence model.

Negative Spalart-Allmaras model

The SA-neg model [Spalart and Johnson, 2012] further develops to enhance the robustness of the original SA-model. Special treatment is given to cases when $\tilde{\nu}$ becomes negative due to numerical reasons. This is done by solving a slightly different transport equation when $\tilde{\nu} < 0$, contrary to original models where the common practice was to clip the solution update. Additionally, it provides an updated definition of the modified vorticity $\widetilde{\Omega}$ to prevent negative values of $\widetilde{\Omega}$. The primary aim of the modification is to keep the original model untouched and only increase numerical robustness. It also includes the laminar suppression term (f_{t2}-term) to suppress turbulence in laminar flows. Therefore the eddy viscosity levels need to be sufficiently high in the farfield (outside boundary layer), i.e. $(\mu_T/\mu)_\infty > 0.2$ to display turbulent behaviour. The relevant terms and equations for this version have also been detailed in Appendix A.

SA-neg model is the default one-equation turbulence model in the TAU flow solver [Schwamborn et al., 2006] available at DLR. Owing to its numerical robustness and stability, this model is used as the baseline model for running the flow calculations in the current project.

2.2 TRANSONIC FLOWS FOR A 2D AIRFOIL

Shock wave - boundary layer interactions (SBLI) are among the most complex flow phenomena in fluid flow around an airfoil, especially in the transonic regime. In transonic flows, it is expected that the flow will reach supersonic speeds on the suction side of the airfoil, where high flow velocities and low pressures are seen, thus leading to shock waves. The boundary layer over the airfoil is subjected to the adverse pressure gradient caused by this shock. Additionally, the shock has to interact with layers of viscous and inviscid flow. The adverse pressure gradient causes the boundary layer to become less full and displaces the inviscid flow just above the boundary layer. If the shock is strong enough, it leads to boundary layer separation, which causes significant changes in an otherwise inviscid, attached flowfield. Such shock-induced separation potentially leads to unsteadiness on an airfoil flow, often seen in the form of buffeting on wings. These interactions get more complex if the flow is turbulent as the viscous dissipation increases and the drag over the wing is increased.

For the transonic regime, the incoming flow $M < 1$ and is accelerated over the suction side into a supersonic regime. In the supersonic region, expansion waves are generated due to the convex shape of the suction side, which further accelerates the flow and decreases the pressure. These expansion waves reflect as compression waves from the sonic line, and compression waves reflect from the airfoil as compression waves unless they are cancelled out by existing expansion waves [Babinsky and Harvey, 2011, p. 88-89]. These compression waves eventually catch up to form a normal shock, and this shock interacts with the boundary layer. This phenomenon can be viewed in Figure 2.1

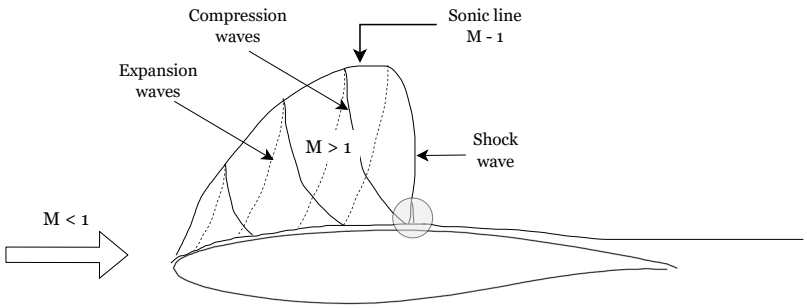


Figure 2.1: Shock wave-boundary layer interactions around a 2D airfoil in transonic flow.

The severity of the SBLI is dependent on the strength of the shock wave, which in turn depends on the increasing free stream Mach number, Reynolds number and angle of attack. Additionally, thicker airfoils lead to stronger shocks. Once the shock is strong enough, the adverse pressure gradient causes local separations or a flow breakdown until the trailing edge. These separations can be labelled as shock-induced separations, and when the flow completely separates, it is also called "shock stall". This term is acceptable as there is a sudden loss of lift and an increase in drag.

Another mechanism by which flow separation is seen in SBLI is the separation at the trailing edge. The presence of shock thickens the boundary, which makes it more sensitive to adverse pressure gradient, making it more likely to separate [Babinsky and Harvey, 2011, p. 91]. Therefore, the boundary layer may start first separating at the trailing edge as in the subsonic flow case than at the shock foot. This trailing edge separation is seen in supercritical airfoils [Schewe et al., 2003]. Figure 2.2 shows the two types of separation over a 2D airfoil in a transonic case.

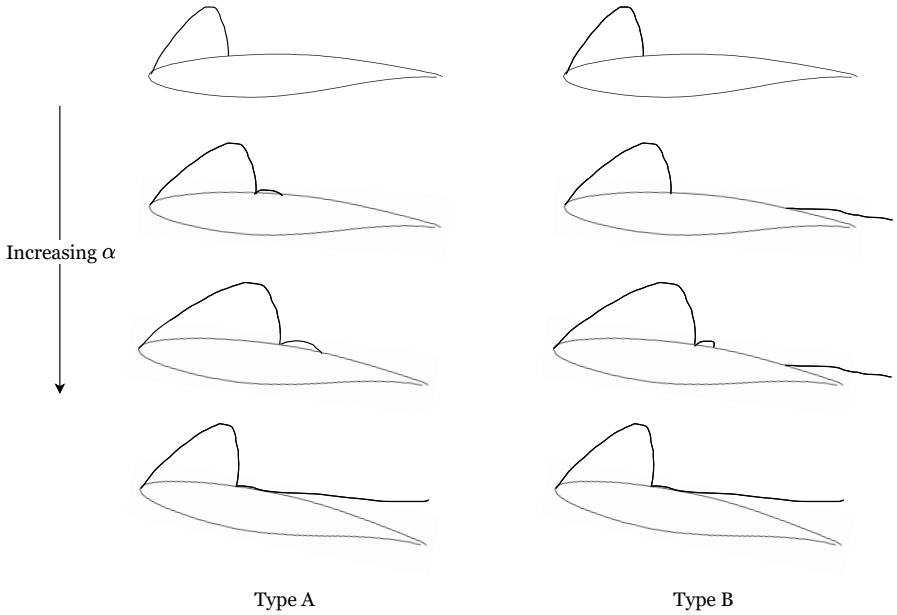


Figure 2.2: Types of shock induced separation around a 2D airfoil in transonic flow conditions. It is expected that the flow around the RAE2822 airfoil will resemble the Type A flow phenomena.

The airfoil in consideration for this thesis is the RAE 2822 airfoil. It is expected that the separation phenomena for this airfoil will resemble the Type A in Figure 2.2 as it is a sub-critical airfoil with a roof-top type pressure distribution at design conditions.

2.3 OPTIMIZATION METHODS

For a classical optimization problem, a quantity of interest (QOI) depending on some design variables is defined. This QOI needs to be optimized by varying the design variables, respecting some constraints. In majority of simulation based design optimization problem in aerodynamics, atleast one equality constraint is defined by the CFD solver based on a RANS/LES method in the loop where the design variable always has to satisfy the CFD solver governing equations while moving in the direction of optimal solution. Therefore, this section will focus on the formulation of an equality constrained optimization problem and the optimization techniques to solve them.

2.3.1 Formulation of an Equality Constrained Optimization Problem

A general unconstrained optimization problem can be defined as follows:

$$\min_x f(x), \tag{2.11}$$

where f is the objective function for the problem and x is the design variable belonging to a domain Ω. This problem has no bounds on the design variable, apart from x not deviating beyond some realistic values. To solve an unconstrained optimization problem, it is assumed that the objective function is continuous and differentiable, and the domain for design variables is closed and bounded. For a minimum to exist, the objective function has to be non-monotonic or it has to increase and decrease within the domain. At a local minimum, a necessary condition is that the first derivative of the objective function is zero or $f' = 0$. However the first derivative can be zero at a maximum or a saddle point, therefore the sufficient condition for minimum is that $f'' > 0$. For a multidimensional problem, where x is a vector of design variables, the conditions for a local minimum can be framed as the following [Papalambros and Wilde, 2000, p. 137]:

$$\text{Necessary :} \quad \nabla f = 0, \tag{2.12}$$

$$\text{Sufficient :} \quad \boldsymbol{H} = \frac{\partial^2 f}{\partial x_i \partial x_j} \text{is positive definite} \tag{2.13}$$

, where $\boldsymbol{H}$ is the Hessian matrix of the objective function. The point at which necessary condition is satisfied is also termed as the stationary point. This local minimum would be a global minimum if the objective function f is convex.

Now, consider a constrained optimization problem with multiple constraints. The simplest case for this kind of problem is an equality constrained optimization problem, where all constraints are equality constraints. This can be written as follows:

$$\min_x f(x), \quad x \in \mathbb{R}^n \quad \text{s.t} \quad h_i(x) = 0, \quad i = 1, ..., m \tag{2.14}$$

where design variables x are n in number; h_i are the independent equality constraints, m in number and $n > m$. Each equality constraint reduces the dimension of the problem. Therefore the solution of this problem can be said to exist in a feasible subspace $\boldsymbol{F}$ of the dimension $n - m = p$. Therefore, the simplest approach would be eliminating the variables in x using equality constraints and solve an unconstrained problem of dimension p. However, this is often not possible as elimination fails when variables cannot be explicitly solved from equality constraints. Additionally, in a CFD environment there is no closed form of objective function as often the simulation output is plugged in the objective function to evaluate its value. Therefore, another approach is needed to deal with the constraints.

In this thesis study, the approach that is considered is formulating the Lagrangian by using the method of Lagrange multipliers. An augmented unconstrained optimization problem is defined using the constraints in the following manner (shown for one constraint here):

$$\mathcal{J}(x,\Lambda) = \boldsymbol{f}(x) + \Lambda^T \underbrace{\boldsymbol{h}(x)}_{=0}, \quad x \in \mathbb{R}^n,\ \Lambda \in \mathbb{R}^m, \tag{2.15}$$

where Λ is the Lagrange multiplier. Now, to find the stationary point of the augmented response, i.e., the necessary condition for the local minimum, the first derivative is checked, as in Equation 2.12:

$$\nabla \mathcal{J} = 0. \tag{2.16}$$

Evaluating for Equation 2.15, this results in:

$$\nabla \mathcal{J} = \begin{Bmatrix} \partial_x \mathcal{J} \\ \partial_\Lambda \mathcal{J} \end{Bmatrix} = \begin{Bmatrix} \partial_x \boldsymbol{f} + \Lambda \partial_x \boldsymbol{h} \\ (\partial_\Lambda \Lambda^T)\boldsymbol{h} \end{Bmatrix} = \begin{Bmatrix} \partial_x \boldsymbol{f} + \Lambda^T \partial_x \boldsymbol{h} \\ \boldsymbol{h} \end{Bmatrix} = 0.$$

Therefore, the necessary condition is ultimately posed as

$$\frac{\partial \boldsymbol{f}}{\partial x} + \Lambda \frac{\partial \boldsymbol{h}}{\partial x} = 0, \quad \text{and} \quad \boldsymbol{h} = 0, \tag{2.17}$$

where second equation is the equality constraint itself. The advantage of the method of Lagrange multipliers is that now the problem can be treated as an unconstrained problem, and the derivative test (as shown above) can be directly applied. To evaluate the sufficiency condition for local minimum of $\mathcal{J}$, the bordered Hessian is formulated and its positive-definite nature is checked [Papalambros and Wilde, 2000, p. 185].

2.3.2 Optimization techniques

To solve an unconstrained optimization problem with multiple design variables, there are two major classes of optimization algorithms. This classification is based on the availability of the gradient information of the objective function:

Gradient-free methods

For a non-continuous, unpredictable design space, gradient-free methods are best suited to solve an optimization problem. They are complex to implement; however, they increase the possibility of achieving the global optimum [Skinner and Zare-Behtash, 2018]. Additionally, these methods can deal with numerically noisy optimization problems as the gradient is not a problem. This makes them a better method to use in case of discontinuities like shock-waves. For a stochastic design variable, for instance number of engines on an aircraft, gradient free methods are the only suitable class of methods. Furthermore, they do not need any *a priori* knowledge of the design space. These methods optimize several solutions in parallel and often require many simulations, often a cause of worry for a high-dimensional problem with many design variables.

The following methods, in no way an exhaustive list, have been used previously in aerodynamic optimization literature:

1. Biologically inspired methods: The algorithms for these methods take inspiration from biological processes, and typically make use of a population. These include methods like Genetic Algorithms (GA) and Particle Swarm Optimization (PSO). The major advantage of using these methods are the global optimization properties, which is difficult to achieve in gradient based methods. Additionally, these methods are easy to combine with parallel computing.

GAs are well suited for complex optimization tasks, and high fidelity codes can be adapted to them easily [Skinner and Zare-Behtash, 2018]. GAs have provided great results in aerodynamic optimization studies, but still require more number of function evaluations compared to gradient based methods. [Obayashi and Tsukahara, 1997; Zingg et al., 2008]. Additionally, the choice of population effects the quality of the solution significantly, and there is no specific way to choose the population.

2. Ensemble Kalman filter (EnKF): The iterative version of the original method was used for solving inverse problems [Iglesias et al., 2013]. The basic idea is to construct a finite dimensional space of solution from which the solution is approximated by guessing the input space which is to be applied to a fixed mathematical operator. The regularized version of this method [Iglesias, 2016] is suitable for the ill-posed field inversion problem to be dealt in this study, and thus this method deserves a separate mention. Yang and Xiao [2020] used this method to predict the correction term for the turbulence/transition model for the field inversion in their implementation of FIML method.

3. Other methods include Simulated Annealing, Random walk methods, Nelder-Mead Simplex methods, etc.

Gradient-based methods

These methods work on the underlying assumption that the gradient information of the objective function is available for all the independent variables involved. These methods are well suited to find the local optimum solutions using the derivative based conditions discussed in Section 2.3.1; however finding the global optimum can be difficult and dependent on the design space. Gradient based methods require an understanding of the design space, which is necessary to supply these algorithms with a good starting point. If the starting point is not appropriately defined, and the design space has multiple local minima, then gradient based methods will converge to a local minimum and struggle to escape the surrounding of the initial guess.

The major advantage gradient based methods provide is the low computational cost even when handling a large number (in order of $10^3 - 10^4$) of design variables. This makes them suitable for aerodynamic applications, given that the sensitivity information is available. Methods for sensitivity analysis will be discussed in Section 2.3.3. In a field inversion context where a spatial field for a correction term in the turbulence model is sought, gradient-based algorithms prove to be useful as the number of parameters are equal to the number of grid points and scale with the grid refinement.

Gradient based methods can be classified on the basis of the order of the gradient information required:

1. First order methods: These methods require the first derivative of the objective function, ∇f. The simplest method here is the Steepest Gradient Descent algorithm, which can be summarized in the following manner:

$$x^{i+1} = x^i + \alpha \nabla f(x^i). \tag{2.18}$$

First, the search direction is identified by movement towards the largest decrease in f, which is given by ∇f. Then, a stepsize α is either chosen or calculated using line search to achieve a desired amount of reduction in the objective function. The $i + 1^{th}$ update in design variable x is then achieved by using Equation 2.18. This method often results in an efficient "zigzagging" behavior near optimum values. Improved methods can be achieved by either

including a momentum term [Qian, 1999] or by taking history of the search direction into account in the conjugate gradient method [Fletcher and Reeves, 1964].

2. Second order methods: These methods require the second-order derivative of the objective function. The most popular method here is the Newton's method, where a local approximation of the objective function is formed using Taylor series expansion, and then the first order derivative test is applied to the approximation to find the search direction of largest decrease. This direction results is $-\boldsymbol{H}^{-1}\nabla f$, where H is the Hessian matrix of the objective function. This results in the following update step:

$$x^{i+1} = x^i - \alpha \boldsymbol{H}^{-1}\nabla f(x^i), \quad \text{where } H = \frac{\partial^2 f}{\partial x_j \partial x_k} \tag{2.19}$$

Newton method has quadratic convergence, which is the best possible. However there are many limitations; calculating costly Hessian matrices (especially for high number of design variables), bad convergence far from optimum and no remedy to deal with singular/not positive-definite Hessian [Papalambros and Wilde, 2000, p. 151]. Methods like Levenberg-Marquardt method [Moré, 1978] and Trust region methods have been proposed to improve the robustness of Newton methods.

3. Quasi-Newton methods: These methods employ the best possible of both methods, by approximating the Hessian matrix. Therefore, a faster calculation of the approximate Hessian matrix is possible which preserve convergence qualities of Netwon methods. Examples include Davidson-Fletcher-Powell (DFP) method [Davidon, 1991] and Broyden, Fletcher, Goldfarb, Shannon (BFGS) method [Fletcher, 2013].

2.3.3 Sensitivity analysis and Adjoint method

The gradient based methods discussed in the previous section are dependent on derivative information of the objective function. The existence of the derivative information is checked using the sensitivity analysis of the objective with respect to the design space. Skinner and Zare-Behtash [2018] distinguished the sensitivity analysis methods in four classes:

1. finite-difference methods,
2. complex-step finite difference methods,
3. automatic differentiation, and
4. analytical methods (direct or adjoint).

Finite difference and complex-step finite difference methods are easy to implement, but can get complicated for a large number of design variables. These methods are often used in low-fidelity computational codes. Out of the remaining two classes of methods, analytical methods, in particular adjoint methods have found a lot of application in aerodynamic optimization. Peter and Dwight [2010] surveyed sensitivity analysis approaches for aerodynamic optimization, and concluded that the discrete adjoint method is the most practical algorithm in modern-day RANS solvers. Therefore, this approach will be discussed below in detail.

Discrete Adjoint method

The main advantage of discrete adjoint method is that the gradients are consistent with the discretized form of the governing equations. The discrete adjoint formulation given by Giles and Pierce [2000] will be presented here from a Lagrangian view

point. Consider the augmented unconstrained objective function for a constrained optimization problem in aerodynamics, as in Equation 2.15:

$$\mathcal{J}(\boldsymbol{x}, \boldsymbol{U}) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{U}) + \Lambda^T \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{U}), \quad \boldsymbol{x} \in \mathbb{R}^n, \ \Lambda, \boldsymbol{h} \in \mathbb{R}^m, \tag{2.20}$$

where $\boldsymbol{U}$ are the state variables at each discrete grid point, which can be a vector of flow variables at each grid point, $\boldsymbol{h}$ is the set of governing equations and boundary conditions to be satisfied, and $\boldsymbol{x}$ is the design variable to be optimized. We require the design derivative of augmented response, i.e., $\frac{d\mathcal{J}}{dx_j}$:

$$\frac{d\mathcal{J}}{dx_j} = \frac{\partial \boldsymbol{f}}{\partial x_j} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{U}} \frac{\partial \boldsymbol{U}}{\partial x_j} + \Lambda_i^T \left(\frac{\partial h_i}{\partial x_j} + \frac{\partial h_i}{\partial \boldsymbol{U}} \frac{\partial \boldsymbol{U}}{\partial x_j} \right) \tag{2.21}$$

Rewriting the equation to group all terms with state derivatives gives,

$$\frac{d\mathcal{J}}{dx_j} = \frac{\partial \boldsymbol{f}}{\partial x_j} + \Lambda_i^T \frac{\partial h_i}{\partial x_j} + \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{U}} + \Lambda_i^T \frac{\partial h_i}{\partial \boldsymbol{U}} \right) \frac{\partial \boldsymbol{U}}{\partial x_j} \tag{2.22}$$

If the number of design variables (x_j) is high and the number of state variables ($\boldsymbol{U}$) are high as well, then one can imagine that calculating the $\frac{\partial \boldsymbol{U}}{\partial x_j}$ becomes a difficult task. Therefore, in adjoint approach the Lagrange multiplier Λ is chosen in such a manner that the state derivatives vanish. This can be done by satisfying the following:

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{U}} + \Lambda_i^T \frac{\partial h_i}{\partial \boldsymbol{U}} = 0 \implies \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{U}} \right)^T \Lambda_i = - \left(\frac{\partial h_i}{\partial \boldsymbol{U}} \right)^T \tag{2.23}$$

Once the Lagrange multipliers are Λ_i is known from Equation 2.23, the required design derivative ($\frac{d\mathcal{J}}{dx_j}$) can be calculated from Equation 2.22:

$$\frac{d\mathcal{J}}{dx_j} = \frac{\partial \boldsymbol{f}}{\partial x_j} + \Lambda^T \frac{\partial h_i}{\partial x_j} \tag{2.24}$$

Adjoint methods become attractive when the number of constraint equations h are significantly less than the number of state variables $\boldsymbol{U}$, which is usually the case in aerodynamic optimization. Evaluating $\frac{\partial h_i}{\partial x_j}$ is much cheaper than $\frac{\partial \boldsymbol{U}}{\partial x_j}$. Equation 2.24 is the only equation needed to be solved to calculate the final sensitivity.

2.4 MACHINE LEARNING AND FEATURE ENGINEERING

2.4.1 Introduction to Artificial Intelligence and Machine Learning

Artificial intelligence was born in the 1950s when computer science engineers believed that a computer could be made to "think" and automate tasks usually performed by humans. AI is a general field that encompasses machine learning and deep learning; however, it also includes many other approaches that do not use any learning approach. Initial approaches to artificial intelligence involved programmers defining "a sufficiently large set of explicit rules for manipulating knowledge" [Chollet, 2018], famously known as *symbolic AI*. This approach was popular until the 1980s; however, it was only useful to solve well-defined and logical problems like playing chess. For more complex problems we face today like image classification, language translation, in which figuring out explicit rules is reasonably tricky, the approach of *machine learning* (ML) was developed, which replaced symbolic AI.

Machine Learning (ML) approaches *learn* how to do a given task by defining the

rules itself by looking at the available data, contrary to symbolic AI where the rules have to be explicitly defined. ML consists of a set of methods that discover patterns in data to generate rules and then use these rules to make predictions or take decisions. Machine learning systems differ from classical programming as they are *trained* rather than programmed. An ML system looks at the data, makes statistical relations, and then the rules to automate the task.

For this *learning* process to work, three things are required. Firstly, input data points are needed. Secondly, examples of the expected data output are needed (in a *supervised* problem). Finally, a measure to see whether the ML algorithm is working is needed. This is usually a parameter that indicates the difference between expected and actual output, and it drives the *learning* process by adjusting the algorithm. In some ML models, a parameter called a *loss function* is defined, which is related to the difference of predictions from the actual target. Further discussion of loss functions will take place in the upcoming sections.

As explained by Francois Chollet in his famous textbook [Chollet, 2018], machine learning can be summarized as *"searching for useful representations of some input data, within a predefined space of possibilities, using guidance from a feedback signal."*. There are various classical machine-learning approaches, and they can be broadly classified into the following categories:

- Supervised learning: The input data is mapped to known targets, and this drives the learning process. Examples: classification, regression, etc.
- Unsupervised learning: The targets are not known beforehand. Search for interesting patterns from data is carried out without any known targets. Examples: dimensionality reduction and clustering.

To achieve a well-performing ML model, the model should learn from the data and create general rules so that their performance is good on data that is unknown to the model. The data available can be divided into three sets:

1. Training data set: The set used to train the model, i.e. the model learns and tries to formulate the rules based on the training data set it is exposed to by varying its parameters.
2. Validation data set: The set used to put this training's performance under the microscope, and the configuration of the model is tuned (by varying its hyperparameters) if it does not perform well on this data set. Thus, the primary purpose of this dataset is the evaluation of the model to make it better by checking for *underfitting* or *overfitting*.
3. Testing data set: Finally, when the engineers feel that the ML model has achieved a certain level of desired performance, it can be ultimately put to the test on the testing data set. This data set is totally new to the model; thus, it is a true test of the generalizing power of the ML model.

There are a few things that need to be kept in mind before selecting the data sets from the available data for the machine learning model:

- The individual data sets should be representative of the data at hand.
- The data should not be shuffled in time if the problem at hand is a temporal prediction. This means that the training data set should not contain any data from the future because the exercise aims to obtain a prediction from the data at hand at present.
- The data sets should not be too repetitive, or training and validation data sets should not contain the same data points as it brings redundancy to the data. A certain level of repetition is tolerated; however, if the network spots the same patterns everywhere, it will only produce those patterns as an output.

As discussed before, a good ML model has a good *generalization* capability. A better generalization can be achieved by tweaking the model accordingly, and this tweaking process is known as *optimization* of the model. At the beginning of the training, the model is newly exposed to the training data, and it starts to learn about it. This learning is indicated by a lowering loss value by applying the model to the validation data set. As the loss shows a downward trend, we can say that the model is still in progress to learn all the relevant patterns in the training data. This situation is called an *underfit*. However, after a certain number of iterations on the data, the generalization ability stops improving, and the loss value trend first gets constant and then degrades. In this situation, one can say that the model has learned all relevant patterns related to the training data. However, they become irrelevant for the new data (validation data) as it loses its generalization factor because it has memorized the patterns in training data too specifically. This situation is known as *overfitting*. It is important to know when a model overfits because the point of best performance is also known as it will be achieved just before it starts to overfit. There are various ways to prevent overfitting in different ML models. Some of the common methods to prevent overfitting have been mentioned below:

- Increase training data: Increasing training data would increase the generalization capability of the ML model, as it will have more patterns to remember statistically speaking.
- Adding dropout: This technique involves randomly forcing the values of training data to zero (known as *dropping out*) to introduce noise which inhibits the ML model from memorizing a pattern that it has seen repetitively.

Therefore to approach every problem in Machine Learning, a universal workflow can be adopted, summarized in the points below:

1. Define the type of problem at hand (classification, regression)
2. Assemble and preprocess the data, define the training, validation, and testing data sets.
3. Choose a measure of success of the model, i.e. an appropriate loss function
4. Develop/choose an ML model that predicts better than the statistical probability of an event happening.
5. Scale up the model further by varying its parameters so that it overfits. Thus the point of best performance is known.
6. Regularize the model and tune its hyperparameters to improve the model further
7. Test the model on anonymous data to see how well the model performs.

2.4.2 Neural Networks

Neural network models are machine learning algorithms that work based on learning in successive layers, creating intermediate data representations to achieve the desired output. They find application in deep learning; deep learning is mostly done through layers of neural networks stacked on top of another. These layers can go in orders of tens or hundreds, and each layer has a meaning no matter how abstract it is to humans.

Figure 2.3 demonstrates how a typical neural network algorithm works. The input is indicated as X, and it goes through layers of data transformations via neural networks which have certain weights associated with the input. The prediction given after it goes through these layers is indicated using Y'. The actual target is defined as Y. Using a Y and Y', a *loss function* is calculated, which indicates how far the predictions are from the actual target. This loss function generates a score that is fed as a feedback signal to the *optimizer*. The optimizer updates the weights in the layers to reduce the score given by the loss function for the next cycle. The process

mentioned above is at the heart of neural network models and is used to reach the desired output. The elements mentioned in Figure 2.3 and neural networks will be discussed in more detail in the upcoming sections.

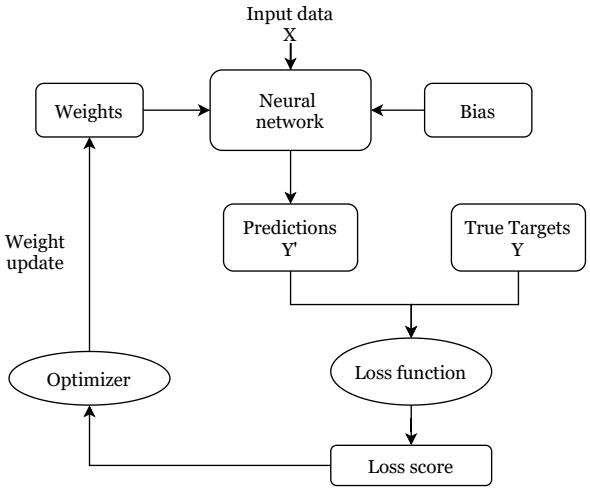


Figure 2.3: Working of a supervised neural network algorithm

The basic unit of a neural network is called a neuron or a node. The anatomy of a neuron can be seen in Figure 2.4. A neuron performs a very straightforward operation; it receives several inputs x_i and associates a weight w_i to all the inputs. An activation function f is then applied on the weighted sum of these inputs plus a bias b.

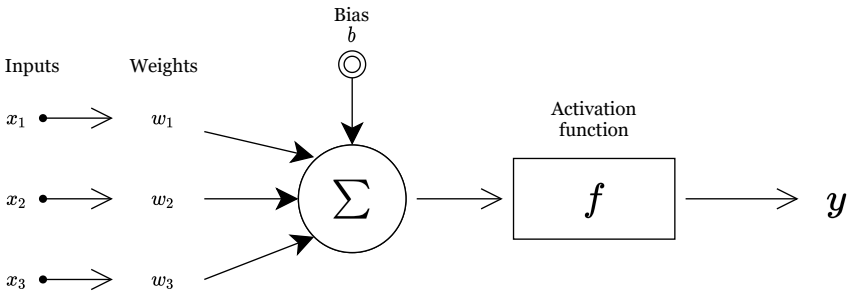


Figure 2.4: Basic unit component of a Neural network: A neuron

Therefore, for N inputs, this neuron has $N+1$ parameters to be tuned, i.e. the N weights and one bias function. The output of the neuron is:

$$y = f\left(\sum_{i=1}^{n} w_i \times x_i + b\right) \tag{2.25}$$

The activation function f is used to introduce non-linearity into the output of a neuron. A summary of activation functions used in deep neural networks can be found in the work by Lau and Hann Lim [2018]. Typical activation functions include the *tanh* function, the *Sigmoid* function:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.26}$$

and the *Rectified Linear Unit (ReLu)* function:

$$f(x) = \max(0, x) \tag{2.27}$$

The neural network model can be viewed as a collection of neurons. Each neuron is a mathematical function in which the weights and biases (the trainable parameters) are associated with the inputs. During the training process, the *parameters* that vary are the weights of the neuron, which are driven by the ML algorithm. The neural network can be designed by deciding the various number of neurons (i.e. size) in a layer or choosing the number of layers. Varying this *hyperparameter* affects the performance of the model. They can be tuned if the engineer thinks the model is not performing well on the validation data set. Picking the appropriate network architecture is learned with experience; however, some best practices and rules of thumb should be followed. The simplest example is stacking multiple layers linearly, mapping a single input to a single output. The topology of the network defines the possible solutions it can give, or the *hypothesis space*. When the network architecture is defined, the tensor operations acting on the inputs are specified, and these operations are used to map the input to the output.

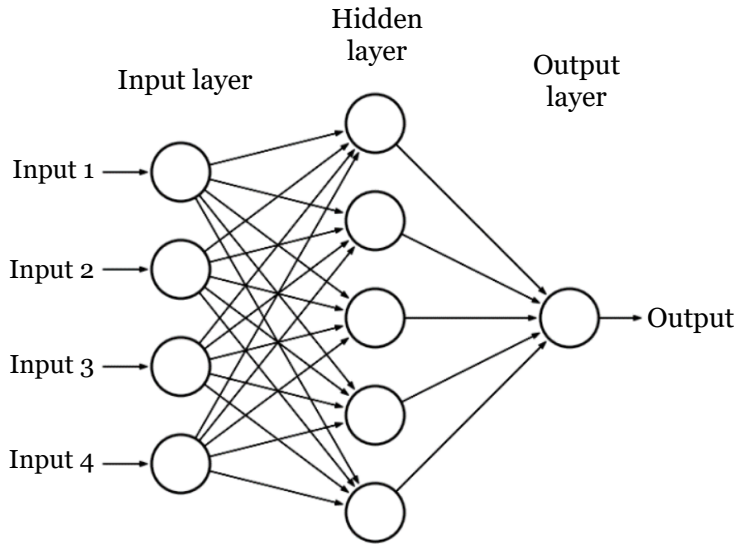


Figure 2.5: Fully connected multi-layer perceptron (MLP), where each neuron is connected to the next layer

One typical architecture is the Multi-Layer Perceptron (MLP) as seen in Figure 2.5. MLP is a feedforward network that consists of several layers of neurons, in which each layer is fully connected to the following one. Each neuron in these layers will receive several inputs and compute the previous equation with its own weights and a bias. Here, fully connected layers are displayed, where neurons of the first layer are all connected to the second one. Consequently, for P neurons with N inputs (either the initial inputs, or ones from a previous layer), this layer will contain $(N+1)P$ parameters to be tuned.

Loss function and Optimizers

In a machine learning algorithm, the *loss function* (also known as *objective function*) is a quantity that needs to be lowered during training the ML model. The lowering value indicates the measure of success of the ML model. The value and trend of the loss function act as a feedback signal to the network. According to this signal, the *optimizer* updates the network. More information about gradient-based optimizers used in neural network models can be found in the review paper by Ruder [2016]. Here, the discussion in the context of an optimizer will focus on a stochastic gradient descent (SGD) algorithm.

The choice of the objective function in accordance with the problem is essential to get the correct output. The objective function should be correlated to the desired output. The optimizer takes the shortest route possible to minimize the loss, and an improperly defined loss function may lead to important patterns being missed. Additionally, the loss function depends strongly on the problem (regression, classification, etc.). A typical example for regression problem is the *Mean Square Error (MSE)*, defined as

$$\mathbf{L}(Y', Y) = \frac{1}{N}\sum_{i=1}^{N} |Y'_i - Y_i|^2, \tag{2.28}$$

where Y' is the prediction from the neural network, and Y is the target or the *ground truth*. Loss functions ($\mathbf{L}()$) can be minimized by varying the value of Y', which is done by varying the value of weights in the layers. To find the minimum value of the loss function, we move in the direction where its derivative is zero. This approach is used to find the appropriate values of the weights, and we can try to move in the direction where the loss value decreases. For a weight W of the network, the gradient $\frac{\partial \mathbf{L}}{\partial W}$ is computed. The parameter weight W is then updated in the direction of the gradient with a relaxation term α (called the learning rate) so that new weight is found using:

$$W_{new} = W - \alpha \frac{\partial \mathbf{L}}{\partial W}. \tag{2.29}$$

This update of weights is called the *gradient descent*. Computing the derivative $\frac{\partial \mathbf{L}}{\partial W}$ is a computationally expensive task for the neural networks due to the large number of parameters involved. This update can be achieved using the *Backpropagation algorithm* Rumelhart et al. [1986], which is inherently based on the chain rule for differentiation. A neural network that has multiple outputs may have multiple loss functions (one per output). However, the gradient-descent process must be based on a single scalar loss value; for multi-loss networks, all losses are combined (via averaging) into a single scalar quantity.

The computational cost may still be high if the derivative is calculated together for all input samples or all the weights. Thus, data can be chosen in small batches at random, and this is called *mini-batch stochastic gradient descent*. Here, *stochastic* refers to the fact that these batches are chosen at random. A *batch* is a collection of a small number of samples. The gradient descent will be performed after the batch is evaluated, averaging the descent over the batch samples. The *batch size* is the number of samples used to approximate the gradient. Small batch size will produce a noisy optimization process as the gradients are produced in different directions, making the descent noisy. A large batch size will average this noise; however, it will create memory issues due to the large number of gradient calculations involved. In practice, the largest batch size allowed by the GPU memory is chosen. The ML algorithm runs over the entire dataset samples by iteratively loading batches of data where the gradient calculation and parameter update is run. This strategy is denoted as the mini-batch SGD. Each complete data cycle that covers the whole data set is called an *epoch*. Many epochs (10-1000 in magnitude) are performed during the optimization process, and the data is reused several times. The number of epochs depends on:

- number of samples in the training data set,
- the size of the neural network, i.e. the number of parameters to be tuned,
- the intrinsic difficulty of the problem,
- the desired accuracy level expected from the network (more accurate often requires more epochs).

2.4.3 Feature Engineering

An additional thing to do before feeding the data to the ML model is to preprocess it so that the ML model understands it well. In the case of neural networks, various techniques can be employed to make the data suitable to the ML model. The data first needs to undergo *vectorization*; all inputs should be in the form of tensors and be in the floating-point format. *Feature engineering* is a broad activity involving extracting relevant features from the existing data desirable for the machine learning algorithm. It can be viewed as hardcoded transformations applied to the data before being fed to the model to make it more representative and improve the model's performance. The discussion in this section will be focused on continuous numerical input features and numerical output variables, which are typical of a regression problem. A well functioning feature engineering pipeline includes the following activities:

1. Feature extraction: This is the process of extracting the important features from the data using a-priori knowledge of the data, machine learning algorithm, and expected outcome. Thus, generating new features by either applying mathematical operations on one or more existing features can be classified in this activity. For the context of this thesis, features describing phenomena related to turbulence modelling and transonic flows will be desirable. More details about feature extraction for the FIML approach will be presented in Chapter 6.

2. Feature cleaning: This activity involves identifying and treating specific data points in the data, which may be incorrect or irrelevant. Cleaning includes treating outliers in the data, missing values, redundant data or removing noise. This activity requires domain knowledge to identify erroneous observations. Typically, the treatment involves deleting these data points from the training data or replacing/imputing them with a pre-decided value like zero, ones, mean or extreme values. In other cases, it is expected that there will be missing values, and in that case, the missing values should be left as it is.

3. Feature transformation: Transforming features so that they are amenable to the machine learning algorithm is the primary purpose of this activity. The input features may vary in their magnitudes, and algorithms sensitive to the magnitude of features (like neural networks) do not perform well on such data. Thus, they need to be transformed so that they are on a similar scale, which is done by using one or more of the following methods:

   - Scaling: Changing the scales of the feature x_i using a mathematical operation like normalization (scaling using min, max value of the feature, Equation 2.30) or standardization (scaling by removing the mean $\bar{x_i}$ and scaling by the standard deviation σ_{x_i} , Equation 2.31). When dealing with physical variables, this can be achieved by making the features non-dimensional by choosing appropriate normalizing factors. One instance of this in data-driven turbulence literature is the transformation is given by Ling and Templeton [2015].

$$x_{i,norm} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{2.30}$$

$$x_{i,std} = \frac{x_i - \bar{x_i}}{\sigma_{x_i}} \tag{2.31}$$

   - Changing the distribution: An input feature x_i with a highly skewed or shifted probability distribution can be transformed into a desirable distribution form for the machine learning algorithm. The change of distribution can be done by either applying mathematical operators like the $ln(x_i)$, $1/x_i$, x_i^n, or by applying transformers like Power transform (to

make the distribution more Gaussian) and Quantile transform (impose a desirable distribution) readily available in scikit-learn [Pedregosa et al., 2011].

4. Feature selection: The process to select the most relevant features from the full feature subspace. The process can also be termed supervised selection when the selection is made keeping the target variable in mind. Selecting the appropriate number of features is important to determine the balance between the information more features provide and the computational cost to train a machine learning algorithm on a high number of input features. Various feature selection techniques can be viewed in Figure 2.6. More detail about these techniques is given in the next section.

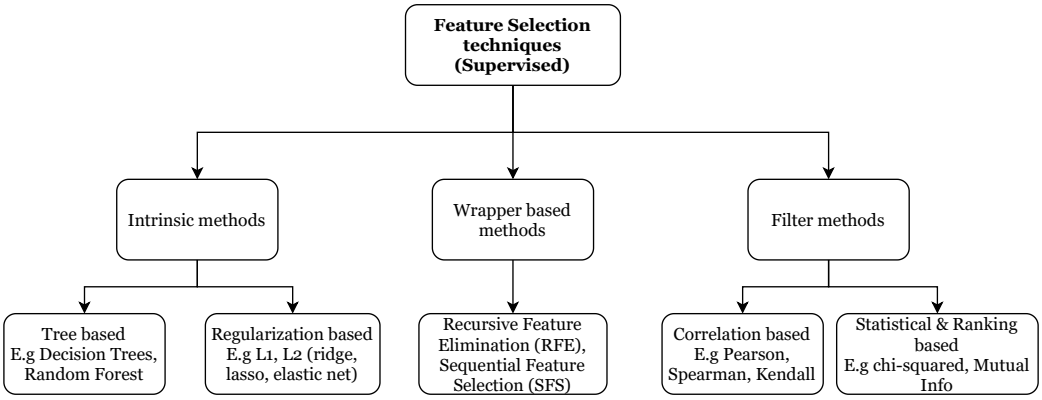


Figure 2.6: Supervised feature selection techniques keeping the target variable in mind

Feature selection techniques

There are three major classes of feature selection techniques:

1. Filter based methods: These methods rely only on the characteristic of the features and are independent of the machine learning algorithm in use. Additionally, they are quick and straightforward in application, making them the go-to methods in any feature engineering pipeline. For a numerical input feature and output variable, correlation-based methods like the Pearson product-moment correlation (r, Equation 2.32) and Spearman's rank correlation coefficient (ρ, Equation 2.33) which calculate the correlation between the input and the output variable. Pearson's coefficient makes several assumptions about the variables, like both the variables should have a normal probability distribution and should have a straight-line relationship. Spearman's coefficient becomes an attractive option where no assumptions about the distribution are needed.

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}} \tag{2.32}$$

where x_i, y_i are the two features and $\bar{x}$, $\bar{y}$ are their means, respectively

$$\begin{aligned} \rho &= 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \\ d_i &= \text{difference between the two ranks of each observation} \\ n &= \text{number of observations} \end{aligned} \tag{2.33}$$

2. Wrapper based methods: These methods wrap around an estimator, which could be any appropriate machine learning algorithm, and evaluate subsets

of the full feature space based on a quality indicator to determine the best subset of features. Usually, the number of required features in the subset is user input. Wrapper based methods tend to result in better predictive accuracy as they involve the estimator the user wants (for instance, neural networks in this case). Another advantage they offer is that they consider the interaction of features in the subspace with each other. Examples of these methods are: Recursive Feature Elimination (RFE) and Sequential Feature Selection (SFS) [Ferri et al., 1994] algorithms, which are readily available in scikit-learn [Pedregosa et al., 2011] and mlxtend [Raschka, 2018] python libraries.

3. Embedded methods: Feature selection is a bi-product of some machine learning algorithms, and these algorithms are said to have embedded feature selection methods. These methods consider the interaction of features like wrapper based methods and are faster and more accurate than filter methods. Furthermore, they are less prone to overfitting. However, the reproducibility of results from these methods is an issue. Examples of these machine learning algorithms with these methods are tree-based algorithms which give a rank of features an output like Decision trees and Random forests, and regularization based algorithms that promote sparsity like L1, L2 regularization (ridge, lasso, elastic net).

3 PARADIGM FOR FIML APPROACH

This chapter lays down the paradigm for the Field Inversion and Machine Learning (FIML) approach. The discussion starts from the closure problem statement in governing equations for fluid flows and how machine learning can address this problem. This discussion provides the preface for model-consistent training and its importance in data-driven turbulence modelling, leading to the emergence of FIML. Consequently, the FIML approach used in the current study is summarized to provide a larger picture before a deep dive further in the report. Finally, the applications of the FIML approach in literature have been discussed along with the evolution of classical FIML towards new, improved, tightly coupled learning approaches.

3.1 PROBLEM STATEMENT

Consider the governing equations for a turbulent flow (for e.g. Navier Stokes equations) in the form:

$$\mathcal{R}(\mathbf{U}) = 0, \tag{3.1}$$

where $\mathcal{R}$ is the system of governing equations with independent flow variables $\mathbf{U}$. On applying the ensemble averaging operator on Equation 3.1 as done for RANS equations, it can be represented as the following:

$$\widetilde{\mathcal{R}(\mathbf{U})} = 0 \longrightarrow \mathcal{R}(\widetilde{\mathbf{U}}) + \mathcal{N}(F(\mathbf{U})) = 0. \tag{3.2}$$

On applying this averaging process, one obtains two terms; first a system of governing equations in the resolved flow variables $\widetilde{\mathbf{U}}$ (i.e. $\mathcal{R}(\widetilde{\mathbf{U}})$) and the second term is used to represent the terms which are *unclosed* in $\widetilde{\mathbf{U}}$, i.e are also dependent on $\mathbf{U}'$. Here, $\mathcal{N}$ is a known mathematical operator and quantity F is used to represent all the unclosed terms in resulting model. All current turbulence models follow the strategy of constructing a closed approximation $F_m \approx F$ (dependent only the $\widetilde{\mathbf{U}}$ and not the $\mathbf{U}'$), and some other secondary variables $\widetilde{s_m}$. Thus, the system of equations for the turbulence model can now be formulated as

$$\mathcal{R}(\widetilde{\mathbf{U}_m}) + \mathcal{N}(F_m(\widetilde{\mathbf{U}_m}, \widetilde{s_m})) = 0, \tag{3.3}$$

$$G_m(\widetilde{\mathbf{U}_m}, \widetilde{s_m}) = 0, \tag{3.4}$$

where $G_m(\widetilde{\mathbf{U}_m}, \widetilde{s_m}) = 0$ is a set of transport equations to solve for the secondary variables $\widetilde{s_m}$. All the modelling assumptions are employed in the terms F_m and G_m while developing a turbulence model.

Over the previous years, the development of traditional turbulence models has been slow due to various reasons summarized by Spalart [2015]. These models rely on data for calibration but are more in line with the modeller's hypothesis. Attempts to improve on this have been made by developing data-driven turbulence models, which by nature are more dependent on the data. The aim of data-driven turbulence modelling is to improve upon the representations of F_m and G_m, by obtaining variables $\widetilde{\mathbf{U}}, \widetilde{s}$ from high-fidelity dataset like DNS or experiments, and then obtaining

the model variables $\widetilde{\mathbf{U}_m}, \widetilde{s_m}$ from them. Thus, the modeler hopes that the representations of F_m and G_m using $\widetilde{\mathbf{U}_m}, \widetilde{s_m}$ from the data will be more accurate. Various ways to represent data-driven model equations have been explored and summarized by Duraisamy et al. [2019], but the focus of the current study is to do this representation using machine learning (ML). These representations are discussed in the next subsection.

3.2 REPRESENTATION OF MODEL EQUATIONS USING ML

The machine learning model is represented as $\delta_m(\widetilde{\eta_m}; w)$ in this work, where $\widetilde{\eta_m}$ are the flow features obtained from $\widetilde{\mathbf{U}_m}, \widetilde{s_m}$ used as input to the learning algorithm and w are the parameters of the learning algorithm. There are various methods to represent the closure terms using δ_m. They can either represent the closure term fully or a part of the closure term. The method in the scope of this document is when δ_m is used to model the impact of unresolved variables $\mathbf{U}'$ on the modelled closure terms $F_m(\widetilde{\mathbf{U}_m}, \widetilde{s_m})$. There are various methods to represent the closure terms using the δ_m. They can either represent the closure term fully or a part of the closure term. Some examples in literature are presented below.

One of the first attempts was made by Tracey et al. [2013], where they improve the Reynolds stress tensor τ_m by modelling the error of the Reynolds stress anisotropy using a Kernel regression based ML model trained on a DNS dataset. Duraisamy and Durbin [2014] and Duraisamy et al. [2015] considered a broad range of features to augment the transport equations (G_m) rather than the Reynolds stresses directly during the learning process. Another approach that tries to model the Reynolds stress anisotropy using invariants of velocity gradient tensor (like strain rate, vorticity) was introduced by Weatheritt and Sandberg [2017], where they use symbolic regression with genetic programming. Schmelzer et al. [2020] use a sparse linear regression method which formulates tensor polynomials of the Reynolds stress tensor from a library of candidate functions, promoting sparsity and amenability.

In each example presented, a ML algorithm δ_m is used to represent a part of the $F_m(\widetilde{\mathbf{U}_m}, \widetilde{s_m})$. From this representation, the ML algorithm needs to be trained on high-fidelity data so as to identify the model which agrees with the representation. This discussion will be continued in the next subsection.

3.2.1 Training of model equations using ML

The ML model $\delta_m(\widetilde{\eta_m}; w)$ can be incorporated in the RANS equations to result in the following machine learning augmented model:

$$\mathcal{R}_a(\widetilde{\mathbf{U}_m}, \widetilde{s_m}, \delta_m) = 0, \tag{3.5}$$

where $\mathcal{R}_a$ is the ML augmented RANS model. This model, after training, can be used to give improved predictions of the flow. The methodology of this training will be discussed in this subsection.

The training process is usually posed as a supervised learning problem. Given the target δ obtained from high fidelity data, the ML model representation posed as the following problem:

$$\min_{w} \mathbf{L}[\delta, \delta_m(-; w)], \tag{3.6}$$

where $\mathbf{L}$ is a loss function driving the training process, the form of input features is still to be decided, and w is the set of varying ML parameters to reach a lower

loss score. Duraisamy [2020] in his review paper summarizes that this training procedure has been done in two broad ways in literature:

1. **A-priori training**: Here, the target of the ML model δ for the training is obtained from a high-fidelity dataset like DNS. At the training time, the input features to the model $\widetilde{\eta}$ are obtained from DNS as well. This training runs separately to the flow solver operating on governing equations and can be written as follows:

$$\min_{w} \mathbf{L}[\delta, \delta_m(\widetilde{\eta}; w)]. \tag{3.7}$$

   At prediction time, the ML model δ_m is incorporated in the RANS equations resulting in the following augmented model:

$$\mathcal{R}_a(\widetilde{\mathbf{U}_m}, \widetilde{s_m}, \delta_m(\widetilde{\eta_m}; w)) = 0, \tag{3.8}$$

   which uses the model features $\widetilde{\eta_m}$ at prediction time. The primary advantage of this approach is that the training is non-intrusive and physics-informed constraints can be directly introduced at this stage. Therefore this approach was used by Tracey et al. [2013] and Weatheritt and Sandberg [2017] in their work. However, the consistency with the model is a problem due to different features being used at training and prediction time. This is a major issue in ML-augmented RANS models [Duraisamy, 2020].

2. **Model consistent training**: To ensure consistency, the input features at training and prediction time need to be in the same space, i.e. either the model $\widetilde{\eta_m}$ or the high-fidelity data $\widetilde{\eta}$ space. As the prediction of the augmented model happens in the model space, the training also needs to be done in the model space, i.e. using the features $\widetilde{\eta_m}$. To obtain these features and enforcing consistency, a solution of an inverse problem is required, which reduces the *discrepancy* between the model and the data. This is done by incorporating the model $\mathcal{R}_a$ at the training time.

   The solution of the inverse problem gives a model-augmentation field δ_m^i using which augmented predictions can be produced. Therefore, this approach is also known as the field inversion approach (FI). Suppose a sparse output variable Y^i is obtained from a DNS field, and the same output variable can be obtained using δ_m^i and is called Y_m^i. The inverse approaches [Duraisamy et al., 2015] can be posed as the following:

$$\min_{\delta_m^i} \mathcal{L}[Y^i, Y_m^i(\delta_m^i)], \quad \text{s.t} \quad \mathcal{R}_a(\widetilde{\mathbf{U}_m}, \widetilde{s_m}, \delta_m(\widetilde{\eta_m}; w)) = 0, \tag{3.9}$$

   where $\mathcal{L}$ is the loss function driving the inverse problem. This inversion process can be applied at multiple design points to obtain multiple δ_m^i (referred as δ_m^*) to be used as targets in the ML process, and multiple corresponding input features $\widetilde{\eta_m^*} = [\widetilde{\eta_m}_1, ..., \widetilde{\eta_m}_N]$. Finally the supervised learning can be performed as follows:

$$\min_{w} \mathbf{L}[\delta_m^*, \delta_m(\widetilde{\eta_m^*}; w)]. \tag{3.10}$$

   This approach has been coined as the *Field Inversion and Machine Learning* (FIML) approach and has been used by many researchers in data-driven augmentation of turbulence models [Parish and Duraisamy, 2016; Singh et al., 2017b; Singh, 2018; Ferrero et al., 2020; Jäckel, 2020]

The FIML approach has been chosen as the method to be used for data-driven augmentation of the SA-neg model for the current project.

3.3 CHOSEN FIML APPROACH

The proposed implementation of the FIML approach in this project follows the work of Parish and Duraisamy [2016], where the discrepancy term is introduced in the transport equation ($G_m(\widetilde{\mathbf{U}_m}, \widetilde{s_m}) = 0$ in Equation 3.4) for the working variable ($\tilde{\nu}$) of the negative Spalart-Allmaras (SA-neg) turbulence model. The approach is summarized in Figure 3.1.

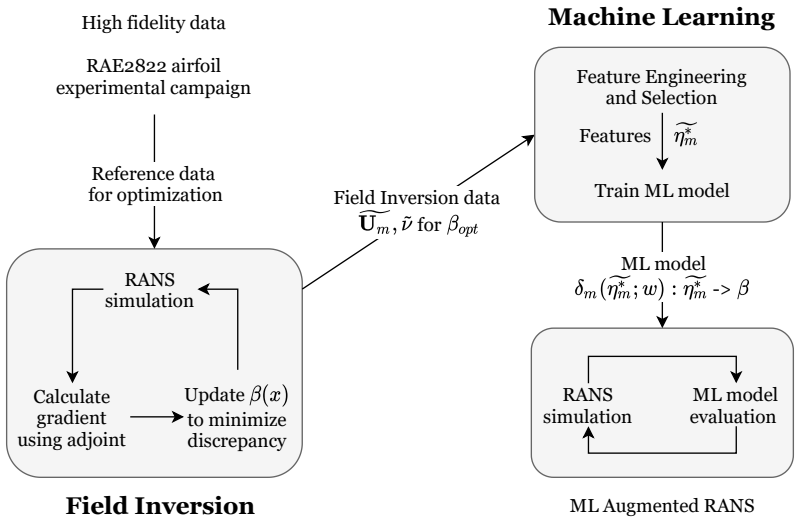


Figure 3.1: Summary of the chosen Field Inversion and Machine Learning approach

Consider the following system of equations for an augmented SA-neg model following from Equation 3.5:

$$\mathcal{R}_a(\widetilde{\mathbf{U}_m}, \tilde{\nu}, \beta) = 0, \tag{3.11}$$

where the $\tilde{\nu}$ is the secondary variable ($\widetilde{s_m}$), and β is the machine learning augmentation term, formerly represented as δ_m. For this implementation, the augmented transport equation of the $\tilde{\nu}$ of the SA-neg model is given by

$$\frac{D\tilde{\nu}}{Dt} = \beta(\widetilde{\mathbf{U}_m}, \tilde{\nu})P(\widetilde{\mathbf{U}_m}, \tilde{\nu}) - D(\widetilde{\mathbf{U}_m}, \tilde{\nu}) + T(\widetilde{\mathbf{U}_m}, \tilde{\nu}), \tag{3.12}$$

where P, D and T are the production, destruction and transport terms, respectively. The discrepancy β is introduced as a functional correction term multiplied by the production term P. For the baseline model, $\beta = 1$, causing no effect to the turbulence model. The β field is assumed to be a function of the SA-neg working variable $\tilde{\nu}$ and resolved model flow variables $\widetilde{\mathbf{U}_m}$.

To obtain β_{opt}, the field inversion problem is posed as an optimization problem. In the inverse problem for this system an optimal mapping for the discrepancy β_{opt} is sought which maps the augmentation field to the high-fidelity data supplied. The optimization problem in this case is given by:

$$\beta_{opt} = \underset{\beta}{\operatorname{argmin}}\ \mathcal{L}, \quad \text{s.t} \quad \mathcal{R}_a(\widetilde{\mathbf{U}_m}, \tilde{\nu}, \beta) = 0, \tag{3.13}$$

where $\mathcal{L}$ is the loss function driving the optimization problem. The flow solution containing flow variables $\widetilde{\mathbf{U}_m}, \tilde{\nu}$ corresponding to the β_{opt} is then used as training data for the machine learning step.

The field inversion problem is solved for the flow cases selected to be used for machine learning. Each flow case will have its own output β_{opt} and flow variables $\widetilde{\mathbf{U}_m}$.

These flow variables are then processed into input features $\widetilde{\eta_m^*}$ which are gathered from an extensive literature survey and selected using feature selection techniques. From the shortlisted $\widetilde{\eta_m^*}$, an accurate and robust machine learning (ML) model δ_m is to be developed. The aim is to identify patterns in the spatial β field and its relation to the set of features $\widetilde{\eta_m^*}$.

$$\delta_m(\widetilde{\eta_m^*}; w) : \widetilde{\eta_m^*} \longrightarrow \beta \tag{3.14}$$

Neural networks are selected as the default choice of the machine learning algorithm for this study. Their capability to learn non-linear function approximations is unparalleled compared to any other ML algorithm, which is a desirable property for current purpose of approximating a RANS model closure. Universal approximation theorem [Hornik et al., 1989] states that any function may be approximated using a sufficiently deep neural network. However, this does not mean that the complexity of the neural network should be mindlessly increased. Determining the network architecture and optimising the learning hyperparameters is a crucial activity of this project to achieve a robust ML model. Finally, the developed ML model will be connected to the flow solver to augment the SA-neg turbulence model. The ML-augmented turbulence model will be applied to the test cases identified during the database analysis stage and further points unrelated to the training database to test the ML-augmented turbulence model's predictive ability.

3.4 APPLICATION OF FIML APPROACH IN LITERATURE

The FIML approach explained in the previous section is also referred to as classic FIML approach. This approach has the following benefits:

- Model consistency: ML training is done with features in the model space by using the inversion solution, enforcing consistency with the flow solver.
- Facilitating sparse use of high-fidelity data: Only a limited number of high-fidelity data points are needed as the feature information from limited inversion solutions is enough to train a generalized Machine Learning model.

The classical FIML approach has been applied extensively on various test cases, some of which are listed below:

- Singh [2018] in his PhD thesis applies this framework on adverse pressure gradient flows on a curved bump using $k-\omega$ model and separated flows on S809 airfoil using SA model.
- Yang and Xiao [2020] apply this framework on the 4-equation $k-\omega-\gamma-A_r$ transition model, and the test case was transitional flow over airfoils.
- Köhler et al. [2020] used their $k-\omega$ based FIML implementation to improve the prediction of separated flows on 2D curved periodic hills.
- An Spalart-Allmaras model based FIML implementation was applied to turbomachinery flows by Ferrero et al. [2020].
- Singh et al. [2017a] augmented the Spalart-Allmaras model for predictions of flows involving shock-boundary layer interactions. Transonic flow cases over geometries like axisymmetric bump, oblique shock-boundary layer interactions and shock train flows were tested.
- Köhler et al. [2020] accurately predicted the separation and reattachment in wall-bounded flows by applying the FIML approach on the two-equation k-ω.

From the literature survey it was found that the classical FIML framework has not been extensively applied to the shock-induced flow separation case over a 2D airfoil. The author proposes to apply the FIML approach on the RAE2822 airfoil to improve the predictions of the shock-induced boundary layer separation over a range of Mach numbers (M), Reynolds Number (Re), and angle of attacks (AoA). The only similar application was made by Holland [2019], where he applied the FIML approach on RAE2822 profile for a flow case where shock boundary layer interaction leads to a strong pressure gradient but not flow separation.

A movement from classical FIML to approaches where the learning process is coupled with the inference process (field inversion in classical FIML) was proposed by Holland [2019]. Embedded FIML and Direct FIML approaches were introduced, in which ML training was driven by the loss function of the inversion process; with each approach having a different coupling procedure. The main advantage these approaches offer is that they enable learning from several field inversion problems simultaneously without any problems in convergence. Another advantage the tight coupling offers is to reduce the loss of information of learning a low-dimensional ML model from a high-dimensional augmentation field.

A further step towards a more integrated approach was presented by Srivastava and Duraisamy [2021], which aims to inform the low-dimensional feature space (used to generate the ML model) using underlying physics of the problem, by carefully constructing the map between feature space to augmentation space. This approach is called Learning and Inference as- sisted by Feature-space Engineering (LIFE), and aims to provide the modeler more control of the feature space. The modeler has the freedom to locally update the feature space so as to generate a more robust and general model. The test case for this application was bypass transition.

4 EXPERIMENTAL DATABASE DESCRIPTION

A high-fidelity database is required for any data-driven turbulence modeling activity, which may be acquired from either experiment, field measurements, or accurate numerical simulations like DNS or LES. The author was provided with an experimental aerodynamic database for 2D airfoils to undertake this master thesis project. The RWC.01 database was generated by Airbus in 2016 and was supplied to the author by DLR. Additionally, the author was provided with the results of a 2D CFD campaign used to validate the RWC.01 database. This chapter describes the procedure employed during the experimental campaign, its assumptions and their implications to this project. Furthermore, the CFD campaign is also described in this chapter.

The primary purpose of the experimental reference database is to understand the scope and applicability of the data for the proposed FIML approach. The database is used to select the flow cases to undergo the field inversion procedure, the output of which is used as the training data for the machine learning algorithm. This chapter discusses the scope and significant conclusions from the database, followed by the discussion of the methodology devised by the author to select the flow cases for the field inversion.

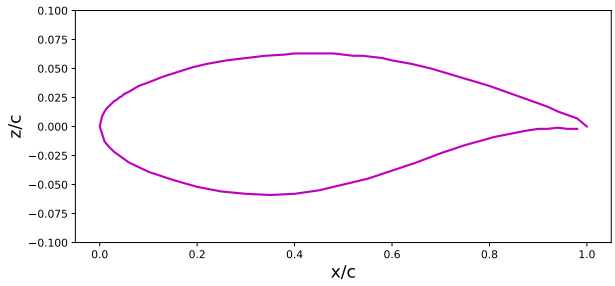


Figure 4.1: RAE2822 airfoil 2D profile. The x-y scaling in this figure is unequal.

4.1 EXPERIMENTAL CAMPAIGN DESCRIPTION

The Airbus RWC.01 database gathers experimental aerodynamic data acquired in 2016 using the pilot facility of the European Transonic Wind Tunnel (pETW) for a series of 2D airfoil sections. The objective of the experiment was to provide new airfoil validation data for various 2D-airfoils in the experiment. In particular, the reference RAE2822 section geometry equipped with thicker trailing edge was tested to cross-check results with *legacy data*, which was collected by NATO Advisory Group for Aerospace Research and Development (AGARD) in the AGARD 138 report in the 1970s [Cook et al., 1979]. The legacy data had been used to refine and calibrate numerical codes at Airbus; however, it remains limited in its scope. The work done by AGARD was repeated and extended to a broader range of operating conditions during the Airbus experimental campaign. Information and figures only for the RAE2822 airfoil campaign (named RWC.01_1) were provided to the author. RAE2822 airfoil at design conditions is a rear-loaded, sub-critical airfoil with a roof-

top type pressure distribution. The 2D profile of RAE2822 airfoil can be seen in Figure 4.1.

4.1.1 Facility and Campaign Description

The pilot facility of the European Transonic Wind Tunnel (pETW) is approximately 1/9 scale of the main ETW facility, achieving similar test conditions to that of the main transonic cryogenic ETW facility. Flows of Mach number 0.15-1.3 can be tested with Reynolds number up to 230 million per meter. Two test sections are available: a) slotted walls, i.e., holes in the top and bottom walls, and b) solid walls, i.e., with all walls closed (achieved by aluminium tape on slots of the walls). The size of the test section is the same in both configurations. The experiment for RAE-2822 airfoil was done in two campaigns:

1. Campaign 1 (RWC.01_1-01): Test section with slotted walls for both fixed and free transition on the airfoils.
2. Campaign 2 (RWC.01_1-02): Test section with solid walls for the free transition case only.

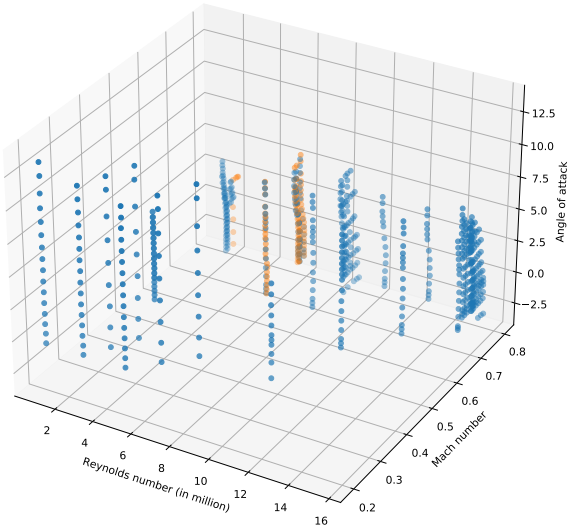


Figure 4.2: Design range of available wind tunnel measurement data for test points of Campaign 1 (RWC.01_1-01) on the RAE2822 airfoil. Orange points are for fixed transition wind tunnel runs, and blue points are for free transition.

The method of fixing the airfoil transition is detailed in the next subsection about the model description. The purpose of the transition fixing activity was to replicate the conditions of the AGARD experiment done by Cook et al. [1979]. Figure 4.2 shows the test points of Campaign 1 (RWC.01_1-01), where the orange points are test points with a fixed transition, mainly around Re of 6 million. The blue points are test points with a free transition. The data available was over a large range of test section flow conditions, with M varying between 0.2-0.96, Re varying between 2.7-15.7 million and AoA between -2.5 ° to 13 °. The variation in Re was achieved through varying temperature (total temperature range of 115K - 296K) and pressure (total pressure range of 145 kPa to 296 kPa). The measurements were acquired with a sample rate of 1 Hz. The exact test matrix for the wind tunnel campaigns has been redacted for this report.

4.1.2 Model and Test Section Description

The model setup of the experiment can be viewed in Figure 4.3. The span length of the model is 271 mm, and the chord length is 90mm. The aspect ratio (AR) of the model is approximately 3, and suppliers of this database expect limited 3D effects. Therefore, the measurements obtained at the Main station (shown in Figure 4.3 in red dotted line) are treated as measurements for a 2D profile of the airfoil section for the purpose of this study. However, it is worth mentioning that Garbaruk et al. [2003] performed an extensive RANS study on RAE2822 airfoil for an AR=3 model and they refute that the 2D assumption is correct.

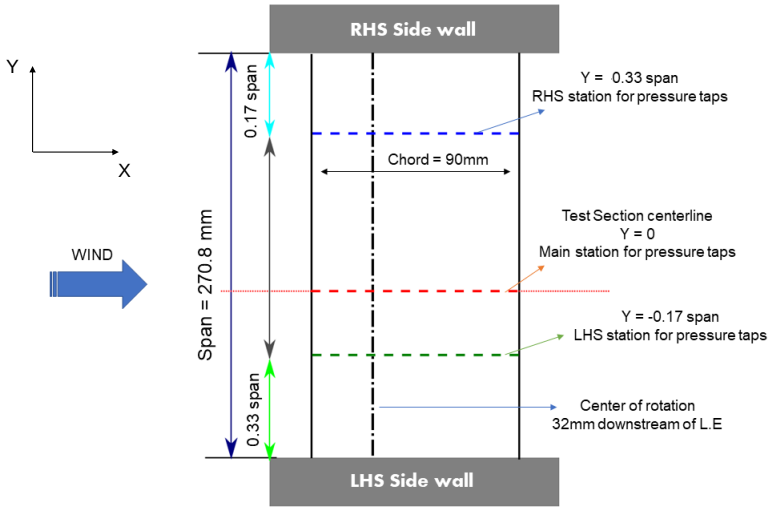


Figure 4.3: Experimental setup of the RAE2822 model in the pETW wind tunnel. The wind is blowing in the x-axis, spanwise direction is the y-axis and z-axis is perpendicular to the diagram. The pressure tappings are at three sections of the airfoil, namely Main (red), Side 1 (blue) and Side 2 (green). Details about the tappings are available in Table 4.1

A total of 100 pressure tappings are distributed at three test sections of the model, locations of which are detailed in Table 4.1. For pressure measurements in the test section, the walls of pETW are equipped with surface static pressure tappings on the top, bottom and LHS walls. The model is not equipped with a force balance, so the force measurements must be derived from the pressure measurements. For varying the angle of attack of the model, the model setup is capable of remote control system incidence at a point 32 mm from the LE along the centerline. This point corresponds to the intersection of the red dotted and black dotted lines in Figure 4.3.

	Span location	Tappings		
	(in % span)	total	upper section	lower section
Main (red)	50% - test section centerline	74	46	28
Side 1 (blue)	17% - from RHS side wall	13	9	4
Side 2 (green)	33% - from LHS side wall	13	9	4
	Total	**100**	**64**	**36**

Table 4.1: Pressure tapppings details on the RAE2822 model in the pETW test section

Transition fixing was achieved by using CAD-cutout dots at 3% x/c from the leading edge on the upper surface and 5% x/c on the lower surface. The location

and dimensions of the dots were precisely the same as the experiment done to produce the legacy data, with a dot diameter of 1.3mm and dot height of 36μm.

4.1.3 Available Data and Other Considerations

The measurement points in the test matrix for these campaigns are defined in terms of polars. Thus, at each Mach number and Reynolds number value, the measurements are recorded for a set of angles of attack. The type of transition is also fixed beforehand. At each measurement point in the test matrix, the pressure measurements are recorded at all the three sections of pressure tappings detailed in Table 4.1 and at all the surface static pressure tappings on the on top, bottom and LHS walls. The example of one such measurement can be viewed in Figure 4.4, where isentropic Mach number distribution over the chord of the airfoil is plotted. Pressure coefficient values are also available at each location. Furthermore, attributes like total force coefficients, moment coefficients, dynamic pressure, reference area, etc., are also available at each point.

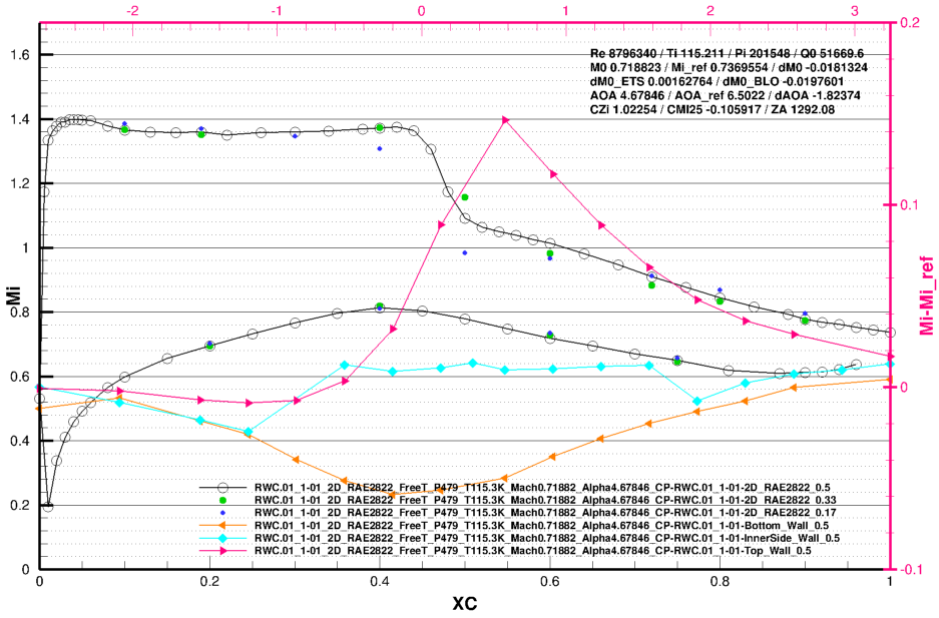


Figure 4.4: Example of the data available in the database. The figure shows the isentropic Mach distribution over the RAE2822 airfoil for a given point with the approximate flow conditions $M = 0.718$, $Re = 8.7$ million and $AoA = 4.67°$. The grey line with dots represents the measurements at the Main section, blue dots represent the Side 1 measurements, and green dots represent Side 2 measurements. The coloured lines give the measurements for the surface static pressure tappings.

The green dots in Figure 4.4 are the pressure measurements at Side 2 section, which is a 33% span distance away from the wall. The green dots are in good agreement with the pressure measurements at the Main section. However, the blue points or the pressure measurements at the Side 1 section, which is a 17% span distance away from the wall, do not agree with the other two sections. Thus, it can be concluded that far enough from the test section walls, the measurements are effectively the same. Therefore, the assumption of limited 3D effects mentioned in Section 4.1.2 is considered valid.

In Campaign 1 (RWC.01_1-01), the data is available for fixed and free transition cases. The effect of fixing the transition on the airfoil can be viewed in Figure 4.5. For the fixed transition case (in red), there is an expected effect on the pressure coefficient around the transition location on the upper (3% x/c from LE) and lower

surface (5% x/c from LE). These are viewed as "kinks" in the pressure distribution on both surfaces, which are not visible for the free transition case. The kink is larger than expected, which may be due to over tripping the boundary layer. The wall normal at the kink location becomes larger than the boundary layer thickness, which causes a strong input of turbulence. The fixed transition case is characterised by a lower "rooftop", and the shock is further towards the leading edge. These effects will be considered while choosing which data points to use for the data-driven modelling activity.

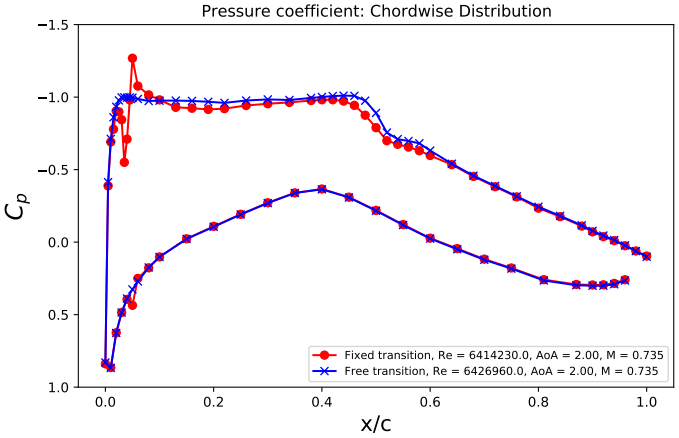


Figure 4.5: Comparison of the fixed vs free transition case for a similar flow condition in the test section. The figure shows the pressure coefficient distribution over the RAE2822 airfoil at the Main section for a given point with the flow conditions M = 0.735, $Re \approx 6.42$ million and $AoA = 2^{\circ}$. The red line with dots represents the measurements for fixed transition at locations mentioned in Section 4.1.2, and the blue line with the crosses represent measurements with the free transition.

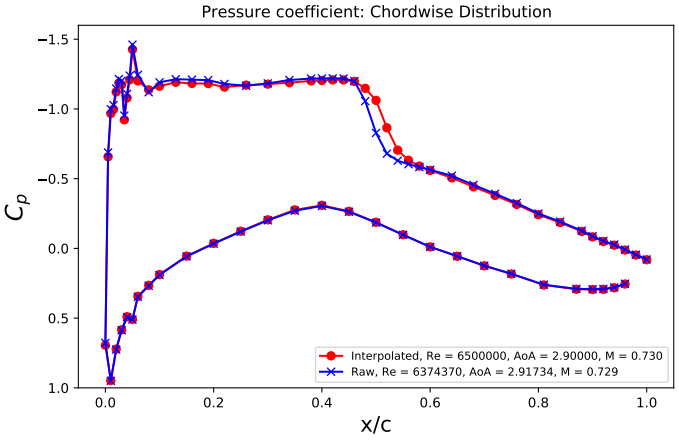


Figure 4.6: Effect of interpolation on the available experimental data. The figure shows the pressure coefficient distribution over the RAE2822 airfoil at the Main section. The blue line with crosses represents the raw measurements at M = 0.729, and AoA = 2.91734°, and red line with dots represents the measurements interpolated to a nominal M = 0.730, and AoA = 2.9°. Both data points are for the fixed transition case.

Corrections to the Mach number and angle of attack were derived using a series of tests. Firstly, empty test section calibrations were done for both slotted and solid wall test sections. This was followed by the derivation of wall effects due to the

blockage caused by the RAE2822 airfoil model. The details of how these corrections were applied have not been mentioned in this document. However, it is worth mentioning that the magnitude of corrections derived for the solid walls is more than the slotted walls primarily due to the thicker boundary layer developing for the solid walls, causing more blockage and streamline curvature effects. Therefore, there is an inclination only to use slotted wall test section results while using the data from the wind tunnel experiments.

For each flow condition, the data is collected at a sampling rate of 1Hz. The raw recorded data has been interpolated to nominal Mach numbers and angles of attack and is available as a separate database. The data is first collected using a pitch pause technique, with 12 data points with slight variations in the angle of attack at a given pause point. This data is first averaged for the 12 data points, providing a measurement at an angle of attack. This data is then interpolated to rounded Mach number values. The interpolation process has its limitations as it averages out the raw measurements, which can be a problem, especially in the regions of the non-linear behaviour of pressure like the shock location. This can be viewed in Figure 4.6, where the shock location (or the sudden increase in pressure coefficient value) for the interpolated data (in red) is shifted further down the chord than the raw data (in blue) for a very similar flow condition. These differences between raw and interpolated data can be as big as the difference between two different wind tunnel entries. Therefore, if one wants to use data points for flow conditions which have shock wave related phenomena, then it is better to use the raw, non interpolated data as it is more representative of the actual flow phenomena.

4.2 CFD DATA SUPPLIED

2D Free-Air CFD data was provided for the isolated section of the RAE-2822 airfoil to supplement the experimental database provided. This CFD data was used at all stages of the wind tunnel testing campaign:

- Before testing: design the test matrix for the wind tunnel campaign and evaluate loading to define the design range limits
- During testing: monitor the wind tunnel results to match expectations, to help identify issues with the sensors
- After testing: checking the validity of the CFD results, assessing the validity of the wind tunnel corrections.

The simulations are done using the TAU flow solver. They use SST turbulence models, which is different from the baseline model intended for this study, i.e. SA-neg model. The simulations are performed consistently with the same single-grid U-RANS based solution procedure - with no convergence criteria to stop the simulation. The simulations are done for a wide range of flow conditions, with 21 M values varying between 0.1-0.78, 6 Re values varying between 2.7-17.0 million and 91 AoA values between -3 $^\circ$ to 6 $^\circ$ with a 0.1° increment. Therefore, there is quite an overlap with the design range of the wind tunnel cases.

Importantly, these simulations are done using a fully turbulent flow assumption. Therefore, no transition models were required to estimate the transition location. This becomes a consideration while comparing the CFD results with the pETW wind tunnel results as it naturally has a transition from laminar to turbulent flow over the airfoil surface.

4.2.1 pETW and CFD Data: Pairing Procedure

The CFD data provided has been paired with the wind tunnel results on the basis of their "closeness" to the pressure distribution (or the isentropic Mach number M_I distribution). In Figure 4.7, it can be seen that for a given wind tunnel run (grey line with circles), there are three lines associated with three different CFD runs. The dotted red line is strictly at the same design condition as that of the corrected flow conditions of the wind tunnel run. The other two dotted lines, i.e. blue and green, are paired with this wind tunnel run based on a surrogate CFD database. For all the CFD runs conducted, a surrogate surface is created based on a similarity factor (SF) defined as follows:

$$SF = \sqrt{\frac{f}{g}} \tag{4.1}$$

$$f = \sum_{j=1}^{n} w_j \left[M_{I_{j_{CFD}}} - M_{I_{j_{WTT}}} \right]^2, \quad g = \sum_{j=1}^{n} w_j \left[M_{I_{j_{WTT}}} \right]^2 \tag{4.2}$$

where $M_{I_{j_{WTT}}}$ corresponds to the isentropic Mach number at the sensor j of the wind tunnel data, and w_j corresponds to the weight given to a sensor based on its surface weighted L2 distribution. $M_{I_{j_{CFD}}}$ corresponds to the value of isentropic Mach number from the CFD runs at the same location to that of the sensor.

The CFD run with the lowest value of the similarity factor was paired with the wind tunnel results. This selected CFD run could have different values of both M and AoA. The green line in Figure 4.7 corresponds to the case where a few sensors were blanked due to questionable accuracy in the post shock flow (called "Blanking-ON"), and the blue line corresponds to all sensors active (called "Blanking-OFF"). The "blanking" of the sensors is achieved by multiplying the w_j by zero. In Figure 4.7, the "Blanking-ON" case has a smaller SF value. However, this does not mean that this will be true for all flow cases, there are many instances where "Blanking-OFF" case had a smaller SF value with the wind tunnel results.

The paired CFD run has a slightly different M and AoA value to that of the wind tunnel run, which in effect acts as a Mach number and Angle of attack correction applied effectively by a surrogate-based optimization. The suppliers of this database are actively trying to correlate these CFD based corrections to the actual wind tunnel corrections that were applied at the stage of wind tunnel experiments.

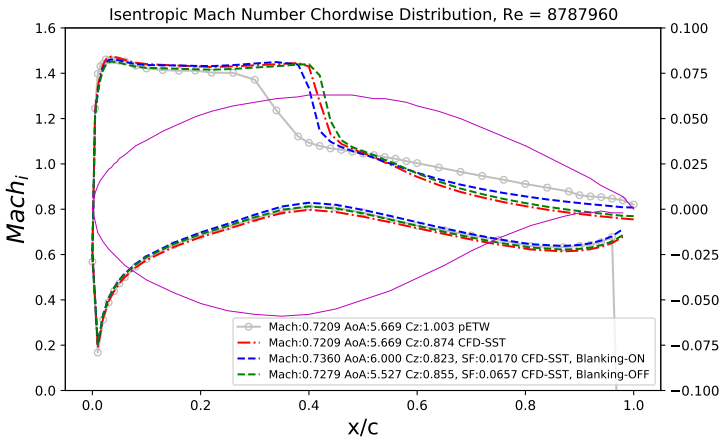


Figure 4.7: Isentropic mach distribution from Wind Tunnel measurements and the paired CFD results for a selected wind tunnel run point (M = 0.7209, AoA = 5.669, Re = 8787960)

4.3 PETW, CFD, LEGACY DATA COMPARISON

In effect, the author has three data sources to compare for understanding the flow on the RAE2822 airfoil section: a) Legacy data: NATO Advisory Group for Aerospace Research and Development (AGARD) experiment done by Cook et al. in the AGARD report 138, b) pETW data: The experimental Airbus RWC.01 database acquired in the pilot facility of the European Transonic Wind Tunnel (pETW) detailed in Section 4.1 and c) CFD data: The CFD campaign done by Airbus for the validation of the RWC.01 experimental campaign detailed in Section 4.2. The comparison of the three sources can provide the following insights:

- deeper understanding of the physical phenomena over the RAE2822 airfoil section for different flow conditions,
- comparison of the experimental setups between pETW and AGARD campaigns, which may cause differences in the resulting measurements,
- correlation of the CFD simulations with the two experimental campaigns, and understanding the reasons for differences (like modelling assumptions), and,
- information about the scope of these data sources, and which one is most suited to be used for the current data-driven activity.

Case no.	Mach number	Angle of attack (in degrees)	Re. no. (in million)
6	0.725	2.92	6.5
7	0.725	2.55	6.5
8	0.728	3.22	6.5
9	0.73	3.19	6.5

Table 4.2: AGARD report 138 test cases with comparable test points in pETW database

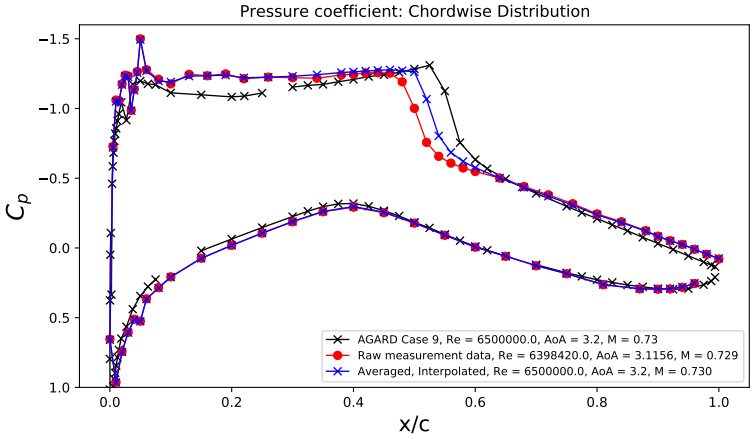
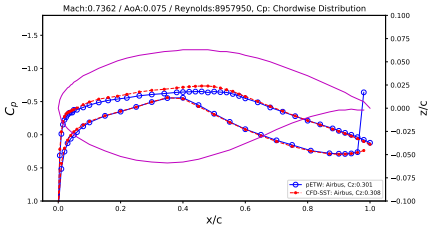


Figure 4.8: Comparison of famous AGARD legacy data case 9 with corresponding pETW data. The figure shows the pressure coefficient distribution over the RAE2822 airfoil at the Main section with fixed transition. The black line with crosses represents the AGARD case 9 measurements, and blue line with crosses represents the measurements interpolated to a nominal M = 0.730, and AoA = 3.2°, the same as AGARD case 9. Red line with dots represents the raw measurements to the closest flow conditions to AGARD case 9. Missing measurements for AGARD data are the points not reported in the original source Cook et al. [1979].

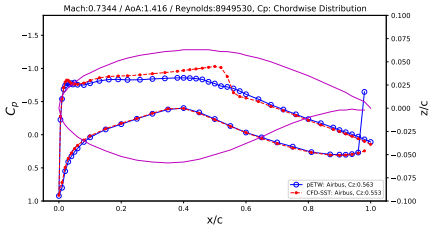
The author observed that Airbus chose four data points for the pETW database at which the flow conditions were comparable to the famous AGARD cases in the legacy data. These cases have been detailed in Table 4.2. Figure 4.8 compares the

pressure coefficient distribution for the AGARD case 9 with the corresponding averaged and interpolated pETW measurement, along with the nearest raw measurement. Assuming that each test campaign has been fully corrected, one would expect a better match despite the differences in model and wind tunnel test sections. The differences between the results are not easy to identify, as the experimental assumptions behind the legacy data are unknown. The difference can originate from model quality, manufactured geometry or wind tunnel correction methodology. The author does not have enough information to make a sure comment.

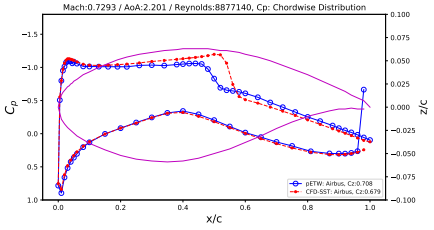
Figure 4.9, Figure 4.10 and Figure 4.11 show a comparison between the pETW data and CFD data for a chosen set of angles of attack for three different Mach numbers. Consider the blue line in all figures, which corresponds to the pETW result. In Figure 4.9, on increasing the angle of attack, the shock seems to get stronger and more developed. The presence of a pronounced "rooftop" or a sudden change in pressure indicates the presence of a strong shock wave. Furthermore, the shock shifts further upstream on increasing the angle of attack, as can be seen moving from Figure 4.9a to Figure 4.9d. The shock is also responsible for the formation of a separation bubble, which goes on an increase in size with increasing incidence. After a certain angle of attack, the separation bubble bursts and the flow is fully separated until the trailing edge. This causes an upstream shift of the shock location, which can be viewed in Figure 4.9f. The flow was confirmed to be separated at the trailing edge for the high angle of attack in Figure 4.9f by observing the skin friction coefficient (c_f) values to be zero beyond 40% of the chord length.



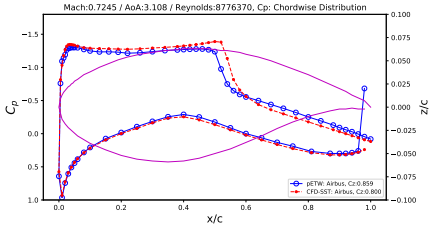
(a) $AoA = 0.075°$



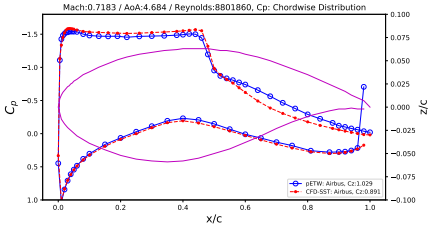
(b) $AoA = 1.416°$



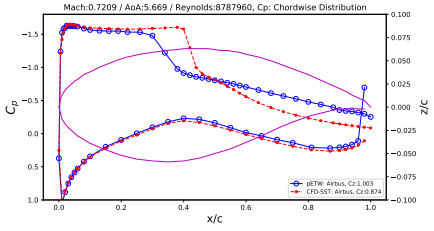
(c) $AoA = 2.201°$



(d) $AoA = 3.108°$

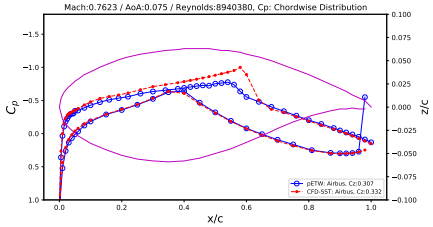


(e) $AoA = 4.684°$

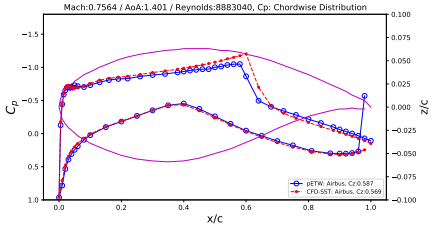


(f) $AoA = 5.669°$

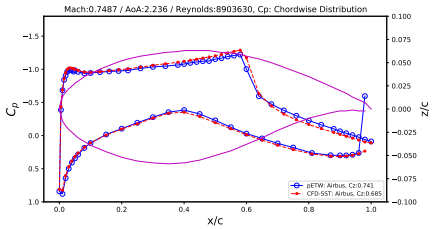
Figure 4.9: CFD vs pETW data pressure coefficient distribution for a polar at $M = 0.728$ and $Re = 8.86$ million. The blue line with circles represents the pETW measurements, and red line with dots represents the CFD data at same conditions. The actual values for M and Re may vary slightly in the wind tunnel, and have been reported in the figures.



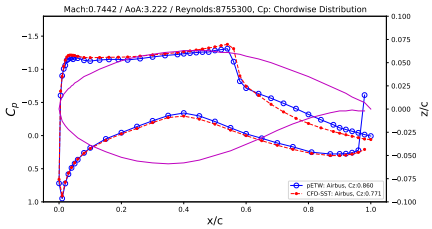
(a) $AoA = 0.075^\circ$



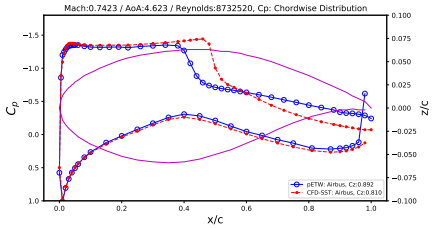
(b) $AoA = 1.401^\circ$



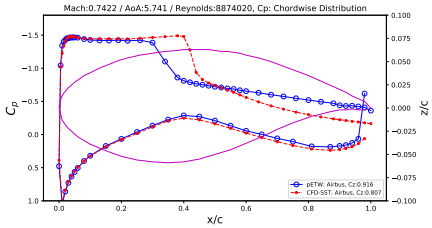
(c) $AoA = 2.236^\circ$



(d) $AoA = 3.222^\circ$

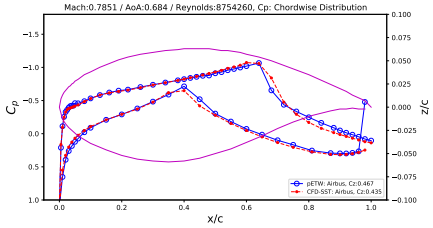


(e) $AoA = 4.623^\circ$

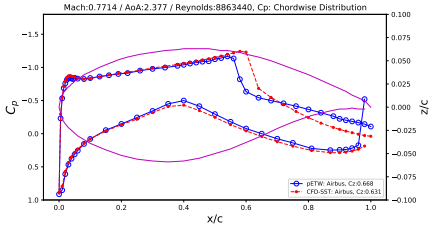


(f) $AoA = 5.741^\circ$

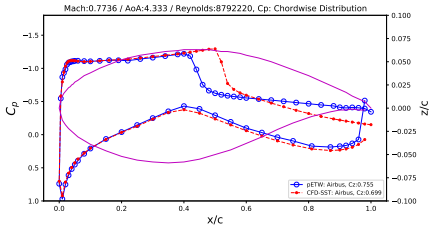
Figure 4.10: CFD vs pETW data pressure coefficient distribution for a polar at $M = 0.750$ and $Re = 8.88$ million. The blue line with circles represents the pETW measurements, and red line with dots represents the CFD data at same conditions. The actual values for M and Re may vary slightly in the wind tunnel, and have been reported in the figures.



(a) $AoA = 0.684^\circ$



(b) $AoA = 2.377^\circ$



(c) $AoA = 4.333^\circ$

Figure 4.11: CFD vs pETW data pressure coefficient distribution for a polar at $M = 0.783$ and $Re = 8.92$ million. The blue line with circles represents the pETW measurements, and red line with dots represents the CFD data at same conditions. The actual values for M and Re may vary slightly in the wind tunnel, and have been reported in the figures.

In Figure 4.10, the Mach number has been increased to $M = 0.750$ from $M = 0.728$ in Figure 4.9. The obvious effect it has is increasing the strength of the shock, with more pronounced "rooftops" at lower angles of attack. One of many instances of this can be viewed by comparing Figure 4.9b to Figure 4.10b. Additionally, the separation bubble burst also happens sooner which shifts the shock location upstream, as seen by comparing Figure 4.9e to Figure 4.10e.

It is crucial to understand how the CFD results vary from the pETW results and identify the causes for these differences. The red lines in Figure 4.9, Figure 4.10 and Figure 4.11 represent the CFD results at the exact flow conditions of the pETW results after applying wind tunnel corrections. The major differences are seen either at the shock location or in the flow downstream of the shock. It can be viewed that the CFD results (for the SST turbulence model) are in good agreement for a specific strength of the shock, i.e. the combination of M and AoA. In other words, for a given M, for instance $M = 0.750$ in Figure 4.10, the CFD results agree well only for a given range of AoA, where the flow situation has matured. These are $AoA = 2.236$ in Figure 4.10c and $AoA = 3.222$ in Figure 4.10d. Above these angles, the shock location is clearly misidentified and post-shock pressure is also overestimated (Figure 4.10e-Figure 4.10f). This behaviour is expected due to the strong shocks and shock-induced separation extending until the trailing edge, and these are not easily estimated by existing CFD solvers.

However, below $AoA = 2.236$, the CFD agreement is also poor. In Figure 4.10b, it can be seen that the C_p value for CFD is underestimated on the upper surface, and a significant difference can be seen near the shock location. The author hypothesizes that this is attributed to the sensitivity of CFD results to the boundary layer thickness at low angles of attack. Incorrect estimation of the displacement thickness of the boundary layer means the CFD algorithm sees the adverse pressure gradient differently from the actual flow, and thus the shock location is not identified well.

The author believes that the primary source of differences between the CFD and pETW data is the modelling error, originating from the assumptions of the turbulence model (k-omega SST) used for the CFD runs. Other models may perform better on a similar flow condition. The turbulence model proposed for this project is the SA-neg model and is expected to perform better than the k-omega SST. It would have been interesting to compare the performance of multiple turbulence models for shock-induced separation flows, but it has not been done due to time constraints. NASA provides a comparison of turbulence models for the RAE 2822 airfoil for two-dimensional, turbulent, transonic flows in the RAE 2822 Transonic Airfoil Study #5 at the NPARC Alliance Validation Archive.

Another important source of difference is the fully turbulent assumption for the CFD runs. The pETW data sees a transition from laminar to turbulent flow at some point of the airfoil (fixed or free), usually close to the leading edge. However, the CFD data is fully turbulent throughout the airfoil surface. This disagreement introduces an unquantifiable uncertainty when comparing CFD and pETW data. Finally, the CFD runs are done using a 2D CFD isolated airfoil section; however, the pETW results are measured in a closed wind tunnel section with slotted/solid walls. This introduces a discrepancy between the two results, whereas a 2D CFD run with top and bottom walls would be closer to the pETW wind tunnel measurement.

For separated flows at high angles of attack, there is a possibility of 3D wall effects being more pronounced. This has implications for the flow measurements at the Main section of the pETW. Separated flow at the airfoil means that the flow is also more susceptible to be separated at the side walls. This separated flow has a displacement effect on the centerline flow, making it even faster as the streamlines

converge towards the centre. Thus, pETW results at high angles of attack may differ from the CFD results where no such phenomenon is expected. Garbaruk et al. verified these side-wall effects for the AGARD case 10, where shock-induced separation is expected.

Finally, minor differences can also originate due to errors in wind tunnel sensors recording measurements. The author and the database supplier observed that the agreement between the CFD and pETW data is generally good for most of the data points available; however, the CFD model still lacks in predicting some flow cases. These cases may include phenomena like a shock-induced boundary-layer separation or a strong shock case for which the CFD model is not prepared.

4.4 CONCLUSIONS FROM DATABASE ANALYSIS

The main purpose of the analysis of the experimental reference database is to understand the scope and applicability of the data for the FIML approach. The following conclusions are reached after the database analysis stage:

- The corrections derived for the slotted wall configuration are crucial to validating the test data. For a slotted test section, the results obtained are better due to fewer wind tunnel corrections that need to be applied. Slotted wall data correlates better with the 2D Free air CFD results provided to the author.
- For most of the points in the database, the agreement between the CFD derived data, and pETW data is reasonably good. The primary reason for the differences between the CFD and pETW data are assumed to be the turbulence model errors. Other reasons like transition, side-wall and 3D effects in the wind tunnel section may contribute, but will be ignored for the purpose of this study.
- There is a fairly poor match between the original AGARD experiment data for the RAE2822 aerofoil section and the equivalent pETW test. However, the agreement between CFD derived data is better for the pETW data than for the original AGARD data.
- 2D Free air CFD results supplied have an assumption of a fully turbulent flow. However, the flow in the wind tunnel results undergoes transition, albeit at an early x/c of the airfoil. Therefore, there is an unquantifiable uncertainty when comparing the two cases, and it is a consideration to be kept in mind.

4.5 FIML CASES SELECTION, RECOMMENDATIONS

After understanding the scope of the database, the next activity is the selection of training and test cases for the field inversion process, which will be ultimately fed to train the machine learning model. The main purpose of the FIML approach in this project is to improve the existing turbulence models, so the data points to be selected should have a big deviation between the pETW data and CFD data to provide that scope for improvement. A good example for this case can be seen in Figure 4.7.

When comparing the CFD data with the pETW data, only the CFD run at exact same flow condition (red-line CFD data from Figure 4.7) as of the corrected pETW result is used rather than the paired CFD line as explained in Section 4.2.1. This choice provides a uniform approach over using the paired CFD data for different wind tunnel runs, as no consistent paired value (from either Blanking-ON or Blanking-OFF) has a better similarity factor value for all wind-tunnel runs.

The improvement in the baseline turbulence models is to be done by alleviating only the modelling errors from the turbulence model in the CFD simulations. Therefore, flow cases in pETW data which are closest in setup to a 2D CFD free air isolated simulation should be chosen. This means that slotted wall data will be preferred over the solid wall data from the pETW database.

From the discussion in the previous sections, the following action plan is devised for the selection of the training and test cases for the FIML approach:

1. There should be a significant deviation between the pETW and CFD data at the same flow conditions. This is primarily observed at high-angles of attack where shock-induced separation extends till the trailing edge.
2. The data from only the slotted wall configuration campaign is considered. The slotted wall configuration requires less wind tunnel calibration and is arguably closer to 2D free air CFD results.
3. The training and test cases are spread over the dataset range, i.e. over the full range of M, Re and AoA values seen in Figure 4.2.
4. Only free transition points should be chosen, as their flow characteristics are natural and not interrupted by forced transitions. The author does not fully understand the effect of fixing the transition, thus it is better to stay with free transition points.

5 FIELD INVERSION

This chapter is about the Field Inversion (FI) part of the FIML procedure. The author is provided with an experimentally obtained high-fidelity C_p database for the RAE2822 airfoil, which was discussed in Chapter 4. This database is used as the reference data for setting up the field inversion problem, introduced in Chapter 3, which will be used to extract the model augmentation field later used for machine learning training. This chapter details the theoretical formulation of the inverse problem for the field inversion procedure. The theory is followed by the author's implementation of the field inversion in the TAU code, first implemented by Jäckel [2020]. The methodology to choose the hyper-parameters for the inversion problem is discussed, followed by the results obtained. Finally, the recommendations to extend the current field inversion problem and improve the existing results are provided.

5.1 FORMULATION OF THE INVERSE PROBLEM

A physical system in the natural world can be described using a mathematical model based on governing equations. A set of input parameters is chosen, and the equations are solved based on some boundary conditions: this is a typical forward problem description. Consider the following system of equations for a forward problem, where $\mathcal{X}$ includes the input parameters and boundary conditions:

$$\mathbf{G}(\mathcal{X}) = \mathcal{Y} \tag{5.1}$$

where $\mathcal{Y}$ is the predicted output data of the model. In the inverse problem, the modeller provides these observations, and the input parameters are found. In other words, an optimal mapping $\mathbf{F}$ is sought, which finds the cause (input parameters or $\mathcal{X}$) to the observed effect ($\mathcal{Y}$). This mapping can be represented in the following manner:

$$\mathcal{X} = \mathbf{F}(\mathcal{Y}) \tag{5.2}$$

The fundamental issue of the inversion problem is that the mapping $\mathbf{F}$ is generally not well-defined, and it may not even exist. Discovering the mathematical model and its parameters requires exploring the huge parameter space, which further makes it more difficult to find this mapping [Tarantola, 2021]. Additionally, it is not necessary that the model parameters have a one-to-one relationship with the measured data. The inverse problem is described in the Figure 5.1.

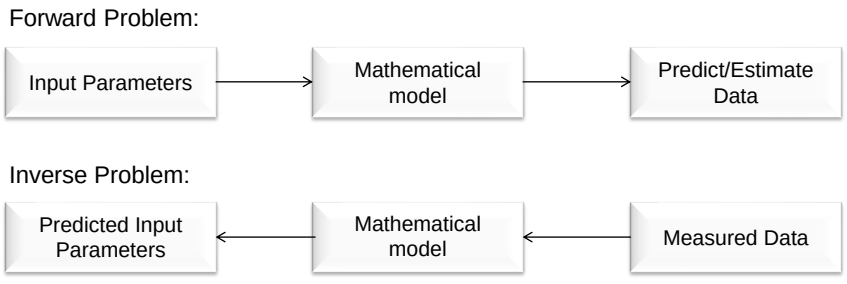


Figure 5.1: Formulation of a forward problem vs an inverse problem

The author is provided with a high-fidelity C_p database for the RAE2822 airfoil, which will serve as the observed data to solve the inverse problem. A baseline turbulence model (SA-neg model in this case) with a correction term (say β) will act as the mathematical model, as in Figure 5.1. The inverse problem aims to infer the correction in the turbulence model from the high-fidelity data. This correction term is, in fact, a measure of the discrepancy between the baseline model and the observed data. The baseline model is iteratively optimized to an augmented turbulence model where the correction term inferred gives close predictions to the ground truth.

The proposed implementation of the FIML approach in this project was briefly explained in Section 3.3, which is repeated here for convenience with more focus on the FI part. The system of equations for an augmented RANS (SA-neg) model, as previously presented in Equation 3.11 is given by:

$$\mathcal{R}_a(\widetilde{\mathbf{U}_m}, \tilde{\nu}, \beta) = 0,$$

where the $\tilde{\nu}$ is the working variable ($\tilde{\nu}$) of the negative Spalart-Allmaras (SA-neg) turbulence model. The discrepancy β is introduced as a functional correction term multiplied by the production term in the transport equation for the working variable $\tilde{\nu}$. The augmented transport equation of the $\tilde{\nu}$ of the SA-neg model is then given by (as of Equation 3.12):

$$\frac{D\tilde{\nu}}{Dt} = \beta(\widetilde{\mathbf{U}_m}, \tilde{\nu})P(\widetilde{\mathbf{U}_m}, \tilde{\nu}) - D(\widetilde{\mathbf{U}_m}, \tilde{\nu}) + T(\widetilde{\mathbf{U}_m}, \tilde{\nu}),$$

where P, D and T are the production, destruction and transport terms, respectively. In the inverse problem for this system an optimal mapping for the discrepancy β_{opt} is sought which maps the discrepancy term to the high-fidelity data supplied. To obtain β_{opt}, the inverse problem is posed as an optimization problem. The optimization problem in this case is given by:

$$\beta_{opt} = \underset{\beta}{\operatorname{argmin}}\ \mathcal{L}, \qquad \text{s.t} \qquad \mathcal{R}_a(\widetilde{\mathbf{U}_m}, \tilde{\nu}, \beta) = 0, \tag{5.3}$$

where $\mathcal{L}$ is the loss function driving the optimization problem. In the "full-field inverse problem", a spatial distribution for the β_{opt} (referred hereafter as β field or discrepancy field) is to be obtained. On the discretization of the model equations ($\mathcal{R}_a$), the β is also discretized to a finite number of grid points. The value of β at each grid point is unknown, and the typically large number of grid points adds to the complexity of the inverse problem.

One may argue that this inversion procedure is not needed, and a β field can be calculated directly by substituting reference data values ($\mathbf{U_{ref}}$) in the Equation 3.12 and Equation 3.11. However, it is to be noted the terms in these equations are not physical terms and are empirically calibrated model source terms ($\widetilde{\mathbf{U}_m}$). Therefore, direct calculation of $\beta(\mathbf{U_{ref}})$ may not result in the correction required for the ideal augmented solution, and there is a case the solution might deteriorate further [Singh, 2018].

5.1.1 Deterministic Field Inversion Problem

As stated previously, to obtain β_{opt}, the inverse problem is posed as an optimization problem. The inverse optimization problem can be formulated in two different ways. The first approach is the Bayesian approach, which allows for the extensive treatment of uncertainties that accompany inverse problems. This approach allows to implementation of prior information about the data into the model using a user-defined prior distribution, which has its advantages [van Korlaar, 2019]. However,

the selection of this prior distribution has a big impact on the output or the posterior distribution, and the approach to select the prior is subjective [Singh, 2018]. Therefore, this study will use the deterministic inversion approach, where a loss function drives the optimization problem. This loss function represents the measure of discrepancy between the model output and input data to be minimized. This approach loses out on the uncertainty analysis aspect but is easier and less computationally expensive to implement. The loss function driving the deterministic inverse problem as first implemented in the TAU solver available at DLR by Jäckel [2020] is defined as follows:

$$\mathcal{L} = \underbrace{\sum_{j}^{N_j} \left[C_{p_{ref}}^{j} - C_{p_{RANS}}^{j}(\beta, \mathbf{U}, \tilde{\nu}) \right]^2}_{\mathcal{L}_1} + \lambda \underbrace{\sum_{j}^{N_j} \left(\beta^j - 1 \right)^2}_{\mathcal{L}_2} \tag{5.4}$$

The optimization problem in this case is given by:

$$\beta_{opt} = \underset{\beta}{\operatorname{argmin}}\ \mathcal{L}, \quad \text{s.t} \quad \mathcal{R}_a(\widetilde{\mathbf{U}_m}, \tilde{\nu}, \beta) = 0, \tag{5.5}$$

The first term in Equation 5.4 ($\mathcal{L}_1$) [1] is the mean square error for the coefficient of pressure (C_p) value between the calculation from (augmented) RANS solution and the reference high fidelity data, summed over all the N_j cells in the grid. This term represents the deviation of the current solution from the reference data for the selected property (C_p). Other properties like C_f, C_l or velocities u could have been used as a substitute to C_p.

The second term in Equation 5.4 ($\mathcal{L}_2$) is the regularization term required to penalize the solution due to the ill-posed nature of the problem. The solution is penalized to $\beta = 1$, corresponding to the case when the production term is not augmented in Equation 3.12. The regularization term is needed because of three reasons:

1. Non-unique solution: The solution is non-unique as the problem is underdetermined because of a high number of unknowns as β is to be found at each grid point.
2. Noisy reference data: The experimental reference data has been procured through pressure sensors, and the spatial resolution of the sensor information is lower than the grid resolution to be used for the RANS calculation. Therefore, the data is interpolated from the sensor points to the CFD grid points on the airfoil surface. Numerical noise is introduced in this interpolation process, and the uncertainities involved with the experiment.
3. Non-linear forward problem: The forward problem or the SA-neg turbulence model is a RANS method, which is inherently non-linear.

This regularization is introduced in the form of *Tikhonov regularization* [Tikhonov and Arsenin, 1977], where λ or the Tikhonov parameter is a parameter of the inversion problem which needs to be determined by engineering judgement. A detailed discussion of how the hyperparameters for the field inversion problem were chosen in this implementation will be done in Section 5.3.

5.2 FIELD INVERSION IMPLEMENTATION

The field inversion was implemented using a code based on the python-based DLR Surrogate-Modelling for Aero-Data Toolbox (DLR SMARTy) [Görtz et al., 2013] developed by the German Aerospace Center, which wraps around the adjoint and normal flow simulation capabilities of the DLR's inhouse flow solver TAU [Schwamborn

1 Please note that here the terms $\mathcal{L}_1$ or $\mathcal{L}_2$ do not correspond to the L_1, L_2 loss associated to loss functions.

et al., 2006]. The ability of the SMARTy toolbox to call the TAU solver for a flow solution while simultaneously implementing various optimization techniques is achieved using the FlowSimulator framework [Reimer, 2015]. This system of software is leveraged in the current implementation, with a similar instance first carried out by Jäckel [2020]. The SMARTy toolbox treats a given flow solution file as "snapshots" or "snaps", and these terms will be used to refer to a solution file containing surface or full-field flow information. The flow cases for the field inversion procedure is selected using the criteria detailed Section 4.5, and the selected cases are listed in Table 5.1.

Reynolds number	Mach number	Angle of attack (degrees)	Static Temperature (K)
2,680,960	0.71734	2.6038	268.9717
6,360,970	0.74208	4.4563	103.7620
8,787,960	0.72089	5.6690	104.4444
10,939,600	0.72401	5.6541	104.4660
13,181,400	0.72418	5.6504	104.4948
15,323,800	0.72355	5.1450	104.5628

Table 5.1: Flow points in the pETW database selected for the Field Inversion procedure

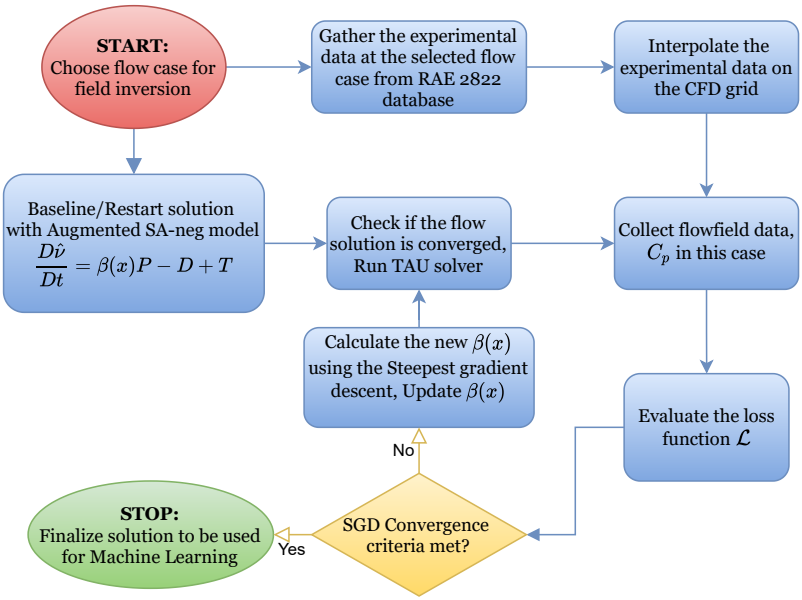


Figure 5.2: Flowchart describing the current Field Inversion implementation

For a selected inversion flow case in Table 5.1, the code starts with a converged solution file ran with the baseline SA-neg turbulence model, containing the flow data at all CFD grid points. The experimental data supplied by the RWC.01 RAE2822 database, containing the surface pressure distribution, is the reference data. A reference snap containing the experimental data at the same flow case is also needed. The reference snap contains the surface pressure coefficient (C_p) distribution information, also available in the baseline snap. This information is passed to the $\mathcal{L}_1$ term to evaluate the residual part of the loss function of the optimization, as in Equation 5.4. A $\beta = 1$ field is initialized for the baseline snap, whereas, for a restart snap, an existing $\beta(x)$ is initialized. The spatial field $\beta(x)$ is passed in the $\mathcal{L}_2$ term of the loss function.

The information about the loss function is passed to the optimizer. For the current implementation, an iterative gradient-based method Steepest Gradient Descent

(SGD) algorithm, as explained in Section 2.3.2, is used. The optimizer updates the $\beta(x)$ using gradients from a discrete adjoint implementation to reduce the loss function. If the convergence criteria of the SGD optimizer is met, the updated $\beta(x)$ along with the flow variables related to the solution are selected to be used for the Machine Learning part of the FIML procedure. The field inversion procedure is shown in Figure 5.2. Further details about the implementation of various components of the field inversion follow in the upcoming sections.

5.2.1 Baseline simulation setup using SA-neg

Before running the field inversion, the baseline simulation was conducted solely using the DLR TAU code, a highly optimized, parallel, state of the art CFD solver for unstructured grids. The grids required for the simulation were provided by AIRBUS and are the same as the ones used for CFD analysis explained in Section 4.2. Six separate structured 2D grids of varied refinements, each containing 157116 points, were provided for different Reynolds number cases; with one each for Re = 2,6,9,11,13,15 million cases. An example of the grid can be seen in Figure 5.3. The negative Spalart-Allamaras turbulence model (SA-neg) was used to run the flow simulations, the default 1-equation eddy-viscosity turbulence model in the TAU solver. This model has a highly increased numerical stability and is robust. Additionally, it is not particularly sensitive to separation. The density residual value of 10^{-10} was set as the convergence criteria for stopping the simulations.

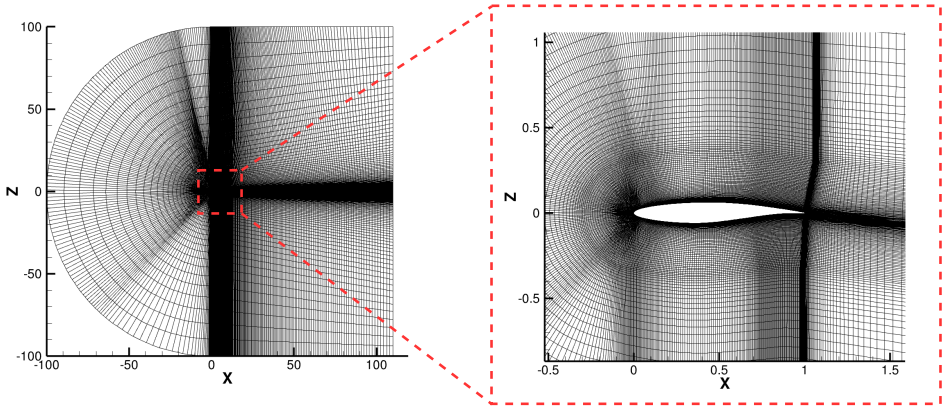


Figure 5.3: Computational grid for the 2D domain for CFD simulations in TAU solver.

The boundary condition on the airfoil is that of a wall with a turbulent flow assumption. A no-slip boundary condition has been applied and the grid is fine enough near the airfoil ($y^+ < 5$) to not require any wall functions. The inflow has been set as a far-field boundary condition, and all the flow conditions (M, Re, AoA) are initialized here. All gradients are assumed to be zero on the far-field boundary, and therefore no viscous effects are taken into account. The simulation is converted to 2D by applying the symmetry plane condition on the sidewalls. A visual representation of the boundary conditions can be viewed in Figure 5.4.

The convective part of the inviscid fluxes of the RANS equations are discretized using a 2nd order central scheme. The flux evaluation has an artificial dissipation term, which is set using a matrix dissipation scheme. This dissipation scheme scales with all three eigenvalues of the flux Jacobian, making it less dissipative than the scalar dissipation scheme, which only scales with the maximum eigenvalue of the flux Jacobian. Less dissipation is helpful for the current flow case as we expect to see shock waves at some point on the airfoil.

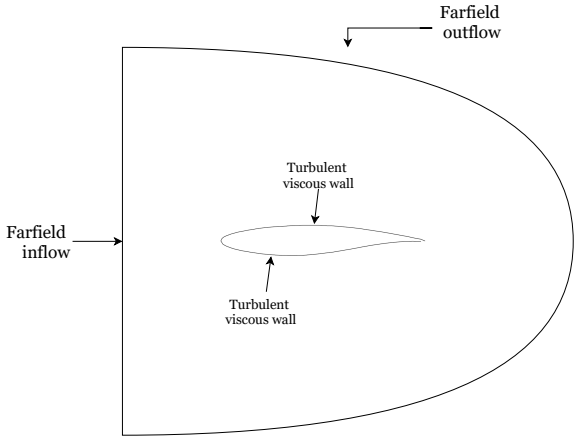


Figure 5.4: Boundary conditions for the 2D domain for CFD simulations in TAU solver. The domain in the figure is not to scale, and the airfoil section has been magnified for representation purposes.

For time-stepping, a semi-implicit LU-SGS scheme is used with a Backward Euler relaxation solver. Even though implicit schemes allow for a bigger time step, it was noticed that the Courant-Friedrichs-Levy (CFL) number value could not be pushed over 5.0 without compromising numerical stability for the current flow case, owing to the transonic flow conditions. To make the CFD run more efficient, defining the multi-grid cycle helped to accelerate the process. Typically, the calculation was started with a single grid configuration, and upon reaching a certain level of convergence of the density residual (typically 1e-6, determined from experience), the multi-grid cycle was switched to a 3-level V cycle.

5.2.2 Field inversion code implementation

Starting the code

To start the code, the following are required as an user input:

- Reference data from the RAE 2822 database
- Baseline/Restart flow solution
- Convergence criteria for TAU flow runs (typically density residual or maximum number of flow iterations) inside the inversion code.
- Stopping criteria for the adjoint solver (typically maximum number of iterations)
- Hyperparameters of the SGD optimizer
- Tikhonov regularization values

The pressure distribution data from a relevant RAE-2822 database point is procured as the reference data for a given set of flow conditions. The sensor information from the pETW database point has a coarser spatial resolution than the CFD grid, therefore the data is interpolated to the CFD grid using a linear interpolation method. A converged baseline run at the same flow conditions using the procedure in Section 5.2.1 is used as a baseline RANS flow solution for the code. In case a restart solution is available, which may be a partially converged $\beta(x)$ solution or a user-defined $\beta(x)$ field, TAU solver is re-ran using the convergence criteria (typically density residual or maximum number of flow iterations) provided in the user input. This selection process can be seen in Figure 5.5. The reference data, baseline RANS data and the $\beta(x)$ field are then sent for loss function evaluation.

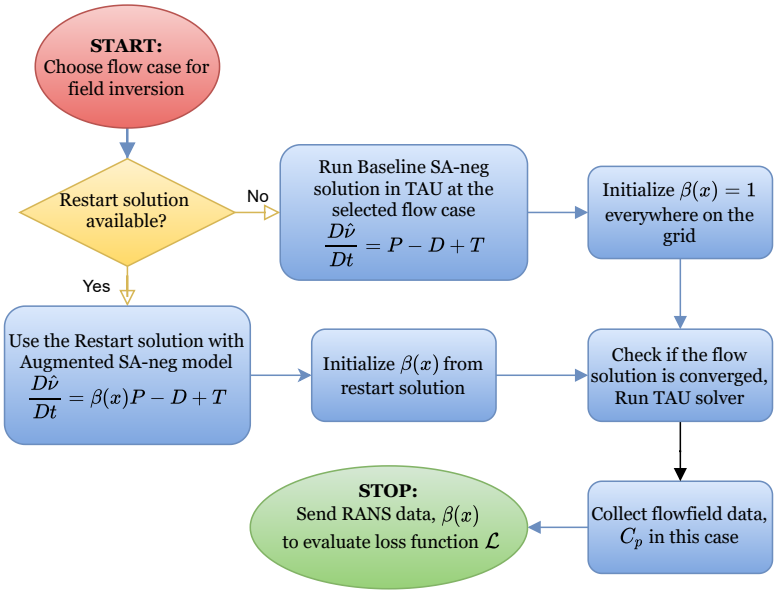


Figure 5.5: Selecting the CFD flow solution for the start of the Field Inversion procedure

Loss function and Steepest Gradient Descent update

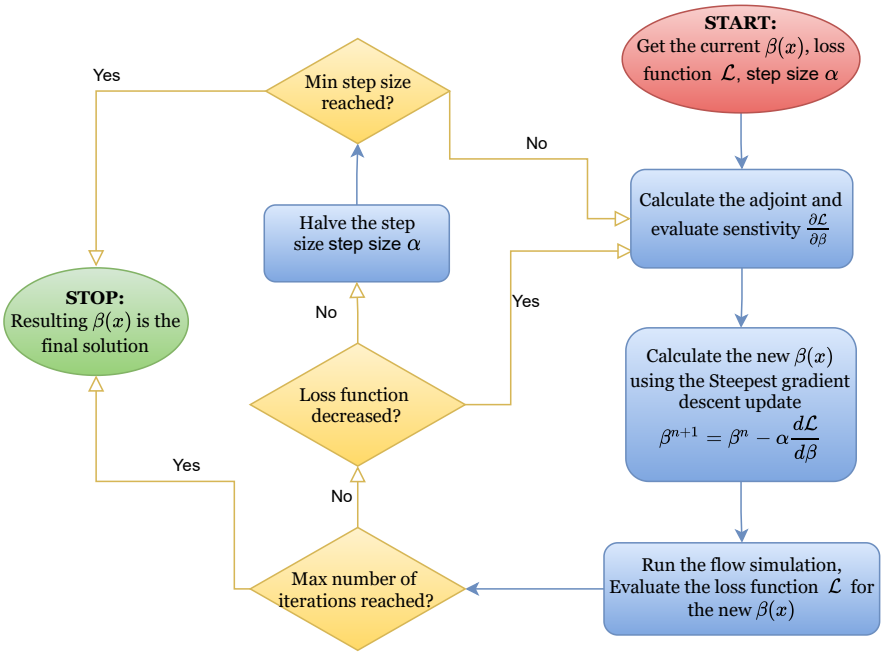


Figure 5.6: Adjoint calculation and Steepest Gradient Descent update in the inversion code. This is the step where the discrepancy field is updated in the code, and the cycle goes on until convergence criteria are met

Using the loss function information, the adjoint calculation is run, which calculates the gradient of the loss function with respect to $\beta(x)$; therefore, the gradient of the loss function is available at each grid point. The gradients $\frac{d\mathcal{L}}{d\beta}$ are calculated using the discrete adjoint method [Giles and Pierce, 2000] in TAU flow solver [Dwight and Brezillon, 2006]. This gradient information is used to update the $\beta(x)$ using a Steepest Gradient Descent (SGD) update step given by

$$\beta^{n+1} = \beta^n - \alpha \frac{d\mathcal{L}}{d\beta}, \tag{5.6}$$

where α is the step size of the update. After this update, the flow solution is recalculated using TAU until the flow solution converges according to user-defined convergence criteria. The loss function is then recalculated again using this updated flow snap with new $\beta(x)$, and the convergence criteria of the SGD is checked. The convergence criteria is a minimum step size of 1e-4, with the step size reducing by half when the problem loss function stops reducing. The starting step size and the maximum number of SGD iterations are a part of the user input. This process, also explained in Figure 5.6, repeats until the convergence criteria of the SGD optimizer is met.

Discrete Adjoint theory and implementation in TAU

The implementation of discrete adjoint method in TAU follows from the implementation by Dwight and Brezillon [2006]. The theory behind the discrete adjoint method has been explained in Section 2.3.3. From Equation 5.5,

$$\beta_{opt} = \underset{\beta}{\operatorname{argmin}}\ \mathcal{L}, \quad \text{s.t} \quad \mathcal{R}_a(\widetilde{\mathbf{U}_m}, \tilde{\nu}, \beta) = 0,$$

An augmented response can be generate from this constrained optimization problem, which can be used to find the β_{opt}, or,

$$\beta_{opt} = \underset{\beta}{\operatorname{argmin}}(\mathcal{J}(\beta)), \quad \text{where} \quad \mathcal{J}(\beta) = \mathcal{L}(\beta) + \Lambda\mathcal{R}_a(\beta). \tag{5.7}$$

The required gradient of the augmented response w.r.t β can be written as

$$\frac{d\mathcal{J}}{d\beta} = \frac{\partial\mathcal{L}}{\partial\beta} + \Lambda\frac{\partial\mathcal{R}_a}{\partial\beta} + \left(\frac{\partial\mathcal{L}}{\partial U} + \Lambda\frac{\partial\mathcal{R}_a}{\partial U}\right)\frac{\partial U}{\partial\beta}. \tag{5.8}$$

Using the adjoint method and eliminating $\dfrac{\partial U}{\partial\beta}$, we can compute Λ by satisfying

$$\frac{\partial\mathcal{L}}{\partial U} + \Lambda\frac{\partial\mathcal{R}_a}{\partial U} = 0. \tag{5.9}$$

After computing Λ and with the terms $\dfrac{\partial\mathcal{L}}{\partial\beta}, \dfrac{\partial\mathcal{R}_a}{\partial\beta}$ known, we can compute the required gradient information

$$\frac{d\mathcal{J}}{d\beta} = \frac{\partial\mathcal{L}}{\partial\beta} + \Lambda\frac{\partial\mathcal{R}_a}{\partial\beta}. \tag{5.10}$$

The derivative of the residual of the RANS equations with respect to the flow variables, $\frac{\partial\mathcal{R}_a}{\partial U}$, is called the Jacobian of the discretization. Note that adjoint problem is a linear problem, and requires a solution of a system of linear equations. In the current implementation, the linear geometric multigrid solution approach accelerated with Generalized Minimal Residual algorithm (GMRes) solver [Saad and Schultz, 1986] is used to solve this system. The user provides the number of iterations for the GMRes solver during the start of the code.

5.3 TIKHONOV REGULARIZATION CHOICE

The regularization constant (λ) or the Tikhonov parameter in Equation 5.4 is a parameter of the inversion problem which needs to be determined by engineering judgement. The expectation is that on increasing the strength of the regularization, the $\beta(x)$ field will be forced towards the baseline solution of $\beta = 1$ and the $\mathcal{L}_2$ term will decrease. A highly regularized field is far away from the inversion solution,

which will lead to an increase in the $\mathcal{L}_1$ term. This behaviour can be seen in Figure 5.7, where field inversions were ran for multiple values of λ for a selected flow case from Table 5.1.

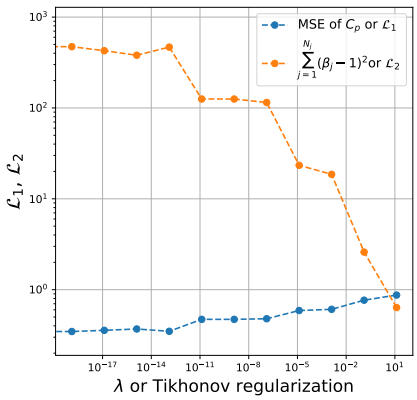


Figure 5.7: Variation of $\mathcal{L}_1$, $\mathcal{L}_2$ for various values of Tikhonov regularization (λ) at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960. The values of λ are relative to the initial value of the loss function ($\mathcal{L}$) at the start of the optimization, 7.8474e-09 in this case

Attempts were made to select its value according to the *L-curve criterion* [Hansen and O'Leary, 1993]. The L-curve is a parametric plot between the regularized solution ($\mathcal{L}_2$) and the corresponding residual of the loss function ($\mathcal{L}_1$). The aim is to find a value of λ (λ_{elbow}) where the value of $\mathcal{L}_1$ does not significantly decrease on reducing the λ further, but the $\mathcal{L}_2$ still varies as it represents the magnitude of model modification. This can be viewed in Figure 5.8.

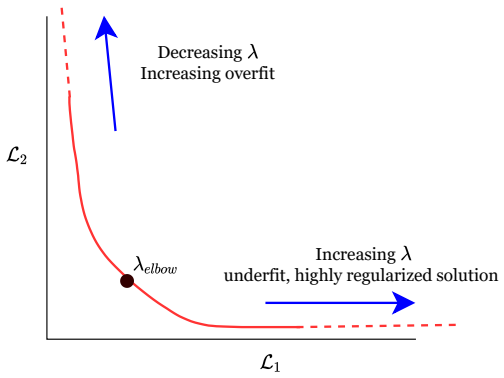
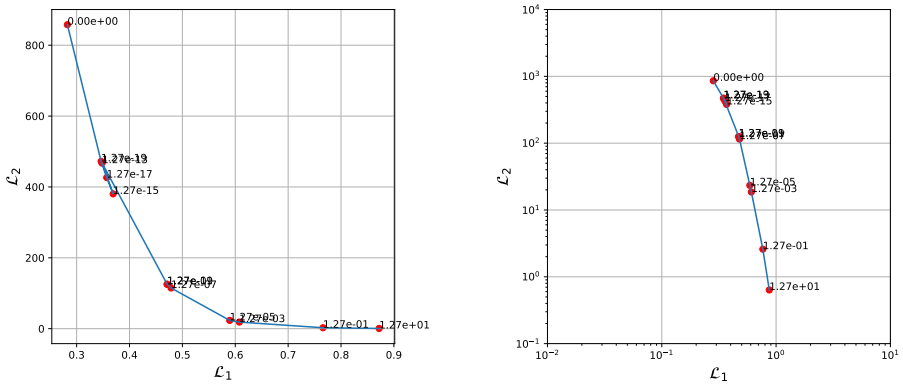


Figure 5.8: Example of a L-curve plotted using inversion solutions at different λ values. Elbow point is usually the point closest to the origin and the point with maximum curvature. Please note that this figure is only for illustration purposes.

Hansen and O'Leary [1993] further state that it is particularly advantageous to look at the L-curve in the log-log scale. A log-log scale emphasizes the elbow point in the L-curve, as it squeezes the points together where either $\mathcal{L}_1$ or $\mathcal{L}_2$ vary and shows a change in the second derivative of the curve near the elbow point. However, the primary advantage of a log-log transformation is to locate the elbow point correctly for noisy data, which is something the author expects in this implementation due to interpolation of reference data to the CFD grid (explained previously in Section 5.1.1) and the uncertainties in the experimental data. For a pure signal, the L-curve in the log-log scale is flat, and it gets steeper as the noise is added to

the signal. This distinction between noisy and pure signal is not seen in a lin-lin L-curve.

Figure 5.9 shows the L-curves in the two different scales for field inversion conducted at M = 0.7209, AoA = 5.669, Re = 8787960 for various values of Tikhonov regularization. Figure 5.9a shows the expected behaviour of the L-curve, but due to the noisy nature of the data it is not seen in Figure 5.9b. Therefore the value of λ chosen using Figure 5.9a may not have been optimal and the picture painted by Figure 5.9b is more trustworthy. However, a lack of a clear elbow point in Figure 5.9b forces us to look elsewhere in terms of choosing the lambda value, and this method is not feasible for the current application. Even after trying extremely low regularization values, there was no point in the graph where the change in the second derivative is observed. This behaviour of the L-curve was observed for all the selected flow cases in Table 5.1 while searching for an elbow point running multiple inversions.



(a) lin-lin plot **(b)** log-log plot

Figure 5.9: L-curve in (a) lin-lin scale, and (b) log-log scale for various values of Tikhonov regularization (λ) at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960. The values of λ are relative to the initial value of the loss function ($\mathcal{L}$) at the start of the optimization, 7.8474e-09 in this case

The inability to locate the elbow point indicates to the author that the optimization may not be fully converged. As we are dealing with a highly underdetermined problem due to many variables (i.e. β at every grid point) and , there is a big chance of getting stuck in saddle points while using a gradient-based method. Using a gradient-free method like Ensemble Kalman filter (EnKF) (used for FIML by [Yang and Xiao, 2020]) was not an option due to the time required for their implementation in TAU. Other options could be using higher-order gradient-based methods like BFGS or Quasi-Newton methods to improve accuracy. However, there is an argument that these methods will also get stuck in the same saddle points because the restart point of the inversion problem remains the same, i.e. the baseline β field of $\beta = 1$.

One strategy that was tried in the hope to escape the saddle points was to start the field inversion from a different restart point. For this purpose, a randomized β field was initialized in a close domain near the airfoil. This can be viewed on the left in Figure 5.10. The inversion field after this restart solution can be seen on the right of Figure 5.10. The output has a lot of noise in the solution, but physical phenomena related to shock-induced separation can still be viewed. The optimization is clearly not converged as even the noise in the optimization output looks very similar to the initial guess (restart solution).

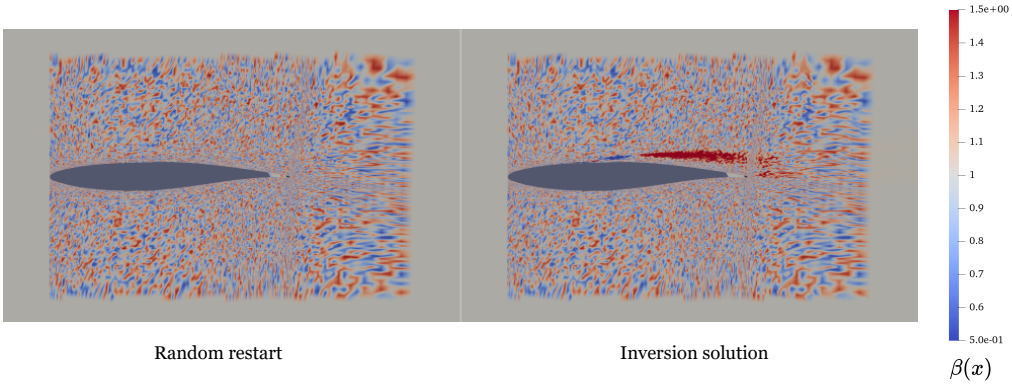


Figure 5.10: Random restart β field and its inversion solution

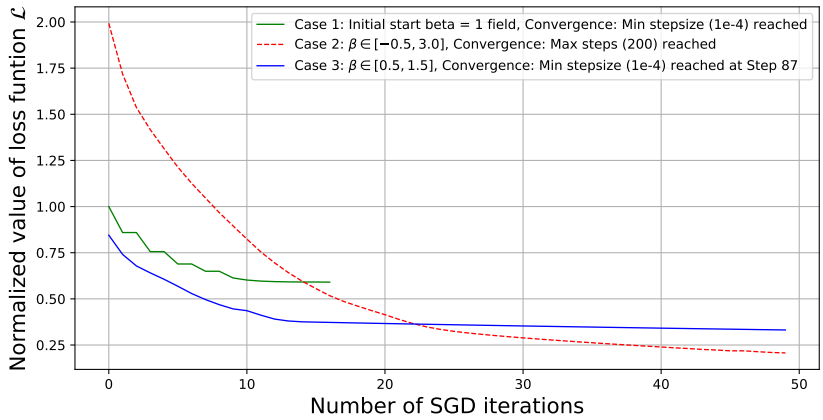


Figure 5.11: Convergence comparison for various restarts for inversion for $\lambda = 10^{-13}$ at flow conditions $M = 0.7209$, $AoA = 5.669$, $Re = 8787960$. The values of loss function are relative to the initial value of the loss function ($\mathcal{L}$) at the start of the optimization for the baseline run, 7.8474e-09 in this case

The comparison of convergence of the inversion procedure with restart solution as random β fields and starting from the baseline $\beta = 1$ can be viewed in Figure 5.11. The convergence criteria of the SGD optimizer for the cases seen in the plot is either the optimizer stops at a minimum step size of $1e-4$ or 200 SGD iterations are run. The cases seen on the plot are as follows:

- Case 1: Starting from a $\beta = 1$. Convergence with the minimum step size criteria is achieved at SGD iteration 17.
- Case 2: Initialize $\beta \in [-0.5, 3.0]$ in the domain $x/c \in [0, 1.5]$ and $z/c \in [-1.5, 1.5]$. Optimization stopped at max number of SGD iteration (200) is reached.
- Case 3: Initialize $\beta \in [0.5, 1.5]$ in $x/c \in [0, 1.5]$ and $z/c \in [-1.5, 1.5]$. Convergence with the minimum step size criteria is achieved at SGD iteration 87.

For Case 2 and Case 3, it can be seen that a lower value of the cost function of the optimization is reached. However, there are two reasons why this approach is not favourable: a) the output inversion field is extremely noisy, like the right picture Figure 5.10, and training ML algorithms on such noisy data is difficult, and b) the field inversion takes extremely long to solve as the number of SGD iterations required are higher, and it takes even longer to converge the flow simulation in TAU for each SGD iteration. The major takeaway from this activity was that the accuracy gain from the random restart cases is not worth the computational cost, the effort put into studying it. More clever ways of initializing restart β fields could

have been looked into, but this activity was dropped in the interest of the time of the master thesis duration.

5.3.1 Final approach for selecting the Tikhonov Regularization

With all the approaches discussed previously failing, the author was forced to select the λ value based on engineering judgement considering the physical flow phenomena. Figure 5.12 gives the β fields after inversion procedure for different values of λ. The physical phenomenon seen in β fields remains exactly the same for all λ values, with the production term being decreased near the shock foot and increased further downstream. the decrease in turbulence production leads to a decrease in effective viscosity, making the flow susceptible to separation earlier. Additionally, a region of production increase can be seen downstream of the shock location at the border of the boundary layer and this region is separated from the airfoil. This behaviour is consistent with the expected physical phenomenon of shock induced separation for these flow conditions.

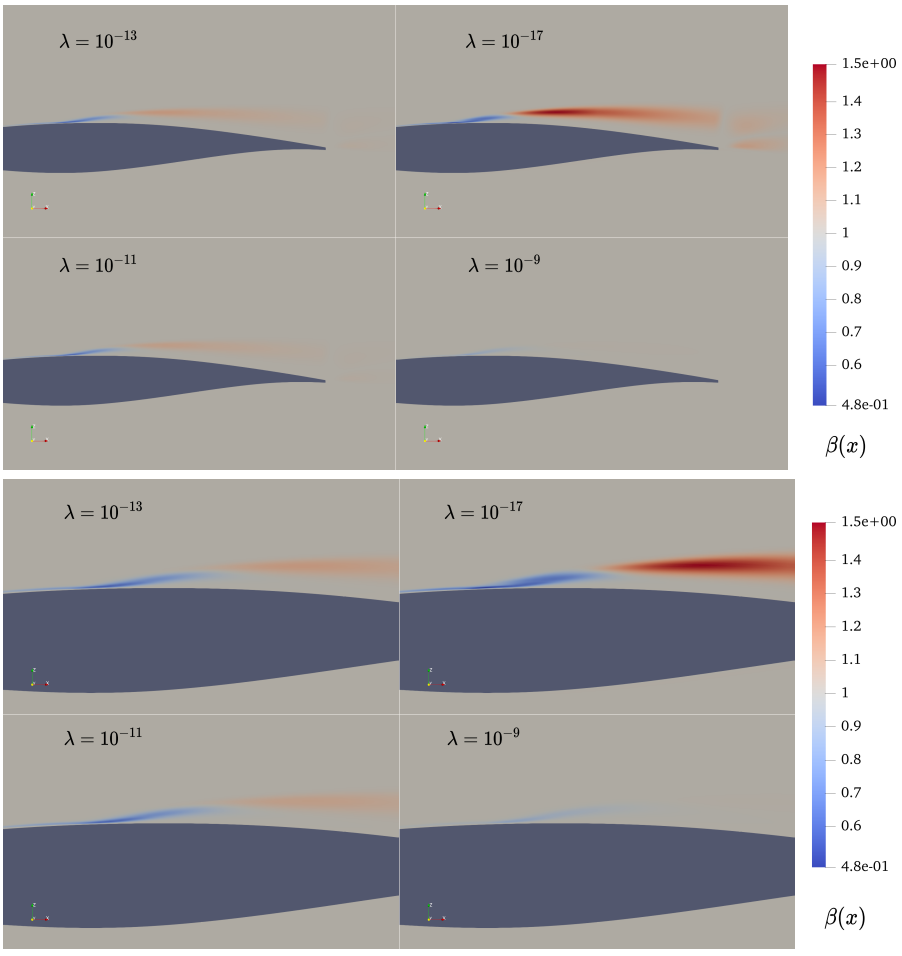


Figure 5.12: β fields in a a) far field view, and b) zoom in view near the shock foot view from inversion procedure for various values of Tikhonov regularization (λ) at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960. The values of λ displayed are absolute values. The initial value of the loss function ($\mathcal{L}$) at the start of the optimization was 7.8474e-09 in this case

A closer look of how the inversion fields look for varying strength of regularization and their effect on the pressure coefficient distribution of the airfoil provides more insight on choosing the regularization value. Weakly regularized inversion procedure go farther away from the baseline value of $\beta = 1$, resulting in larger range

of values for β. Comparing the pressure distribution, as seen in Figure 5.13, one can observe that for all $\lambda \leq 10^{-11}$, the shock location is identified well ($x/c \approx 0.3$) by the resulting β fields. The only difference between the $\lambda = 10^{-17}$ and other λ cases is that the resultant β field tries to fit closely to the behaviour in the region downstream of the shock ($x/c \approx 0.4 - 0.7$). Comparing the β fields in Figure 5.12 for $\lambda = 10^{-13}$ and $\lambda = 10^{-17}$, the major difference is the strength of production increase downstream of the shock. Minor differences can also be seen near the shock foot, but it does not have a big impact on the shock location identification in the C_p distribution. Thus the conclusion can be drawn that on decreasing the regularization, the resultant inversion field first fits the shock location and then the trailing flow.

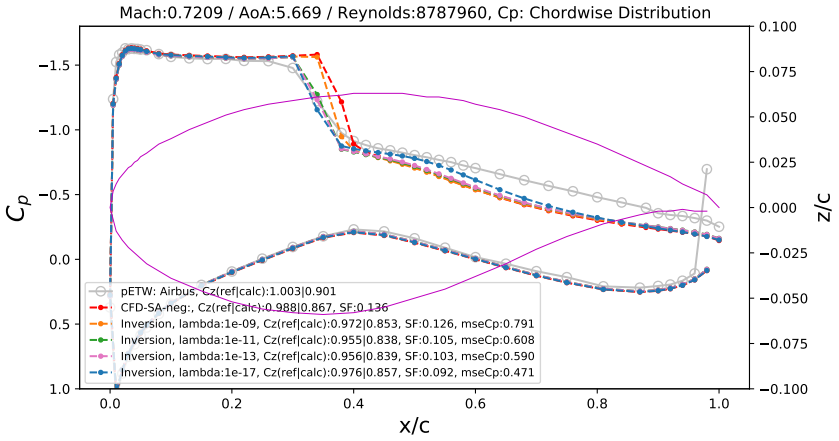


Figure 5.13: C_p distribution comparison for various inversion solutions at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960. 'ref'= reference output from TAU solver, 'calc' = calculated output from a user-defined function, 'SF' = similarity factor with respect to wind tunnel results, 'mseCP' = normalized loss function value for the inversion problem.

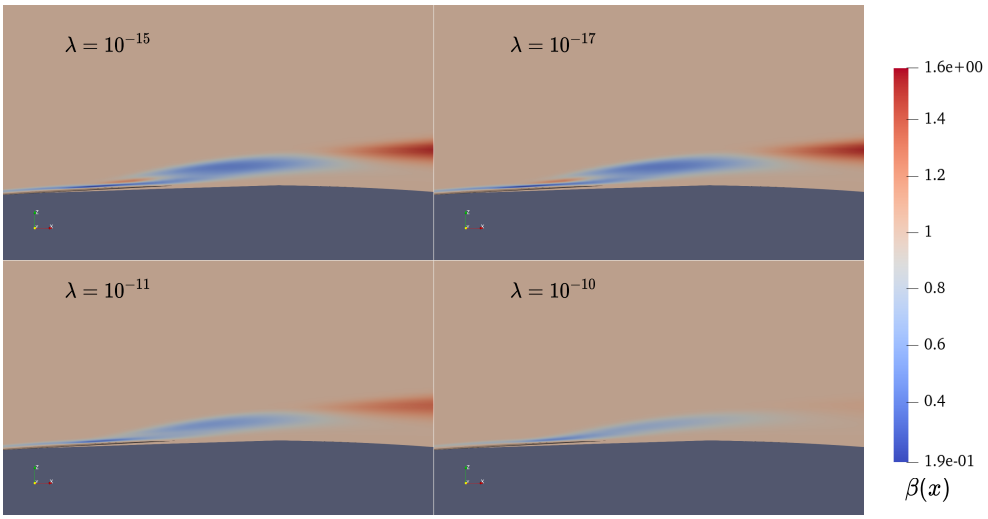


Figure 5.14: β fields in a zoom in view near the shock foot view from inversion procedure for various values of Tikhonov regularization (λ) at flow conditions M = 0.7242, AoA = 5.650, Re = 13181400. The values of λ displayed are absolute values. The initial value of the loss function ($\mathcal{L}$) at the start of the optimization was 7.9477e-09 in this case

Another behaviour that was observed in the inversion fields was the development of a production increase spot near the shock foot, which is predominantly a region

of production decrease for most inversion solutions seen until now. In Figure 5.14, a red spot can be observed within the blue region near the foot of the shock for low regularization values of $\lambda = 10^{-15}$ and $\lambda = 10^{-17}$. This spot is not observed for high regularization. The effect of this small production increase can be seen in C_p distribution for $\lambda = 10^{-17}$ in Figure 5.15. Near the shock location at $x/c \approx 0.3$ for $\lambda = 10^{-17}$, the pressure increase is not abrupt and is a bit curved.

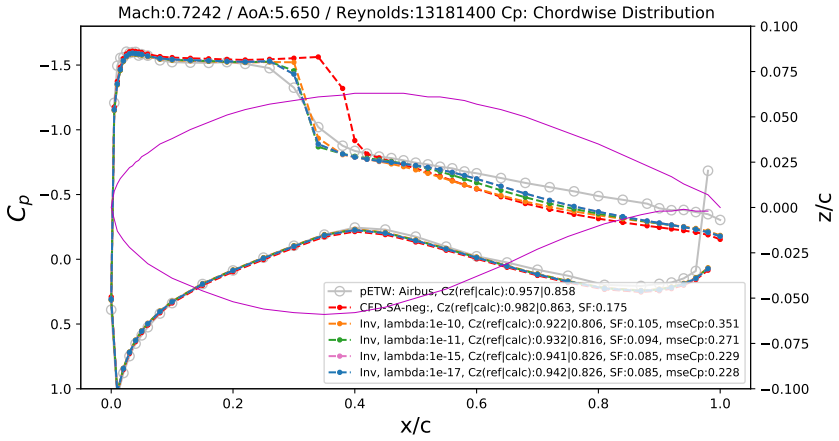


Figure 5.15: C_p distribution comparison for various inversion solutions at flow conditions M = 0.7242, AoA = 5.650, Re = 13181400. 'ref'= reference output from TAU solver, 'calc' = calculated output from a user-defined function, 'SF' = similarity factor with respect to wind tunnel results, 'mseCP' = normalized loss function value for the inversion problem.

Here, it can be said that for low regularization values λ, the inversion field adjusts so that it tries to fit even closer to the pETW database reference line, which has a smoother behaviour near the shock location. Thus, the inversion result matches closely to the smeared shock foot behaviour of the pETW data. On decreasing the regularization of the loss function, this phenomenon was observed sooner for high Re cases in Table 5.1.

Observing the inversion fields for all the flow cases in Table 5.1, it was decided to choose the λ value in such a manner that the resulting inversion field correctly identifies the shock location and the pressure increase must be sudden. Correctly identifying the flow phenomena in the region downstream of the shock was not given much importance due to two reasons: a) unsteady, turbulent, unpredictable behaviour, and b) unreliability of the sensors in the downstream region during the experiment, as stated in the experiment report by Airbus. Additionally, efforts were made not to capture the smeared shock phenomenon, which results in a smoother pressure increase at the shock location. The author believes this smeared shock behaviour in the wind tunnel data is either due to the unsteady behaviour of the shock in reality (which the author is not focusing on in this study) or due to inaccuracies in the setup and data collection methods of the experiment. This design choice corresponded to regularization fields of high strength, resulting in smoother inversion fields with small variance in the range of β. Flow data from these smooth inversion fields will be easy to learn for machine learning algorithms.

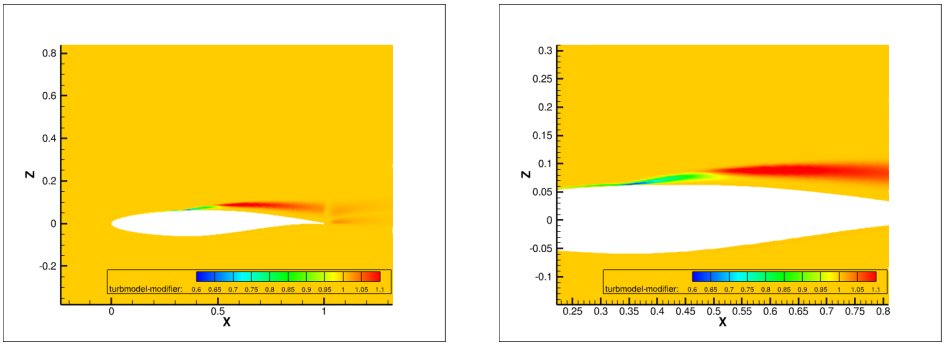
5.3.2 Selected values

From the selection approach discussed above, the inversion solutions were run for multiple values of λ determined by trial and error. The lowest value of λ where the shock location is identified correctly, and there is no effect on the flow downstream of the shock, is chosen. The chosen values can be seen in Table 5.2. One of the results from field inversion for the chosen λ values are seen in Figure 5.16

and Figure 5.17, rest can be found in Section B.1. The shock-induced separation phenomenon was not observed from the chosen cases for the $Re = 2$ million case; therefore, it was decided that this flow case will not be taken forward for further analysis and machine learning.

Reynolds number	Mach number	Angle of attack (degrees)	Chosen λ	Initial $\mathcal{L}$	Relative regularization
2,680,960	0.71734	2.6038	N/A	1.60E-08	N/A
6,360,970	0.74208	4.4563	2.00E-11	1.14E-08	1.75E-03
8,787,960	0.72089	5.6690	1.00E-11	8.41E-09	1.19E-03
10,939,600	0.72401	5.6541	1.00E-10	7.95E-09	1.26E-02
13,181,400	0.72418	5.6504	1.00E-11	4.42E-09	2.26E-03
15,323,800	0.72355	5.1450	1.00E-10	7.85E-09	1.27E-02

Table 5.2: Selected Tikhonov regularization values for the inversion flow cases. For Re = 2,680,960 case, no λ was chosen and data is not used for feature engineering and machine learning.



(a) Full view **(b)** Zoom at the shock foot

Figure 5.16: $\beta(x)$ field for the selected inversion solution (λ =1.00E-11) at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960.

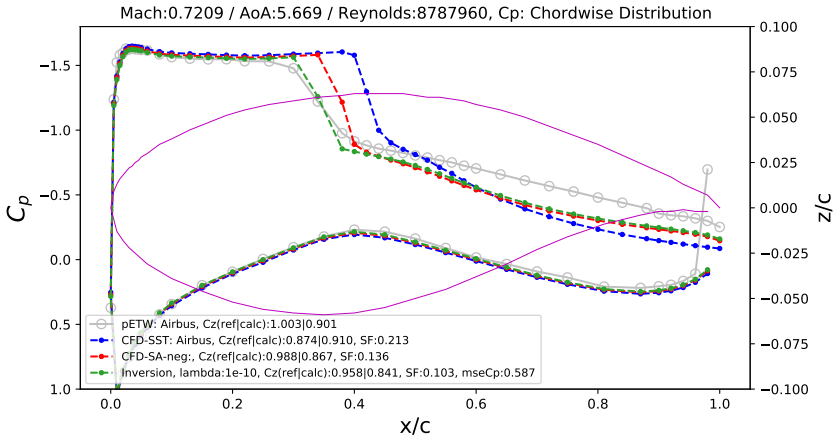
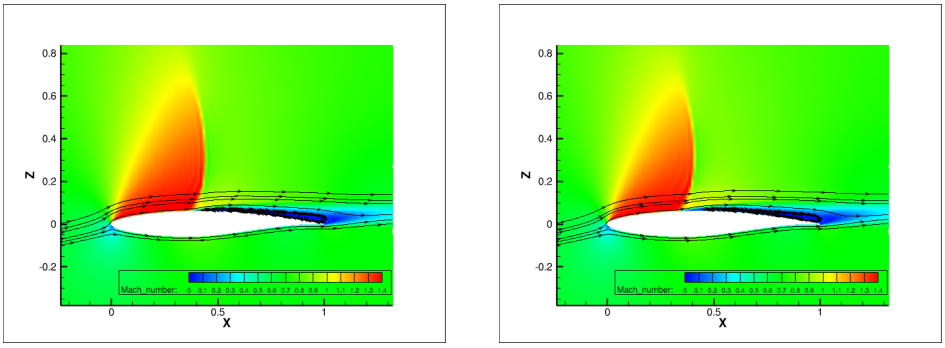


Figure 5.17: C_p distribution for the selected inversion solution (λ =1.00E-11) at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960. 'ref'= reference output from TAU solver, 'calc' = calculated output from a user-defined function, 'SF' = similarity factor with respect to wind tunnel results, 'mseCP' = normalized loss function value for the inversion problem.



(a) Baseline solution

(b) Inversion solution

Figure 5.18: Comparsion of the Mach number contours for the baseline vs. inversion solution at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960.

5.4 RECOMMENDATIONS

The underlying assumptions behind the design choices for formulating the current field inversion problem work for this project. However, if the project's aim is a generalized ML augmented turbulence model, there are ways to improve the current field inversion procedure. It is important to remember that the machine learning algorithm is only as good as its data, and better field inversion results will lead to a high-quality dataset for machine learning. In this section, a few recommendations are discussed that could improve the results but were not implemented due to constraints on the time available for the master's thesis.

It was mentioned in Section 5.2.1 that a fully turbulent assumption had been chosen as a boundary condition on the airfoil surface for the baseline runs using the SA-neg model. This leads to a similar problem as the one discussed previously in Section 4.3. The pETW data has a transition at some point close to the leading edge of the airfoil, and there is an unquantifiable uncertainty when comparing the baseline CFD and pETW data. However, unlike the CFD data provided by AIRBUS, this time, generating the CFD data is in the author's control, and efforts could have been made to include the transition phenomenon in the CFD data. The following recommendations are provided:

- Determine the transition location *a-priori* using transition models like the $\gamma - Re_{\theta t}$ Transport Equation Model available in TAU or by approximation after running multiple TAU simulations with fixed transition locations. After determining the location, the field inversion process can be run normally, with baseline simulation ran with the known transition location.

- Include the transition location as one of the design variables of the optimization ran during the inversion. While using gradient-based methods, the gradient of the loss function for transition location can be found using more straightforward methods like finite differences, and there is no need to involve adjoints.

- Use fixed transition data from the pETW database as reference data and use the exact transition location in the baseline CFD run. In the current implementation, the data for fixed transition is only available for Re = 6 million experiments (Figure 4.2), and it limits the potential of how the whole database can be used.

In Section 4.2.1 a pairing methodology of AIRBUS CFD data with pETW data was discussed based on a surrogate-based optimization with the QOI 'Similarity Factor'

(SF) (Equation 4.1). SF is a measure of discrepancy using the isentropic Mach number distribution between CFD data and the wind tunnel data. A similar QOI can be used to replace the isentropic Mach number with C_p to generate a surrogate surface using Re, AoA, M and transition location. The surrogate surface would help choose the appropriate (Re, AoA, M, transition location) combination to run the baseline SA-neg simulation instead of running it at the exact same condition as that of the pETW run. This will potentially ease the job for the inversion field to correct the turbulence model. Another alternative is to directly include Re, AoA, M and transition location in the optimization problem for field inversion. However, one may expect that the optimization procedure will become more complex due to increased design variables, and normalization of all design variables will be important.

Looking toward improving the current optimization process, one can focus on starting the inversion procedure using a clever choice of the initial β field. During the discussion in Section 5.3 about starting inversion from randomly initialized inversion fields, it was observed that the output β field is as noisy as the starting β field. A first-level improvement can be made by using random inversion fields with a low-pass filter. Other avenues include initializing the field based on the wall distance using a polynomial function fit as it is seen that far away from the airfoil, the output β field is not affected much.

6 FEATURE ENGINEERING

This chapter describes the feature engineering pipeline to prepare and select the input feature for the Machine Learning (ML) part of the FIML procedure. At the end of Chapter 5, the solution of the Field Inversion procedure was achieved, which contains relevant flow information to be used for machine learning. This chapter will briefly overview how feature selection has been made previously in data-driven turbulence and FIML literature. The overview is followed by the presentation of features considered in this work and the rationale behind the choice. Finally, the discussion and results of the feature engineering pipeline used to select the input features for the machine learning activity are presented. The feature engineering methodology is the focus of the current work.

6.1 REQUIRED FEATURE PROPERTIES

The machine learning algorithm's quality depends on the data, and if the input features are amenable to the algorithm, the performance is improved manifold. The solution of the field inversion procedure will contain flow data and the resulting discrepancy field $\beta(x)$, from which an appropriate selection of the input features has to be made for the learning process. Preparation and selection of the input features are among the essential activities for solving physical problems using machine learning. These features are gathered from an extensive literature survey and selected using feature selection techniques in machine learning literature.

For the current FIML framework, the required machine learning model was presented in Equation 3.14 and has been repeated here for convenience:

$$\delta_m(\widetilde{\eta_m^*}; w) : \widetilde{\eta_m^*} \longrightarrow \beta$$

Thus, the output variable is fixed for the machine learning problem, i.e. β. The input features $\widetilde{\eta_m^*}$ need to display a strong functional relationship to the output so that the relationships can be learnt easily by the ML algorithm [Holland, 2019]. The choice of input features must be physically motivated so that the resulting function can be used to interpret the physical phenomena of the flow problem being solved [Wu et al., 2018; Wang et al., 2017]. The features must be non-dimensional and Galilean-invariant, i.e. invariant under transformations of coordinate frame and reference system to improve the generalization and extrapolating capabilities of the model. [Singh, 2018; Wang et al., 2017; Ling et al., 2016]. Conventional RANS modellers choose the flow features, keeping these considerations in mind [Spalart, 2000], and the same should be followed for data-driven RANS models. Another desirable property is that the flow features must be local or must be formulated using local features, i.e. available at every CFD grid point to facilitate ease of implementation [Wang et al., 2017; Holland, 2019]. Wall-distance (d) based features are an exception as they are not defined locally but still prove important for turbulence modelling. Matai and Durbin [2019] preferred not to use wall-modelled features to enable easier implementation in unstructured solvers. The input needs to be coming from RANS results and not directly from the high fidelity data sets to enable the machine learning model to be plugged inside the RANS model solvers [Ling and Templeton, 2015]. For RANS modelling, a popular strategy has been using the

features derived from mean flow quantities. [Wang et al., 2017]

The current FIML implementation aims to learn the discrepancy fields that can be used for a general flow condition, for instance, the use of different geometry or Reynolds numbers than that used for training. Another consideration is the number of features; more features would allow the discrepancy function to be modelled better but increase the computational cost for training and prediction. Apart from being non-dimensional, the features must be normalized in magnitude and tend towards a Gaussian distribution as the current implementation proposes to use neural networks.

6.2 METHODS IN DATA-DRIVEN TURBULENCE MODELING LITERATURE

Duraisamy et al. [2015] used a hill-climbing technique to select the feature subset for training on Gaussian processes and neural networks applied on a transition modelling problem. This algorithm appends the input features to the selected subset until the point the model stops improving. The full feature set for their application included all the elements of the velocity gradient tensor (S_{ij}, Ω_{ij}), transition-turbulence model transportation scalars (k, ω, γ) and a few non-dimensional eddy-viscosity (ν_t) related features. Ling and Templeton [2015] used raw local flow variables related to mean pressure gradient vector (∇p), turbulent kinetic energy gradient vector (∇k) in addition to the velocity gradient tensor to create a set of 12 input features based on domain knowledge and physical intuition. Furthermore, they introduced a feature normalization scheme of $\frac{x_i}{|x_i|+|norm_{x_i}|}$ where x_i is the input feature and $norm_{x_i}$ is the chosen normalization factor for x_i, which became very popular in further publications. Wang et al. [2017] used a similar set of features and normalization scheme but selectively did not apply the normalization to the features which were non-dimensional by construction. They also included features like wall-distance based Reynolds number and Q-criterion.

A lot more focus went into ensuring Galilean invariance in the input features in the works of Ling et al. [2016] and Wu et al. [2018]. A systematic approach to construct invariant bases for input features was provided by Ling et al. [2016] to eliminate missing out on crucial information by relying on physical intuition. They generated a minimal invariant set of features using the minimal integrity basis to represent all polynomial invariants ($Tr(S_{ij}^2), Tr(\Omega_{ij}^2), Tr(\Omega_{ij}S_{ij})$, ..., where Tr is the trace of the tensor) of the tensorial set containing strain rate (S_{ij}) and vorticity rate (Ω_{ij}) tensors. The Hilbert basis theorem [Hilbert and David, 1993] states that this minimal invariant set is a finite integrity basis set for a finite tensorial set ([S_{ij}, Ω_{ij}] in this case).

Wu et al. [2018] provided the most comprehensive feature engineering framework known to the author. Wu et al. [2018] started with listing the features to use based on the physical considerations of the problem they wanted to solve; not only they selected the strain-rate and vorticity rate tensors, but they also included pressure gradient (∇p) and turbulent kinetic energy gradient (∇k) vectors considering their influence on turbulence. In addition to these tensor-based features, they listed three wall distance (d) and turbulence model-based features (k, ϵ) to supplement the initial set and inform the machine learning model about wall distance and turbulent length, time scales. The normalization of features was done using the Ling and Templeton [2015] transformation. Finally, Galilean invariance is achieved by generating the minimal integrity basis from a tensorial set of [S_{ij}, Ω_{ij}, ∇p, ∇k] and choosing normalization factors which are inherently Galilean invariant. Special treatment

was done for reflection invariance considerations.

Many publications for the FIML approach use feature-selection techniques from machine learning literature after gathering a complete feature set. Holland [2019] visualized the relationship among the features by visualizing scatterplots and their correlation coefficients, which was seen in the form of heatmaps. Highly correlated features were removed from the initial set on the hypothesis that only one of the correlated features is enough to provide the information they offer [Guyon and De, 2003]. This elimination would avoid over-training and improve performance. These features were then ranked using Sequential Backward Selection (SBS) and random forest algorithms. Köhler et al. [2020] also used the SBS algorithm and used the neural network as the estimator. Singh [2018] was a critic of such methods and argued that these automatic selection processes could be misleading due to lack of repeatability of the results, favouring domain knowledge of the physical phenomena as the main criteria for selection. Matai and Durbin [2019] in their application of the FIML approach, selected the features from an initial Galilean invariant set by looking at the visual correlation of the input features with the discrepancy field β; however, they did rank the selected features using a decision tree algorithm.

6.3 SHORTLISTED FEATURES

A set of 13 features is shortlisted from a literature review of various data-driven modelling approaches. The features selected do not directly use the flow properties like $\mathbf{U}$, p or Re to enable extrapolation capabilities. All input features are non-dimensional using appropriate normalization factors. The features are also Galilean invariant (i.e. stay the same in all inertial frames) because either they have terms associated with elements of the gradient tensor/vector of the flow properties, or they are scalar features. The features using the tensors are also rotational invariant because they have been constructed using anti-symmetric tensors (i.e. using S_{ij} and Ω_{ij}).

Additionally, most selected features are local, but some of them are based on wall-distance based properties. Wall-distance-based features may reduce their applicability in non-structured grids, but their information is valuable for the current test case of turbulent separated flows. There were two primary considerations while shortlisting this feature set: a) description of the shock-induced separation phenomena and b) functional relation to the turbulence model with a focus on SA model properties. The following features were shortlisted:

1. χ: Normalized SA kinematic viscosity or turbulent viscosity. $\tilde{\nu}$ is the transport important variable for the SA-model. This feature has proved to be crucial in FIML applications using the SA model. The feature is defined as:

$$\chi = \frac{\tilde{\nu}}{\nu} \tag{6.1}$$

   where ν is the kinematic laminar viscosity of the flow.

2. $\frac{P}{D}$: The ratio of the production to the destruction term in the SA-neg model. Using this feature areas of turbulent production changes will be identified. The exact formulation of the feature follows from the SA-neg model terms explained in Section A.1:

$$\frac{P}{D} = \frac{c_{b1}(1 - f_{t2})\widetilde{\Omega}\tilde{\nu}}{\left(c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right)\left[\frac{\tilde{\nu}}{d}\right]^2} \tag{6.2}$$

3. f_w: SA wall function as formulated in Equation A.5. This feature was used by Holland [2019] to reliably predicted adverse pressure gradient flows on an airfoil.

4. $\overline{\nabla\tilde{\nu}}$: Normalized version of SA viscosity gradient magnitude. The hope with this feature is to gain information about regions of strong variation of eddy viscosity. This was used by Ferrero et al. [2020] for their FIML implementation for turbomachinery flows using SA model.

$$\overline{\nabla\tilde{\nu}} = \frac{d}{\nu + \tilde{\nu}}|\nabla\tilde{\nu}| \tag{6.3}$$

where d is the wall distance

5. $\overline{\Omega}$: Normalized vorticity tensor magnitude, used for identifying regions of vorticity in the flow.

$$\overline{\Omega} = \frac{d^2}{\nu + \tilde{\nu}}\Omega, \tag{6.4}$$

where $\Omega = \|\Omega_{ij}\|$ i.e. the magnitude of vorticity and d is the wall distance.

6. δ: Ratio of the local turbulent strain rate to the shear stress. This feature originates from the work by Medida [2014] where this term was used as an adverse pressure gradient correction term for the SA model. This feature was identified to be important for FIML approaches using SA model [Holland, 2019; Jäckel, 2020].

$$\delta = \frac{\mu_T S}{1.5\tau_w} \tag{6.5}$$

where τ_w is a reference quantity based on a wall based quantity u_τ i.e. the skin friction velocity at the nearest wall point. $\tau_w = 0.5\rho u_\tau^2$ and $S = \|S_{ij}\|$ i.e. the magnitude of strainrate tensor.

7. $\frac{S}{\Omega}$: Strainrate to vorticity magnitude ratio. This information will provide the information about the full velocity gradient tensor but will still remain acceptable under the feature invariance requirements laid down earlier.

$$\frac{S}{\Omega} = \frac{\|S_{ij}\|}{\|\Omega_{ij}\|} \tag{6.6}$$

where, $\Omega = \|\Omega_{ij}\|$ i.e. the magnitude of vorticity tensor and $S = \|S_{ij}\|$ i.e. the magnitude of strainrate tensor.

8. $\frac{\tau}{\tau_{ref}}$: Normalized Reynolds stress tensor magnitude. This feature is normalized using a normalization factor constructed using local quantities i.e. $\tau_{ref} = \frac{\rho(\nu + \tilde{\nu})^2}{d^2}$. This is unlike in δ where a wall distance based normalization was used. Here $\tau = \|\tau_{ij}\|$ or the magnitude of Reynolds stress tensor. This feature is also inspired from the work of Ferrero et al. [2020].

9. H_{12}: The boundary layer shape factor H_{12} is a classical boundary layer parameter which gives information about nature of the flow and the adversity of the pressure gradient. Typically, low values of shape factor (1.3-1.4) indicate full boundary layer profiles which are associated to turbulent flows.

$$H_{12} = \frac{\delta^*}{\theta} \tag{6.7}$$

where δ^* = displacement thickness, θ = momentum thickness of the boundary layer.

10. k_{QCR}: Quadratic constitutive relation (QCR) for SA models. This feature originates from the work by Mani et al. [2013] where inadequacy of eddy viscosity turbulence models to predict secondary vortices (primarily for internal flows) was addressed by adding this relation to the SA model. A part of this relation was used by Volpiani et al. [2021] to reconstruct a turbulent kinetic energy term for the SA model in their machine learning based data-driven turbulence modelling framework for separated flows. This feature is defined as follows:

$$k_{QCR} = 1.5 C_{Cr2} \nu_t \sqrt{2 S_{ij} S_{ij}} \tag{6.8}$$

where $C_{Cr2} = 2.5$ [Mani et al., 2013], ν_t is the eddy viscosity and S_{ij} is the strainrate tensor.

11. f_d: DDES wall shielding function. This feature is used in a Detached-eddy simulation framework for SA models where the switching between LES (typically near the wall) and RANS (far from wall) is done using this function, where $f_d = 1$ implies the LES region. [Spalart et al., 2006]. This feature will be useful to provide information about the boundary layer and the near-wall regions.

$$f_d = 1 - tanh(8 r_d^3), \quad r_d = \frac{\tilde{\nu}}{\kappa^2 d^2 \sqrt{\frac{\partial u_i}{\partial x_j} \frac{\partial u_i}{\partial x_j}}} \tag{6.9}$$

r_d is a parameter for the length scale defined for the SA model but can be used for any eddy-viscosity model. r_d is 1 in the log-layer near the wall and falls to zero near the edge of the boundary layer. Köhler et al. [2020] used this feature for their FIML implementation and Ferrero et al. [2020] used a modified version to suit turbomachinery flows.

12. β_{RC}: Rotta and Clauser pressure gradient parameter [Clauser, 1954]. This parameter is used to provide pressure gradient information for equilibrium/near-equilibrium adverse pressure gradient turbulent boundary layers. This feature will provide pressure gradient information.

$$\beta_{RC} = \frac{\delta^*}{\rho u_\tau^2} \frac{\partial p}{\partial s} \tag{6.10}$$

where δ^* = displacement thickness of the boundary layer, ρ is the density and u_τ i.e. the skin friction velocity at the nearest wall point. $\frac{\partial p}{\partial s}$ stream-wise pressure gradient $\ln(\beta_{RC})$

13. Δp_{s+}: Inner pressure gradient parameter [Mellor, 1966]. This parameter used to extend the classical law of the wall in the viscous sublayer region of the boundary layer with the assumption that viscous shear stresses are dominant compared to Reynolds stresses [Patel, 1973]. Here, it is just used as a normalized form of streamwise pressure gradient $\frac{\partial p}{\partial s}$.

$$\Delta p_{s+} = \frac{\nu}{\rho u_\tau^3} \frac{\partial p}{\partial s} \tag{6.11}$$

6.4 CURRENT FEATURE ENGINEERING PIPELINE

This section details the steps taken by the author to finally select a subset of the shortlisted 13 features to be used for machine learning activity. The aim is to achieve the smallest subset of input features required to accurately capture all the flow

phenomena. The steps presented here consider neural networks as the estimator and vary for other ML algorithms like Gaussian processes or Random Forests. The discussion will now continue for the feature data processed from the inversion solution at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960, which gives a dataset of 157116 points to start with. The results of this activity will vary when the dataset is changed, which may be done using more flow cases together or using data with different transformations (applied using mathematical operations).

6.4.1 Checking for correlations

The first step in the pipeline is to assess the correlations between the input features and the target variable $\beta(x)$, where a high correlation between the input and output makes a model approximation easier, ultimately improving the learning process. Additionally, the correlations of input features are compared with other input features. This activity is based on the idea that truly correlated variables do not add any additional information to the model, and only one is representative of a truly correlated set [Guyon and De, 2003]. In this study, this is done by studying scatter plots and using Spearman's correlation coefficient (Equation 2.33). The feature data originating from the inversion solution is based on a RANS solver and is inherently non-linear. Therefore Spearman's correlation is a good choice as it does not make any assumptions about the probability distribution of the data.

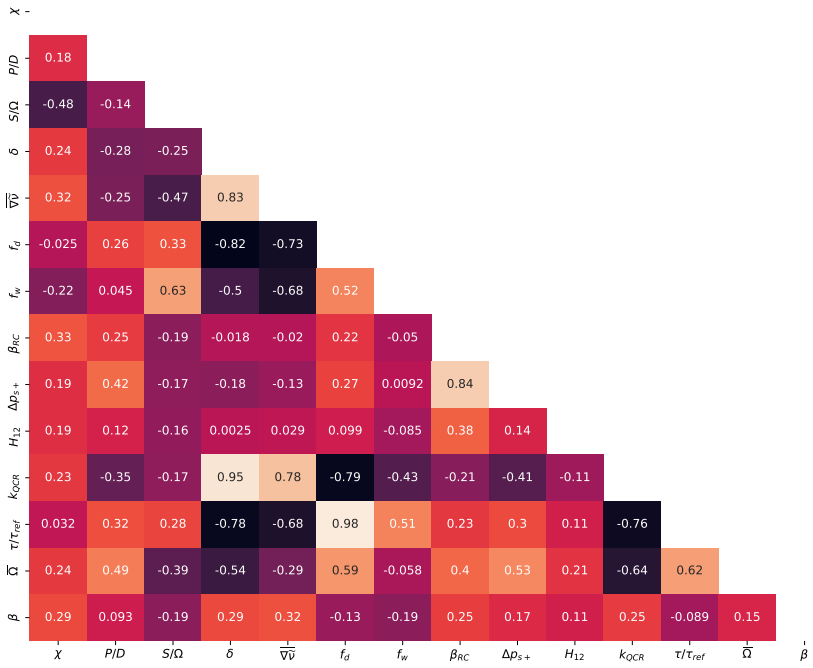


Figure 6.1: Heatmap for Spearman's correlation coefficient for the feature data processed from the inversion solution at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960. No filters have been applied, the number of points in the dataset for the heatmap is 157116. β is the target variable in the data.

Figure 6.1 gives the correlation heatmap for the full feature set. The row for the target variable represents the correlation of the target variable with all the input features, and the correlation with itself is equal to one (true correlation). First, high absolute values of correlation for any two input features are identified with a correlation value of more than 0.9 regarded as high. One of the two input features is then eliminated based on their correlation with the target variable. For Figure 6.1 this results in removal of:

- k_{QCR}: Highly correlated with δ (0.95) and lower correlation with β. In this case, both these expressions are proportional to $\nu_t S$.
- $\frac{\tau}{\tau_{ref}}$: Highly correlated to f_d (0.98) and lower correlation with β. Here, the mathematical similarity is not apparent.

6.4.2 Applying transformations and removing features with information loss

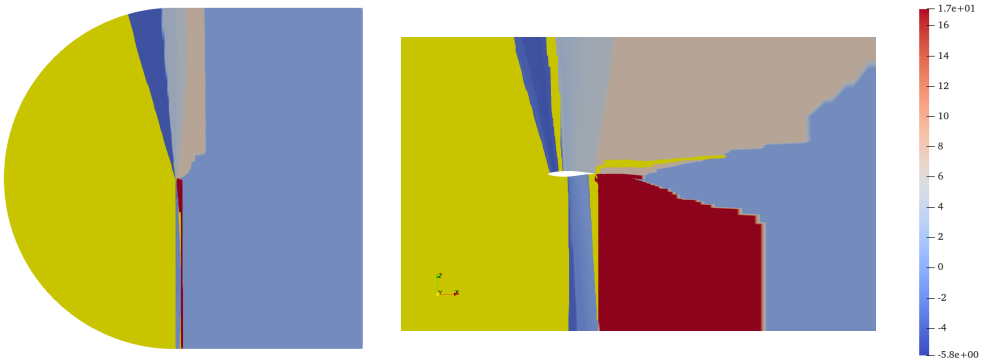
The input features in the dataset may vary a lot in magnitudes. Additionally, the data may have outliers, or the probability distribution of the input features is highly skewed. All these factors may limit the neural network's performance, and it is crucial to apply appropriate transformations to make them amenable. For the current project, a $\ln(q_\beta)$ is applied where ln is the natural logarithmic operator, and q_β is the input feature. This operator has an effect of changing the distribution of data which is needed in our dataset in consideration as most features have extreme outliers and varying magnitudes (the distribution can be viewed in Figure C.1). The transformation was applied selectively to some features to match the magnitude of the target variable (which is ≈ 1), and this can be viewed in Table 6.1.

Feature (q_β)	Description	Feature definition	Final feature
q_1	Normalized SA viscosity	$\chi = \frac{\tilde{\nu}}{\nu}$	$\ln(\chi)$
q_2	Production to Destruction ratio	$\frac{P}{D} = \frac{c_{b1}(1-f_{t2})\tilde{\Omega}\tilde{\nu}}{\left(c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right)\left[\frac{\tilde{\nu}}{d}\right]^2}$	$\ln(\frac{P}{D})$
q_3	SA wall function	f_w	f_w
q_4	Normalized SA viscosity gradient magnitude	$\overline{\nabla\tilde{\nu}} = \frac{d}{\nu+\tilde{\nu}}\lvert\nabla\tilde{\nu}\rvert$	$\ln(\overline{\nabla\tilde{\nu}})$
q_5	Local turbulent strainrate to the shear stress	$\delta = \frac{\mu_T\lVert S_{ij}\rVert}{1.5\tau_w}, \tau_w = 0.5\rho u_\tau^2$	$\ln(\delta)$
q_6	Strainrate to vorticity magnitude ratio	$\frac{S}{\Omega} = \frac{\lVert S_{ij}\rVert}{\lVert\Omega_{ij}\rVert}$	$\ln\left(\frac{S}{\Omega}\right)$
q_7	Normalized Reynolds stress tensor magnitude	$\frac{\tau}{\tau_{ref}} = \frac{\lVert\tau_{ij}\rVert}{\tau_{ref}}, \tau_{ref} = \frac{\rho(\nu+\tilde{\nu})^2}{d^2}$	$\ln\left(\frac{\tau}{\tau_{ref}}\right)$
q_8	Boundary layer shape factor	$H_{12} = \frac{\delta^*}{\theta}$	H_{12}
q_9	DDES wall shielding function	$f_d = 1 - tanh(8r_d^3)$	f_d
q_{10}	QCR for TKE	$k_{QCR} = 1.5C_{Cr2}\nu_t\sqrt{2S_{ij}S_{ij}}$	$\ln(k_{QCR})$
q_{11}	Normalized vorticity tensor magnitude	$\overline{\Omega} = \frac{d^2}{\nu+\tilde{\nu}}\lVert\Omega_{ij}\rVert$	$\ln(\overline{\Omega})$
q_{12}	Rotta and Clauser pressure gradient parameter	$\beta_{RC} = \frac{\delta^*}{\rho u_\tau^2}\frac{\partial p}{\partial s}$	$\ln(\beta_{RC})$
q_{13}	Inner pressure gradient parameter	$\Delta p_{s+} = \frac{\nu}{\rho u_\tau^3}\frac{\partial p}{\partial s}$	$\ln(\Delta p_{s+})$

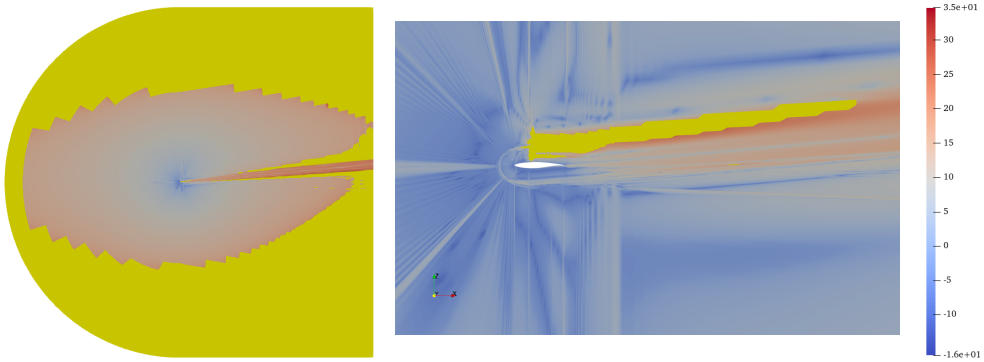
Table 6.1: Shortlisted input features from literature to be used for the neural network training, and final feature after applying the $\ln(q_\beta)$ transformation to selected features. q_7 and q_{10} were removed after checking for correlations, but have still been mentioned in this table for the sake of completeness.

Due to the nature of ln operator, there is a risk of losing information as $\ln(x) < 0$ is not defined. Information loss happens when the input features values are less than or equal to zero. These features can not be used at a given grid point; all the features must be available to proceed further, or it gets filtered out. If too many points are filtered from the dataset, it leads to the loss of crucial information. In this case, the feature is removed from the input set of features. One may argue that a choice of a different transformation (like a scaling operation by Ling and Templeton

[2015], or a quantile transformer by [Holland, 2019]) may remove this problem altogether. However, ln operator was preferred because it changes the distribution of the input features, and the nature of the operator is well understood. Furthermore, losing out on a few features may result in loss of crucial information, possibly deteriorating model performance, but arguably the current, complete feature set is big enough for this not to be a problem.



(a) Contour of β_{RC}



(b) Contour of P/D ratio

Figure 6.2: Feature contours after applying the $\ln(q_\beta)$ transformation for the feature data from the inversion solution at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960. Yellow color displays the NANs in the domain after applying the transformation, which corresponds to information loss

Figure 6.2 shows the contours for features most affected by the transformation. The region in yellow signifies the regions where the information is lost. $\ln(\beta_{RC})$, $\ln(\Delta p_{s+})$ (as Δp_{s+} formulation similar to β_{RC}, based on the pressure gradient) and $\ln(\frac{P}{D})$ are removed from the input feature set. For $\ln(\beta_{RC})$ the regions of favourable pressure gradient and for $\ln(\frac{P}{D})$ regions where the production term is zero are lost. The remaining feature set comprises of q_1, q_3, q_4, q_5, q_6, q_8, q_9, q_{11} from Table 6.1. Their distribution can be viewed in Figure C.2.

6.4.3 Sequential feature selection for final feature subset

The final step to select the input features before the feature data goes to the machine learning procedure is using Sequential feature selection (SFS) algorithms [Ferri et al., 1994]. The SFS algorithms are a family of greedy search algorithms that automatically select a subset of features most important to solving the problem. SFS allows the use of a user-preferred estimator and performance metric. The goal of using this algorithm in the feature selection pipeline is two-fold: a) to estimate the number of features finally needed for machine learning, and b) to figure out which features

perform well with neural networks as the estimator.

Traditional SFS algorithms work either in a *forward selection* or a *backward elimination* mode. In the forward mode, a null estimator (neural network) is started to fit with one feature at a time, and the best one is selected according to the scoring parameter provided to the model (e.g. mean squared error). In the second step, another feature is added in combination with the previously selected feature and the best combination of two is selected. This process goes on until a subset of the required number of features is an output. In the backward mode, the model starts to fit with all features and removes one by one to achieve the best combination.

The algorithm used for this activity is the Bidirectional elimination variant of the SFS algorithm family. This variant is also known as the "floating" variant in the SFS package in mlxtend [Raschka, 2018]. This combines a *forward selection* and *backward elimination* in one algorithm to sample a larger number of feature subset combinations. Suppose in a full feature space of n features, k best features are required where $k < n$. In a forward setting at the i^{th} step, the bidirectional variant selects a subset of i features using forward selection where $i < k$. Now, the significance of already selected i features is checked using backward elimination (line 7), and any low performing features are removed. This extra step ensures a larger number of combinations are tried. In a backward setting, the opposite process of first backward elimination and then the forward selection is applied. The algorithm for a forward mode is explained in Algorithm 6.1.

Algorithm 6.1: Bidirectional elimination SFS variant in the forward mode

Input: Full feature set $Y = \{y_i\}, i \in [1, n]$, $\mathcal{J}$ = estimator
Output: Feature subset $X = \{x_j\}, j \in [1, k], k < n$

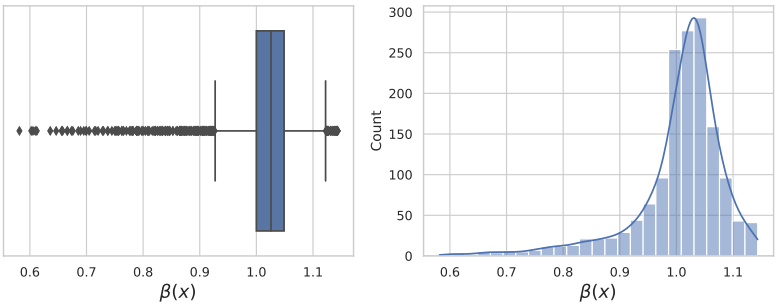
1 Initialize $X_0 = , i = 0$;
2 **while** $i == k$ **do**
3 x^+ = argmax $\mathcal{J}(X_i + x)$, where $x \in Y - X_i$;
4 $X_{i+1} = X_i + x^+$;
5 $i = i + 1$;
6 x^- = argmax $\mathcal{J}(X_i - x)$, where $x \in X_i$;
7 **if** $\mathcal{J}(X_i - x) > \mathcal{J}(X_i)$ **then**
8 $X_{i-1} = X_i - x^-$;
9 $i = i - 1$;

A neural network of 2 layers with 50 neurons each was passed through the SFS algorithm. The activation function was chosen to be 'relu', and a constant learning rate of 0.001 was set. The automatic batch size option was chosen, and the solver was the 'Adam' optimizer [Kingma and Ba, 2014]. This neural network model was implemented using the "MLPRegressor" class in *scikit-learn* python library [Pedregosa et al., 2011]. This network architecture is not the one that will be finally used for the final neural network training; however, at this point, the aim is to involve neural networks at some stage of the feature engineering process.

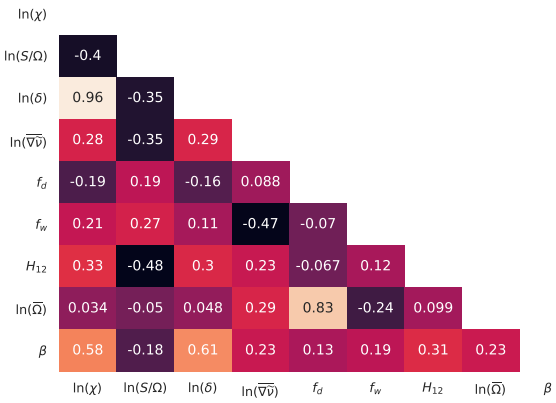
The training data to be passed through the SFS algorithm must be similar to the one used for the final machine learning process. The original data has 157116 points, and the majority of the target variable values are one as $\beta(x) = 1$ for most CFD grid points and only varies close to the airfoil. To avoid the neural network to predict $\beta(x) = 1$ everywhere, a suitable method for filtering the data has been devised as follows:

- Select the "Non-ones": The data corresponding to a range far away from $\beta(x) = 1$ are selected as it is to be used for training. Currently, this range is set to be $\beta \notin (0.98, 1.02)$.
- Randomly choose the "Ones" and collate: In the step above, the data points not selected are called "Ones", and their range is $\beta \in [0.98, 1.02]$. Suppose the number of "Non-ones" are n in number. From the "Ones" left, x% of n data-points are chosen randomly to be collated with the "Non-Ones". Currently, x percentage is chosen to be 20.

The resulting distribution of the target variable β can be seen in Figure 6.3a. The design choices while constructing this training data is crucial to the performance of the machine learning model, and the effect of these choices will be discussed in the next chapter (Chapter 7). After applying the filters, only $\approx$ 1500 points are left out of 157116 points. The correlation heatmap for the features in the remaining set can be viewed in Figure 6.3b



(a) Box and Whisker plot (left) and Histogram (right) for the distribution of β



(b) Spearman's correlation heatmap. β is the target variable in the data.

Figure 6.3: Information about remaining data to be used for SFS algorithm after applying filter criteria of $\beta \notin (0.98, 1.02)$. The filter is applied on remaining $\ln(q_\beta)$ features from the inversion solution at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960

At this stage, the SFS algorithm is ready to be applied with one important decision left to the author; the number of features to select in the final subset. The following methodology is defined:

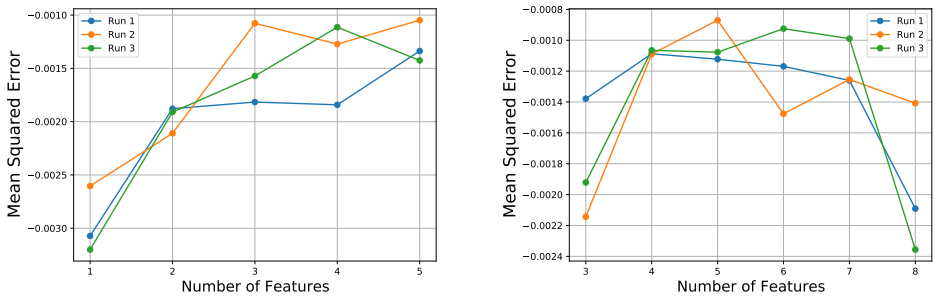
- Set an input range of length of the best subset. For the current data (3-5) is chosen, i.e. the best subset of 3-5 features is provided as an output by the algorithm from the eight input features.
- Run the SFS algorithm multiple times, because the results of the algorithm are non-deterministic due to the involvement of neural networks. This was done three times each for the current data in both forward and backward mode.

- Score the features for every run of SFS. The feature subset provided will contain somewhere between 3-5 features. Provide the score of '5' to the most important feature and decrements of one until the last feature is assigned a score.
- Total the score for a given feature. The feature with the maximum score is the most preferred.

Another advantage this methodology provides is that it somewhat accounts for the variability in results that can be there while running SFS algorithms. The results of the SFS runs can be viewed in Table 6.2 and Figure 6.4.

Score	**Forward floating search**			**Backward floating search**		
(Priority)	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
5	$\ln(\chi)$	$\ln(\chi)$	$\ln(\chi)$	$\ln(\chi)$	$\ln(\chi)$	$\ln(\chi)$
4	$\ln\left(\frac{S}{\Omega}\right)$	$\ln(\delta)$	$\ln\left(\frac{S}{\Omega}\right)$	$\ln(\delta)$	$\ln\left(\frac{S}{\Omega}\right)$	$\ln(\overline{\Omega})$
3	$\ln(\overline{\nabla\tilde{v}})$	$\ln(\overline{\nabla\tilde{v}})$	$\ln(\overline{\Omega})$	$\ln(\overline{\Omega})$	$\ln(\delta)$	f_d
2	f_d	$\ln(\overline{\Omega})$	H_{12}	H_{12}	$\ln(\overline{\Omega})$	H_{12}
1	H_{12}	f_w			f_d	

Table 6.2: Bidirectional SFS runs with neural networks as the estimator.



(a) Forward floating selection runs

(b) Backward floating selection runs

Figure 6.4: Negative of mean squared error at different stages of the Bidirectional SFS runs with neural networks. The number of features with the maximum score i.e. closest to zero has the best performance. The Bidirectional SFS algorithm looks for the best score in the input range provided to the algorithm, which is (3-5) features in this case

Feature		**SFS score**	**Correlation with $\beta(x)$**	
			Full data	Training data
q_1	$\ln(\chi)$	30	0.29	0.58
q_{11}	$\ln(\overline{\Omega})$	14	0.15	0.23
q_6	$\ln\left(\frac{S}{\Omega}\right)$	12	-0.19	-0.18
q_5	$\ln(\delta)$	10	0.29	0.61
q_8	H_{12}	7	0.11	0.31
q_4	$\ln(\overline{\nabla\tilde{v}})$	6	0.32	0.23
q_9	f_d	6	-0.13	0.13
q_3	f_w	1	-0.19	0.19

Table 6.3: Selection table for input features to be used for neural network training

Table 6.3 summarizes the scores for all the features and gives their correlations with the β, sorted in decreasing order of the SFS scores. This table is used to finally decide the priority order of the features for machine learning. From Figure 6.4 it

is seen that the choice of 3-5 features was an agreeable decision, but in Figure 6.4b shows that the backward Run 3 has the best performance with a six feature subset. The backward Run 3 output in Table 6.2 has four features which is the second-best. An advantage of finalizing features from Table 6.3 is that finally, a desired number of features can be chosen considering both SFS and correlation results. The author gives more priority to the SFS results, and correlation results are used as a tie-breaker. Special care must be taken if the correlation with β is low, which is not the case in Table 6.3.

Using Table 6.3, f_w and $\overline{\nabla\widetilde{\nu}}$ are removed from the input features. The final feature set is the following:

$$\text{Features} = [q_1, q_5, q_6, q_8, q_9, q_{11}] = \left[\ln(\chi), \ln(\delta), \ln(\frac{S}{\Omega}), H_{12}, f_d, \ln(\overline{\Omega})\right] \quad (6.12)$$

f_w is not selected enough by SFS to warrant further usage. For $\overline{\nabla\widetilde{\nu}}$ the SFS score is low but the fact that it is highly correlated to δ (Figure 6.1) is the reason that causes its elimination. The six remaining features are used to train the final neural networks, discussed in the next chapter. This selection does not mean that a good neural network training can not be done with features less than six; in fact, the next chapter discusses the effect of further reducing the features. This reduction will be primarily based on visual correlation of the input feature with the target variable β and its rank in the Table 6.3.

6.5 RESULTS FOR THE FULL DATABASE

The procedure described in Section 6.4 is also applied to a larger dataset. This dataset is generated by collating all the flow data from the selected inversion solutions in Table 5.2. The input features are then processed from this flow data using the feature definitions provided in Section 6.3.

The first step is to check the correlations in the new dataset to assess feature importance and remove redundant highly correlated features. The heatmap for the Spearman's correlation coefficient for this dataset is given in Figure 6.5. The correlation values are not highly different from the ones seen in in Figure 6.1. This is because all the flow cases selected for inversion are expected to have shock induced separation and the values of M and AoA are not very different from each other. From the correlation heatmap, the following features were (again) eliminated:

- k_{QCR}: Highly correlated with δ (0.94) and lower correlation with β.
- $\frac{\tau}{\tau_{ref}}$: Highly correlated to f_d (0.98) and lower correlation with β.

The second step is to apply the ln transformation to the features and removing features with major information loss. Here, it was difficult to visualize the inversion data for five different flow cases on one computational grid. However, due to the flow phenomena being similar for all cases considered the author expects that the features $\ln(\beta_{RC})$, $\ln(\Delta p_{s+})$ and $\ln(\frac{P}{D})$ will again incur significant loss of information (as seen in Figure 6.2). This approach works for this specific training data, but for a more general model with different type of flow phenomena in the training data will warrant a more careful treatment. The remaining feature set comprises of q_1, q_3, q_4, q_5, q_6, q_8, q_9, q_{11} from Table 6.1. Their probability distribution can be viewed in Figure C.5. The third step is to apply the SFS algorithm to the remaining feature data. Before this, the data was again filtered appropriately using the same criteria ("Non-Ones" and "Ones') on $\beta(x)$ to avoid the neural network to learn to predict $\beta(x) = 1$ everywhere.

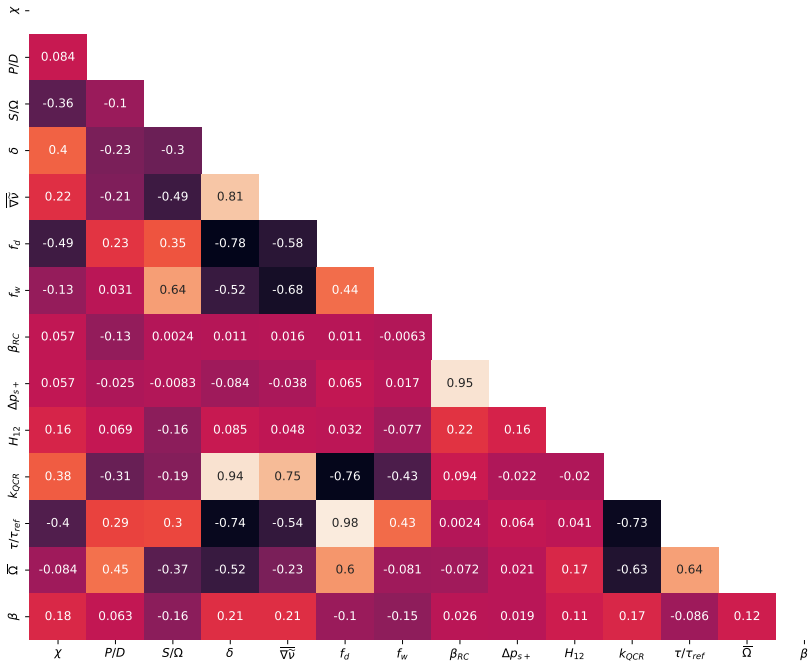
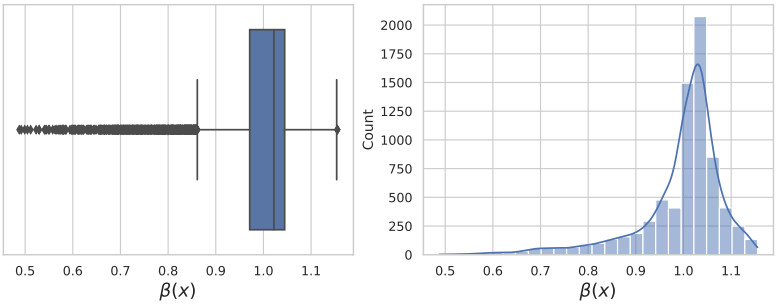
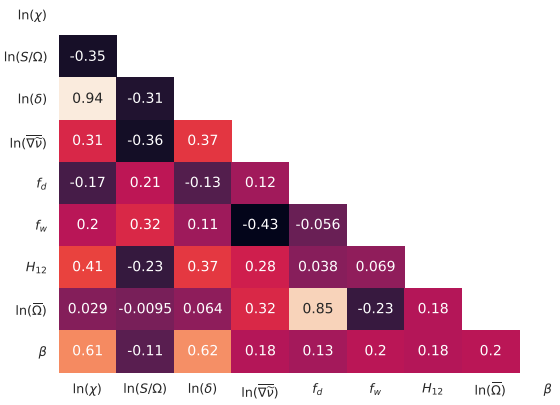


Figure 6.5: Heatmap for Spearman's correlation coefficient for the feature data processed from all the inversion solutions at flow conditions in Table 5.1. No filters have been applied, the number of points in the dataset for the heatmap is 782980. β is the target variable in the data.



(a) Box and Whisker plot (left) and Histogram (right) for the distribution of β



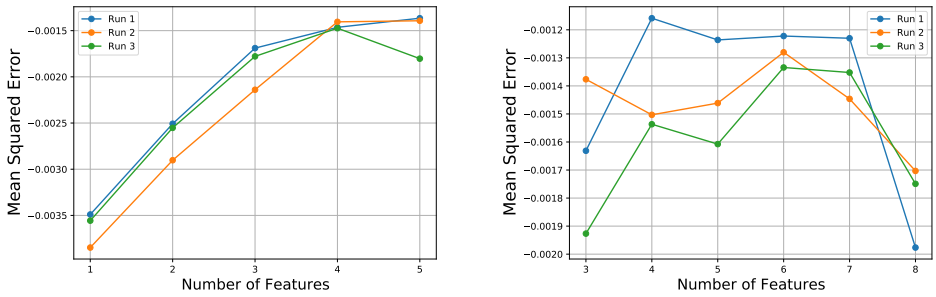
(b) Spearman's correlation heatmap. β is the target variable in the data.

Figure 6.6: Information about remaining data to be used for SFS algorithm after applying filter criteria of $\beta \notin (0.98, 1.02)$. The filter is applied on remaining $\ln(q_\beta)$ features from all the inversion solutions at flow conditions in Table 5.1.

This resulted in a training dataset of $\approx$ 7100 points from a total of 782980 points. The resulting probability distribution of the target variable β and correlation of the input features in the final training data can be seen in Figure 6.6. Finally, the SFS algorithm is run on the data thrice, in both forward and backward modes. The results can be seen in Table 6.4 and Figure 6.7. From Figure 6.7 it can be seen that 4-6 features will be required.

Score (Priority)	Forward floating search			Backward floating search		
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3
5	$\ln(\chi)$	$\ln(\chi)$	$\ln(\chi)$	$\ln(\chi)$	$\ln(\chi)$	$\ln(\chi)$
4	$\ln\left(\frac{S}{\Omega}\right)$	$\ln(\delta)$	$\ln(\delta)$	$\ln(\delta)$	$\ln(\overline{\Omega})$	$\ln\left(\frac{S}{\Omega}\right)$
3	$\ln(\delta)$	$\ln(\overline{\Omega})$	$\ln(\overline{\Omega})$	$\ln(\overline{\Omega})$	H_{12}	$\ln(\delta)$
2	$\ln(\overline{\Omega})$	f_d	H_{12}	H_{12}		$\ln(\overline{\Omega})$
1	H_{12}	f_w				

Table 6.4: Bidirectional SFS runs with neural networks as the estimator.



(a) Forward floating selection runs

(b) Backward floating selection runs

Figure 6.7: Negative of mean squared error at different stages of the Bidirectional SFS runs with neural networks. The number of features with the maximum score i.e. closest to zero has the best performance. The Bidirectional SFS algorithm looks for the best score in the input range provided to the algorithm, which is (3-5) features in this case

Feature		SFS score	Correlation with $\beta(x)$	
			Full data	Training data
q_1	$\ln(\chi)$	30	0.18	0.61
q_5	$\ln(\delta)$	18	0.21	0.62
q_{11}	$\ln(\overline{\Omega})$	17	0.12	0.20
q_6	$\ln\left(\frac{S}{\Omega}\right)$	8	-0.16	-0.11
q_8	H_{12}	8	0.11	0.18
q_9	f_d	2	-0.1	0.13
q_3	f_w	1	-0.15	0.20
q_4	$\ln(\overline{\nabla\tilde{\nu}})$	0	0.21	0.18

Table 6.5: Selection table for input features to be used for neural network training

Table 6.5 summarizes the scores for all the features and gives their correlations with the β for the new data, sorted in decreasing order of the SFS scores. Using Table 6.5, f_w, f_d, and $\overline{\nabla\tilde{\nu}}$ are removed from the input features simply because they are not selected enough by the SFS algorithm. The resulting set of features for this data to be used for machine learning are as follows:

$$\text{Features} = [q_1, q_5, q_6, q_8, q_{11}] = \left[\ln(\chi), \ln(\delta), \ln(\frac{S}{\Omega}), H_{12}, \ln(\overline{\Omega})\right] \tag{6.13}$$

7 MACHINE LEARNING

This chapter is about the Machine Learning (ML) part of the FIML procedure. At the end of Chapter 6, a list of features was provided to be used for neural network training. This chapter will describe the training process chosen by the author and motivate the design choices for the selection of training hyperparameters and neural network architecture. Following this, the results from neural network training will be presented where analysis is done on the effect of using a smaller subset of features than the one to start with. Finally, the results of connecting these trained neural networks to the TAU flow solver are provided i.e. the NN-augmented SA-neg model. Testing is done on various flow conditions to evaluate the generalization capability of the current augmented turbulence model.

7.1 MACHINE LEARNING METHODOLOGY

From the shortlisted input features $\widetilde{\eta_m^*}$, an accurate and robust machine learning (ML) model is to be developed. The aim is to identify patterns in the spatial β field and its relation to $\widetilde{\eta_m^*}$. The required machine learning model was presented in Equation 3.14 and has been repeated here for convenience:

$$\delta_m(\widetilde{\eta_m^*}; w) : \widetilde{\eta_m^*} \longrightarrow \beta$$

Neural networks (NN) are selected as the default choice of the machine learning algorithm for this study, mapping input features from the model $\widetilde{\eta_m^*}$ to output β. Neural networks are one of the most popular methods for supervised ML. They work well for non-linear function approximations and provide a high generalisation ability. Deep neural networks have been widely used in literature for aerodynamic predictions using high-fidelity data sources (i.e. experiments, simulations). Many variants of neural networks like sparsely connected deep NNs, convolutional neural networks (CNNs) and Recurrent neural networks (RNNs) have found application in flow modelling [Brunton et al., 2019]. However, this study will employ a fully-connected neural network, a Multi-layer perceptron (MLP). The design choices for its architecture and the training hyperparameters will be detailed in the upcoming sections. For the neural network implementation, the TensorFlow library [Abadi et al., 2015] with the Keras backend is used.

The selected network architecture is trained on the feature data generated at the end of Chapter 6 and their respective selected features (e.g. Equation 6.12). This trained NN is then plugged into the TAU flow solver to provide augmented SA-neg model predictions. Specifically, tests on various combinations of input features to train the neural networks will be presented to determine the best feature subset. The performance of the NN-augmented predictions will be assessed by comparing them with the wind tunnel results.

7.1.1 Choice of network architecture and hyperparameters

Determining the network architecture and optimising the learning hyperparameters is crucial to achieving a robust ML model. The current implementation follows the methodology by Sabater et al. [2021], where a Multi-Layer Perceptron (MLP) with a shrinking network shape is employed. The shape of the network is defined using three parameters: a) the initial number of neurons in the first hidden layer, b) the number of hidden layers, and c) the shrinkage factor. An example of such a network can be seen in Figure 7.1.

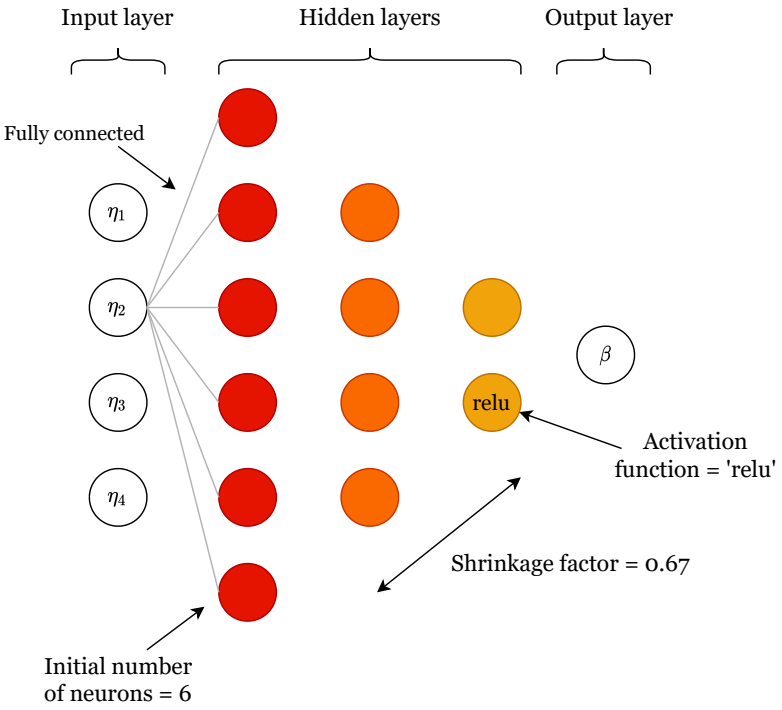


Figure 7.1: Multi-layer perceptron architecture by Sabater et al. [2021]: For a set of input features and output the architecture is defined using three shape parameters: a) number of hidden layers, b) shrinkage factor, and c) number of neurons in first layer.

All layers consist of a Rectified linear unit (ReLU) [Glorot et al., 2011] activation function. It is the most commonly used activation function in deep learning applications owing to its simplicity while introducing non-linearity in the network. Even though ReLU is not differentiable like *sigmoid* or *tanh* activation functions, but it solves the problem of gradients diminishing in magnitude during backward propagation [Tan and Lim, 2019], also known as the "vanishing gradient problem". This problem is typical in MLPs, and its symptoms are a slow rate of improvement of the model during training, continued training providing little improvement and little/no effect of increasing the size of the network.

The learning rate of the neural network is chosen to be an exponential decrease with the number of epochs (n_{epoch}). The learning rate is defined as:

$$l_r = 10^{-l_0} \exp(10^{-K_l} \times n_{epoch}) \tag{7.1}$$

where 10^{-l_0} is the initial learning rate, and l_0 is the initial learning rate exponent. 10^{-K_l} is the learning rate decay with K_l as the learning rate decay exponent. An appropriate learning rate is needed so that the model does not get stuck while learning at a low rate, but it also should not be faster than needed as it could converge the network to a sub-optimal solution. The final hyperparameter for the

neural network is the batch size, which is the number of data points that will be propagated through the network at one time. Therefore, for the current application, the neural network can be fully defined using the following set of hyperparameters:

1. number of hidden layers,
2. number of neurons in the first hidden layer,
3. shrinkage factor,
4. initial learning rate exponent (l_0),
5. learning rate decay exponent (K_l), and
6. batch size.

The optimal choice of these six hyperparameters is made using a Surrogate Based Optimization (SBO) approach used by Sabater et al. [2021]. In an SBO, a metamodel is generated to replace the original, expensive black-box model resulting in a significant reduction of function evaluations to find the optimal solution [Forrester and Keane, 2009]. This optimization routine is implemented using the DLR-SMARTy framework [Görtz et al., 2013] with a Keras-TensorFlow backend [Abadi et al., 2015] to deal with the neural network evaluations. To start the optimization, the user-defined input ranges of the design variables (hyperparameters) need to be provided. The objective function for the optimization is the Mean Square Error (MSE) of the neural network prediction on the validation data or the validation loss. These hyperparameters are a combination of discrete and continuous variables. The output of the optimization is in a decimal form, which is approximated to the nearest integer for discrete variables like number of hidden layers.

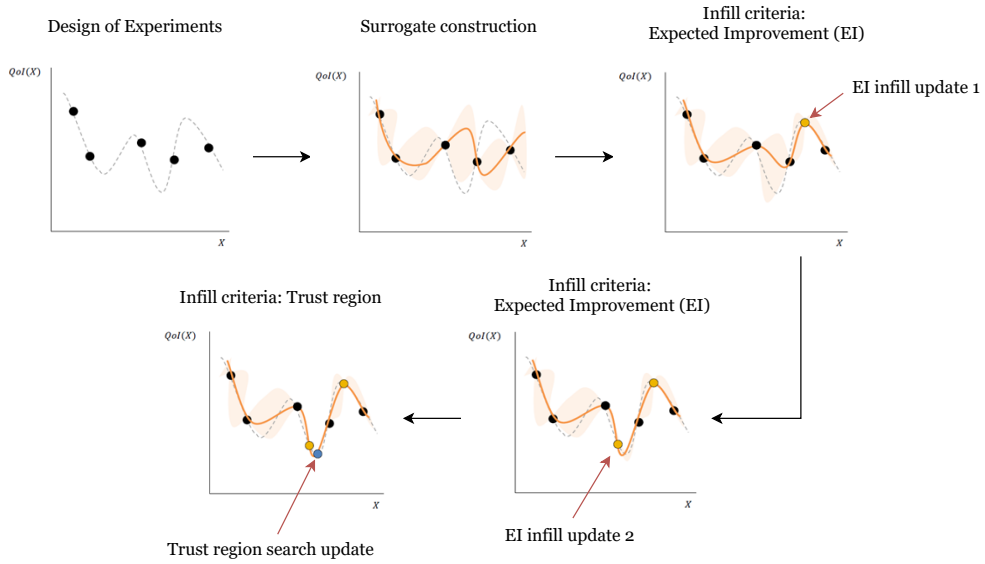


Figure 7.2: Optimization procedure used for choosing the hyperparameters for the neural network in the current study. The grey dotted line represents the true function and the surrogate model approximates this function.

The steps in the current optimization process follow from [Sabater et al., 2020] and can be seen in Figure 7.2. The following optimization steps are implemented:

1. Design of Experiments: First, a Design of Experiments (DoE) is used to construct the surrogate surface from the respective input ranges of the hyperparameters. A rule of thumb instructs that the number of samples for an effective DoE must be ten times the size of the input vector dimension. Thus for six hyperparameters, this means 60 design points where each resulting point is a neural network.

2. Surrogate construction: Once the DoE is complete and initial sampling is done, the surrogate is constructed based on the value of the chosen objective function (validation loss) at each point. The approximation of the objective function is made using a universal Kriging with a Gaussian kernel of exponent order two.

3. Infill through Expected Improvement (EI): The surrogate model is refined by infill criteria in the locations where there is scope to improve. A large expected improvement (EI) is found in the regions where a better solution than the current best is possible, or the surrogate model has a large error at the location of consideration. The location of maximum improvement in the surrogate is found using differential evolution, and the population size is set to be five times the size of the input vector dimension.

4. Trust region method: In this method, the search is centred near the current best solution with a pre-defined trust-region length. The new optimum is found again using a differential evolution algorithm but only in the trust region. If a point with a lower objective function value is found, the trust region is again centred at the new optimum. The surrogate is then updated with this newfound point. The population size is again set to be five times the size of the input vector dimension. Thus, this is a local exploitation method in the region of influence of the current minimum. This method is required in addition to the EI method. For a high-dimensional non-linear problem (transonic aerodynamic analysis, shock waves), the convergence of EI is not guaranteed, and trust-region infill criteria increase the chances of finding the global minimum.

During the optimization process, each neural network is run for 1500 epochs. Furthermore, additional early stopping criteria are implemented. The training stops if the validation loss value of $1E-06$ is reached. An early stopping monitor with the patience of 500 epochs and tolerance of $1E-05$ is set, which stops the training if the validation loss has not improved in the patience epochs over the tolerance level. After the optimization process, a set of hyperparameters is obtained, which is used to train the final neural network. The final hyperparameters depend on the input features and output data supplied to the optimizer and the input range of the hyperparameters provided. Therefore, the user can still approximately control the size of the desired neural network.

7.1.2 Training the final neural networks and integration with TAU

The neural networks used in this study were trained on flow features generated from two datasets:

- Re = 9 million dataset: Training data generated from inversion solution at M = 0.7209, AoA = 5.669, Re = 8787960. This data is generated for a first test of the current FIML implementation and to assess whether data from a single flow case is enough for a turbulence model augmentation. The features for this database are seen in Equation 6.12. The distribution of the target variable and correlation of the input features can be seen in Figure 6.3. The distribution of the input features can be seen in Figure C.3.

- Full dataset: Training data generated from all inversion solutions detailed in Table 5.2 collated in one database. This data is the final database for the current FIML implementation. The features for this database are seen in Equation 6.13. The distribution of the target variable and correlation of the input features can be seen in Figure 6.6. The distribution of the input features can be seen in Figure C.6.

The optimization routine is run for both these datasets, and the resulting neural network architecture can be seen in Table 7.1. The datasets are standardized around

a zero mean and standard deviation of one before the optimization run, as this kind of data is more amenable to neural networks. The convergence information for the optimization routine was collected but is not reported here.

Hyperparameter	Re = 9 million dataset		Full dataset	
	Input bound	Optimal value	Input bound	Optimal value
Neurons in first hidden layer	(10,50)	31	(10,100)	86
Shrinkage factor	(0.1,1)	0.88	(0.1,1)	0.78
Number of hidden layers	(1,10)	4	(2,6)	4
Initial learning rate exponent (l_0)	(1,5)	2.5332453	-	2.5332453
Learning rate decay exponent (K_l)	(1,7)	2.55625	-	2.55625
Batch size	(4,300)	28	-	28

Table 7.1: Input bounds for the hyperparameter optimization and its result. Values are displayed for the two datasets considered for this study. The input bounds are selected with a prior idea of the expected neural network size and with trial and error. Only three design variables are optimized for the Full dataset to reduce the time to run the optimization code.

Only three design variables were optimized for the Full dataset instead of the Full input vector of size six. This reduction was made to reduce the time elapsed for the optimization process as the number of sampling points was lower at each optimization step, for e.g. 10*3 DoE points instead of 10*6. This was necessary because training the neural network at each step with a larger dataset (Full dataset) took more time.

Initially, the input bounds for both datasets were kept the same. However, for the Full dataset, this resulted in a smaller neural network (neurons in first hidden layer = 45, shrinkage factor = 0.74, hidden layers = 4) that did not perform well as the final chosen neural network during prediction time. On simply increasing the bound of neurons in the first hidden layer to (10,100), the resulting neural network was too large (neurons in first hidden layer = 71, shrinkage factor = 1.0, hidden layers = 10) with a high number of trainable parameters. Therefore, the bounds of the number of hidden layers were decreased to (2,6) to achieve the final neural network architecture for the Full dataset. The discussion here is supported in the results in Table 7.2.

NN architecture	Trainable parameters	Training time (min)	MSE loss: Test data	R^2: Test data	R^2: Full field
50,50,50,50	8,001	15.607	1.50652e-04	0.9790	-3.324
86,67,52,41	12,096	12.575	1.79340e-04	0.9750	-0.474
45,33,24,18	3,073	15.395	1.80762e-04	0.9748	-1.593
71,71,...,(10 times)	46,506	18.419	1.90110e-04	0.9735	-0.259

Table 7.2: Performance comparison of various neural network architectures for the Full dataset. The features used for training these neural networks were: $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$, H_{12}, f_d. The last column was a test done using features from the inversion solution at flow conditions M = 0.7421, AoA = 4.456, Re = 6360970 to predict on the whole computational grid. This test is the best reflection of the performance as when the neural network is plugged to TAU flow solver it has to predict on the whole grid.

The final neural network architecture and hyperparameters obtained can be used directly to make predictions. However, the training was repeated again with the 5000 training epoch and patience of 1000 epochs for early stopping using the validation loss. The tolerance for early stopping and minimum value for validation loss is kept the same. The training-test split was set as 80-20, and the training data was further split into training-validation set in an 80-20 ratio. The statistics of the num-

ber of data points in each split are reported in Table 7.3. The training is done using the 'Adam' optimizer [Kingma and Ba, 2014] with the exponential decay learning rate obtained from hyperparameter optimization. The loss function for the training is the MSE of the β prediction. The weights of the neural network are restored to the epoch with minimum validation loss and not the last epoch. The features used for the training will be explained in detail along with the results section in this chapter.

Data points	Re = 9 million dataset	Full dataset
Rows input database	157116	782980
All "Non-ones" rows	1275	6149
Randomly chosen "Ones"	255	1229
Total rows training dataset	1530	7378
Training rows (80%)	1224	5902
Test rows (20%)	306	1476

Table 7.3: Details regarding the number of data points in each dataset used in this study.

The trained neural network is then ultimately plugged into the TAU flow solver. The neural network augmentation of the turbulence model is tested on wind tunnel data at unseen flow conditions. The following steps are undertaken:

1. For a chosen testing flow case, the pETW data and converged TAU solution with the baseline SA-neg model is collected.

2. The TAU solution is restarted from the converged state, with the trained neural network in the loop. This connection was achieved using the DLR-SMARTy framework's capability of calling the TAU flow solver calculation in Python code.

3. The trained neural network predicts the $\beta(x)$ field based on the input flow features processed from the restarted TAU solution. This prediction is made after a particular frequency of TAU flow iterations, initially chosen as 500 flow iterations.

4. At the chosen iteration frequency, the flow data from TAU is processed into the selected input features for the neural network. These features are then standardized (zero mean, a standard deviation of one) using the mean and standard deviation of data at training time. This is done to maintain uniformity with the generation of the features at training time.

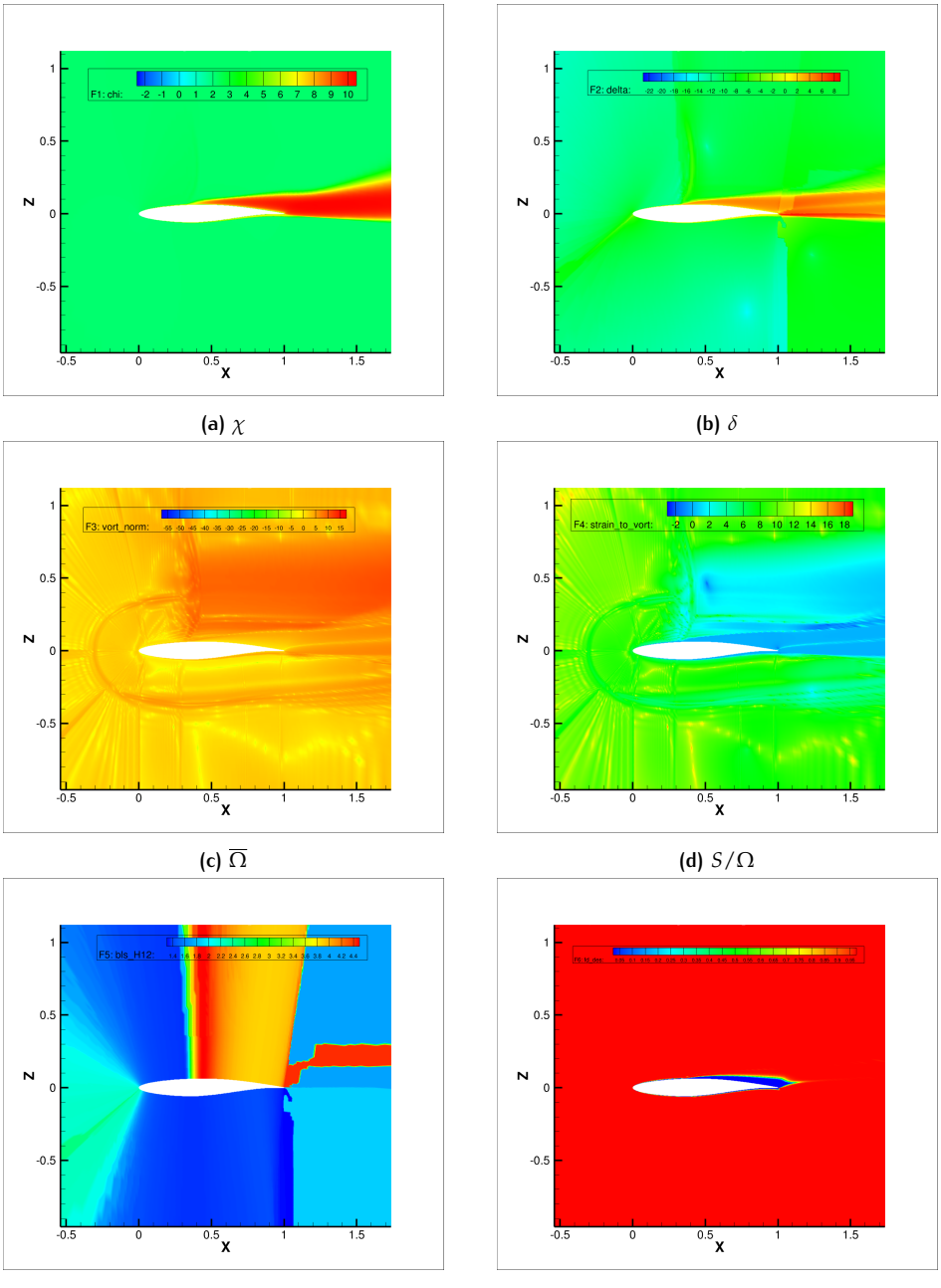
5. On processing the input features, the values at some grid points may be NaNs, $+\infty$ or $-\infty$. These values are replaced as: a) NaNs $\longrightarrow 0$, b) $+\infty \longrightarrow \bar{x} + 3\sigma$, and c)$-\infty \longrightarrow \bar{x} - 3\sigma$, where $\bar{x}$ is the mean and σ is the standard deviation of the feature at training time. The replacements for $+\infty$ and $-\infty$ are based on the extreme ends of the normal distribution curve. This replacement was deemed better than supplying very large finite values to the neural network, giving no meaningful predictions.

6. The processed feature data is finally passed to the neural network to predict the $\beta(x)$ field at that stage. The new $\beta(x)$ augments the flow data, starting from the production term in the transport equation. The flow is then run as usual to converge to the augmented solution.

7. The steps above are repeated at the chosen iteration frequency of plugging the neural network until the flow simulation converges. The convergence criteria are set based on the density residual reducing to $1E-07$ of its original value. If the simulation does not converge, the iteration frequency of NN predictions is increased, which was enough for most calculations in this study.

7.2 RESULTS FOR NN-AUGMENTED SA-NEG MODEL

Using the procedure mentioned in the previous section, the augmented SA-neg model is tested on unseen flow conditions from the pETW database. This section provides details behind the choice of features for training the neural networks and their predictions on seen and unseen flow conditions at training time. This section is divided into two subsections, with the first one providing results for Re = 9 million dataset followed by the results for the Full dataset.



(a) χ (b) δ

(c) $\overline{\Omega}$ (d) S/Ω

(e) H_{12} (f) f_d

Figure 7.3: Contours of the input features on the comuptational grid. These input features are formulated using the inversion solution flow variables at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960. The Re = 9 million dataset is generated by filtering this data according to target variable $\beta(x)$

7.2.1 Re = 9 million dataset results

The Re = 9 million dataset acts as the first database used for the current implementation. The database serves a dual purpose; acting as a test database for the first FIML implementation in the current study and providing insight into the amount of data required to implement a general model. In previous sections of this chapter, all relevant design decisions regarding the neural networks were mentioned apart from the features used for training. Equation 6.12 provides the final set of features after the feature engineering pipeline to be used for this dataset. However, there is a potential that even fewer features are required to achieve a successful ML model. Testing all subset combinations is time-consuming, but subsets chosen using intelligent criteria may provide some unexpected gains in terms of data needed or model performance.

The tools at the author's disposal to make this choice are the SFS scores, the correlation of input features with the target variable, and the input features on the computational grid. Additionally, shortlisting the features based on visual correlation of input features with the desired outcome (inversion $\beta(x)$ solution) can be a viable strategy based on the idea that this correlation may ease function approximation. Figure 7.3 shows the contours of input features on the computational grid, and the expected outcome can be seen in Figure 5.16. Therefore, the following features are selected to be used for training the neural networks:

- Training 1 - $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$, H_{12}, f_d: All features from Equation 6.12
- Training 2 - $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$: Top 4 features only considering SFS scores provided in Table 6.3
- Training 3 - $\ln(\chi)$, $\ln(\delta)$, H_{12}, f_d: Features selected using visual correlation from contours in Figure 7.3.
- Training 4 - $\ln(\chi)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$: Top 3 features only considering SFS scores provided in Table 6.3
- Training 5 - $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$: Selected on the basis of SFS score and correlation. (Table 6.3)

The neural network architecture and hyperparameters for these training were previously mentioned in Table 7.1. The time for training and their performance on the segregated test data can be viewed in Table 7.4. Figure 7.4 shows the evolution of validation MSE loss with the training epochs. The time to train is approximately the same for all feature subsets, but the performance on test data is affected by the number of features supplied. Furthermore, there are performance gains for certain subsets. Training 1 and Training 2 have approximately the same performance on the test data, with Training 2 working with two fewer features. The validation loss evolution for these NNs is the best among the cases considered with similar minimum validation loss values.

Features for training	**Training time (min)**	**Test data results**		
		R^2	MSE	MAE
$\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$, H_{12}, f_d	2.139	0.9934	4.77805e-05	3.18223e-03
$\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$	2.060	0.9931	4.96044e-05	2.94032e-03
$\ln(\chi)$, $\ln(\delta)$, H_{12}, f_d	1.967	0.9856	1.04270e-04	4.91505e-03
$\ln(\chi)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$	1.910	0.8216	1.29143e-03	1.08401e-02
$\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$	2.042	0.8707	9.36220e-04	1.58657e-02

Table 7.4: Information about NN training with various feature subsets for the Re = 9 million dataset. MSE = Mean Sqaure Error, MAE = Mean Absolute Error. Test data is the data split before starting the training, containing 306 data points.

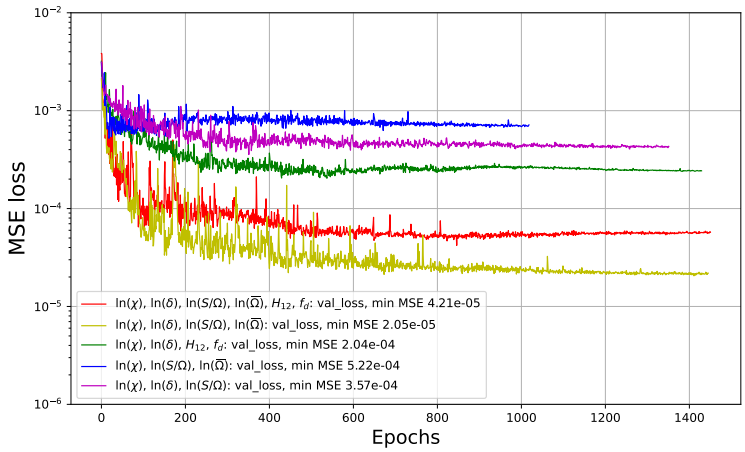


Figure 7.4: Validation losses for neural network training using different features. The feature data is processed from the inversion solution at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960. Each line corresponds to one training, and the minimum validation MSE loss of the training is reported.

More observations can be drawn from Table 7.4 and Figure 7.4 but the final performance of the NN-augmented SA-neg model in TAU solver is the most important in this study. These trained neural networks were tested first on the training data flow case to evaluate whether it worked and then unseen flow conditions. The results of this testing can be viewed in Section D.1. In this chapter, only relevant results supporting the discussion are displayed, and the reader is encouraged to view the Section D.1 for a better understanding. Conclusions on which input feature subset performs the best will be held until after the results are displayed. The testing is done on the following flow conditions:

- Testing on training data flow cases: To test whether the NN-augmented TAU works.
  1. M = 0.7209, AoA = 5.669, Re = 8787960
- Testing on unknown flow cases: To test the generalisation capability of the NN-augmented SA-neg model.
  1. M = 0.7421, AoA = 4.456, Re = 6360970
  2. M = 0.7235, AoA = 5.145, Re = 15323800
  3. M = 0.7173, AoA = 2.604, Re = 2680960

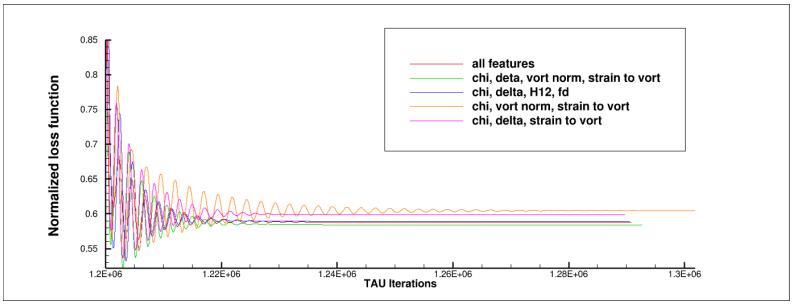
All the unknown flow cases for this dataset have an inversion solution available to compare the results of NN-augmented predictions. The details of the inversion solution used for comparison are provided in Table 5.2. The frequency of plugging the NN predicted discrepancy field is 500 flow iterations unless stated otherwise.

Testing on training data flow cases

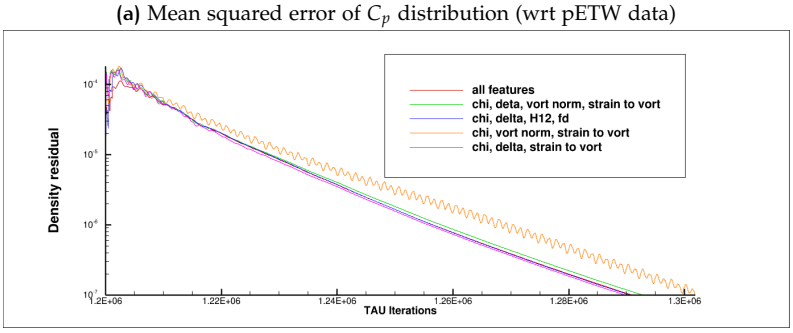
Figure 7.5 shows the residual evolution of the NN-augmented TAU solution starting from the baseline SA-neg solution. Oscillations can be observed in these plots, which are caused mainly due to a new discrepancy field being predicted during the flow calculation at every 500 iterations. The MSE of the augmented model solution with respect to the pETW results decreases, indicating an improvement caused by this augmentation.

Figure 7.6 shows that all feature subsets give an improved estimation of the shock location compared to the baseline turbulence model. The results vary slightly for the different sets of features. Beyond the shock location, the augmentation does not affect the predictions. This behaviour is expected as the inversion solutions were

chosen not to capture the post-shock flow phenomena. In particular, Training 4 and 5 slightly underpredict the pressure at the shock location compared to Training 1,2,3. Figure 7.7 shows the $\beta(x)$ distribution, focused on the shock foot. The predicted fields are compared to the inversion solution used for training. From a first view, it is seen that Training 1 (Figure 7.7b), Training 2 (Figure 7.7c) and Training 3 (Figure 7.7d) identify the production decrease well, compared to inversion solution.



(a) Mean squared error of C_p distribution (wrt pETW data)



(b) Density residual

Figure 7.5: Residual variation with the flow solution in TAU after augmenting the baseline SA-neg model with the trained neural network at every 500 iterations. The neural networks on the plot are trained using feature data from the inversion solution at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960. The NN-augmented TAU run is also at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960

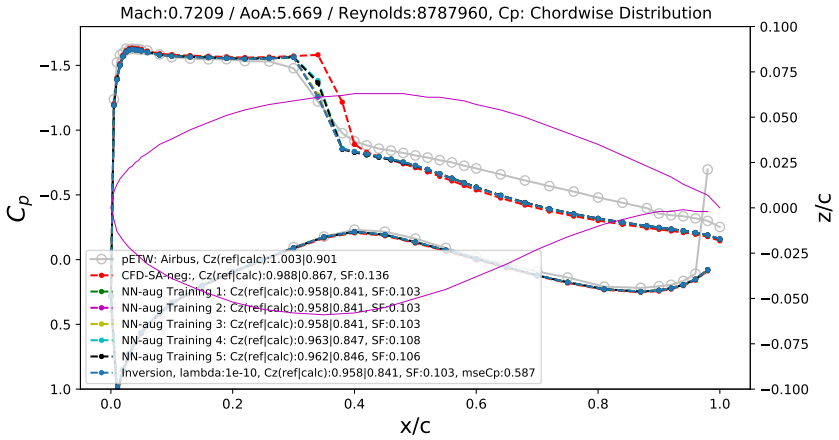
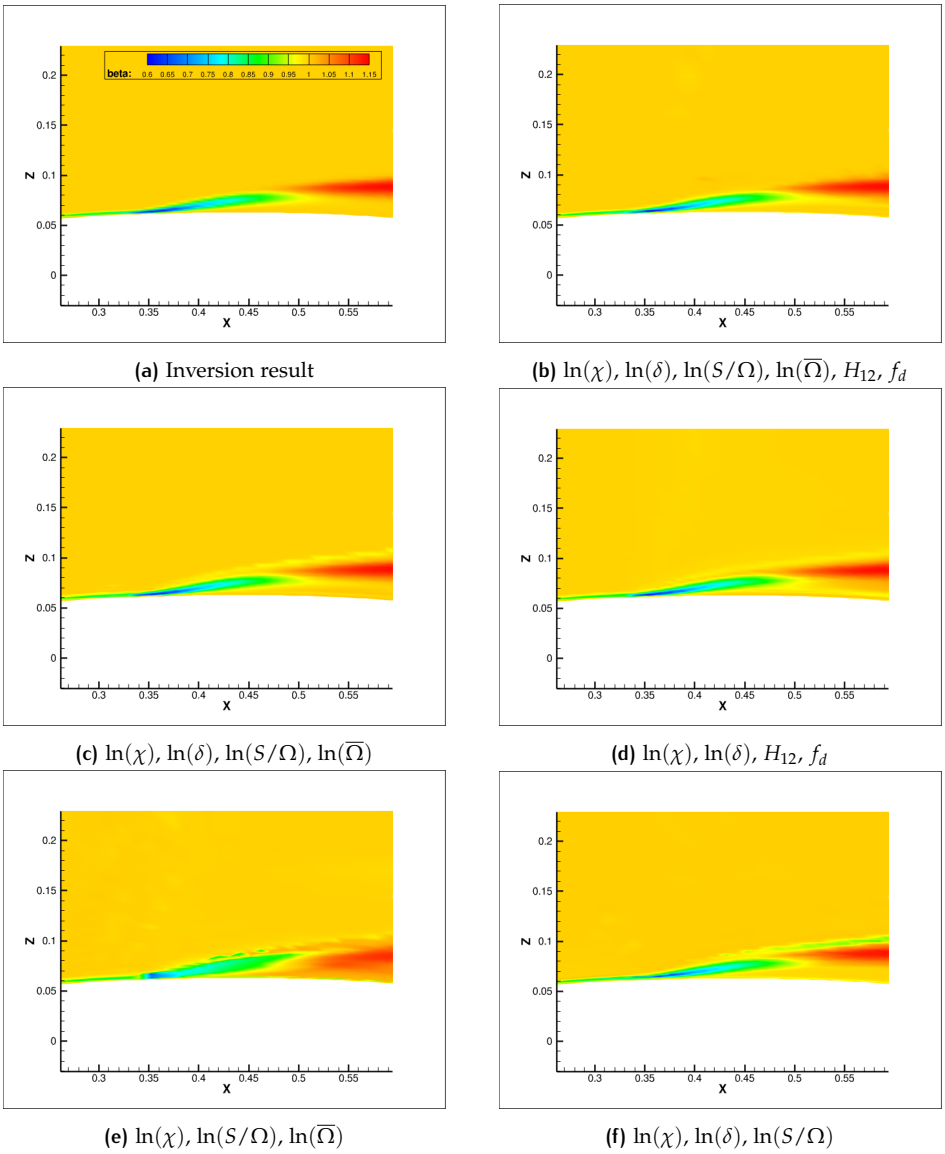


Figure 7.6: C_p distribution for the NN-augmented TAU solutions at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960. 'ref'= reference output from TAU solver, 'calc' = calculated output from a user-defined function, 'SF' = similarity factor with respect to wind tunnel results, 'mseCP' = normalized loss function value for the inversion problem. Training 1-5 are indicated in Section 7.2.1.



(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$, H_{12}, f_d

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, H_{12}, f_d

(e) $\ln(\chi)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$

(f) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$

Figure 7.7: Results after connecting the trained neural networks with different features to TAU flow solver. Focus on the shock foot to identify the area of production decrease. Testing on the flow conditions $M = 0.7209$, $AoA = 5.669$, $Re = 8787960$

The choice of features influences the output field; for instance, Training 3 does not use the features $\ln(S/\Omega)$ and $\ln(\overline{\Omega})$, which showed some oscillations in their input feature contours seen in Figure 7.3. Thus, these oscillations are not seen in Figure 7.7d, unlike all other cases. Based on these results, it was decided that Training 4 and Training 5 will not be used to test on unseen flow cases as this feature combination does not perform as well as the other three. The author believes that at least four features are needed for the current dataset as the database is too small. This is reflected in the predictions seen in Figure 7.7e and Figure 7.7f. Furthermore, the Table 7.4 and Figure 7.4 also suggested that using three features was not good enough in terms of the MSE on test data. Thus, in the next section, only results for Training 1,2 and 3 are presented.

Testing on unknown flow cases

The first test flow case is at flow conditions $M = 0.7421$, $AoA = 4.456$, $Re = 6360970$, where it is expected that shock-induced separation will occur. Here, the Reynolds number is less than the flow case used for the training data; thus, it is a good test case to see the extrapolation capability. Figure 7.8 shows that Training 1-3 perform

very similarly on this test case. Figure 7.9 shows the pressure distribution results, and the predicted pressure agrees very well with the inversion solution (which was not used for these neural networks) and identifies the shock location better than the baseline SA-neg results. Therefore, using the full feature shortlist of six features is unnecessary for this flow case as even well-chosen four feature subsets provide equivalent performance.

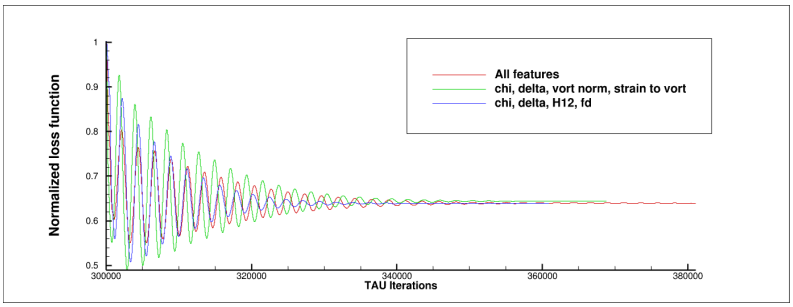


Figure 7.8: Mean squared error of C_p distribution (wrt pETW data). This residual variation with the flow solution can be viewed in TAU after augmenting the baseline SA-neg model with the trained neural network at every 500 iterations. The neural networks on the plot are trained using feature data from the inversion solution at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960. The NN-augmented TAU run is done at flow conditions M = 0.7421, AoA = 4.456, Re = 6360970

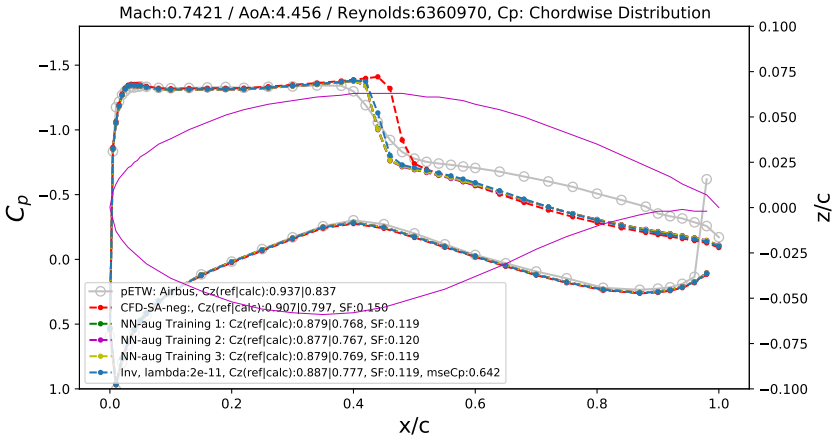
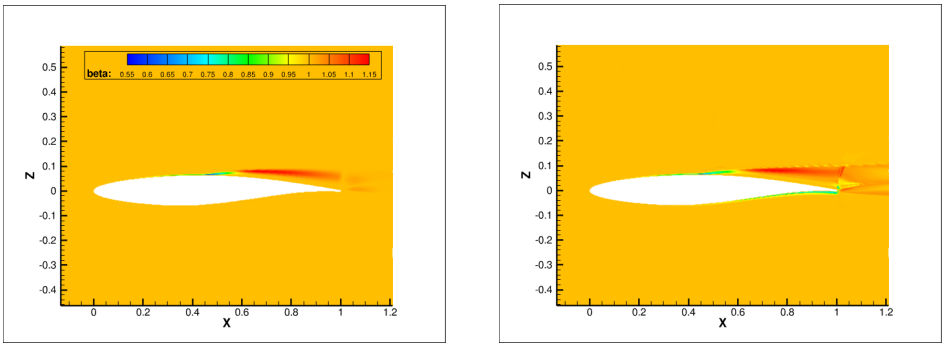


Figure 7.9: C_p distribution for the NN-augmented TAU solutions at flow conditions M = 0.7421, AoA = 4.456, Re = 6360970. 'ref'= reference output from TAU solver, 'calc' = calculated output from a user-defined function, 'SF' = similarity factor with respect to wind tunnel results, 'mseCP' = normalized loss function value for the inversion problem. Training 1-3 are indicated in Section 7.2.1.

Viewing the discrepancy fields in Figure 7.10 and Figure 7.11, the basic behaviour from the inversion solution is captured well by the augmented RANS results. On the lower surface, there is a slight production decrease near the trailing edge for all the results, which is incorrect compared to the inversion solution. The choice of features again influences the output field, with Figure 7.11c having the most oscillations and an extended production decrease region (green colour) compared to the inversion result. Figure 7.10b and Figure 7.10c have a sudden discontinuity at the trailing edge, which can be attributed to usage of the features H_{12} (Figure 7.3e) and f_d (Figure 7.3f).



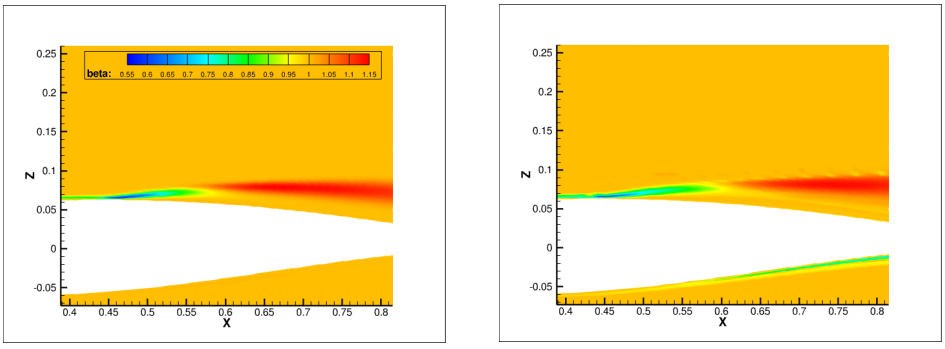
(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$, H_{12}, f_d

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, H_{12}, f_d

Figure 7.10: Results after connecting the trained neural networks with different features to TAU flow solver. Testing is done at the flow conditions $M = 0.7421$, $AoA = 4.456$, $Re = 6360970$



(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$, H_{12}, f_d

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, H_{12}, f_d

Figure 7.11: Results after connecting the trained neural networks with different features to TAU flow solver. Focus on the shock foot to identify the area of production decrease. Testing is done at flow conditions $M = 0.7421$, $AoA = 4.456$, $Re = 6360970$

The next testing flow case is $M = 0.7235$, $AoA = 5.145$, $Re = 15323800$, where again the shock-induced separation is expected. This time the extrapolation capability is evaluated at a higher Reynolds number. For this flow case, Figure 7.12 shows that Training 3 gives the closest prediction to the pETW data. This agreement shows that choosing the features using visual correlation is a viable feature selection strategy for this case. Figure 7.13 shows that the shock location prediction has improved from the baseline SA-neg result, and the neural network has learnt the intended behaviour of having a sudden pressure increase at the shock location. The displayed inversion solution fits more closely to the pETW data, but the current dataset is not aware of this behaviour. The $\beta(x)$ fields are not displayed here, but can be seen in Figure D.11 and Figure D.12.

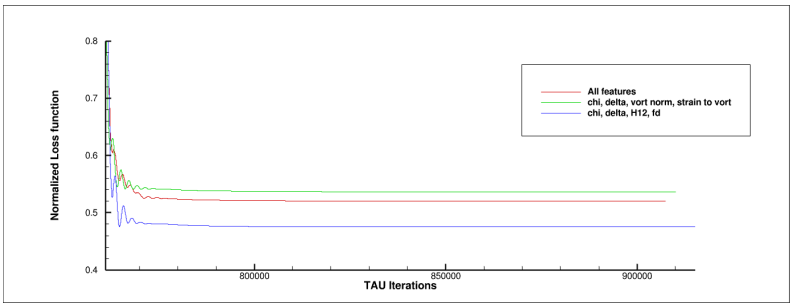


Figure 7.12: Mean squared error of C_p distribution (wrt pETW data). This residual variation with the flow solution can be viewed in TAU after augmenting the baseline SA-neg model with the trained neural network at every 500 iterations. The neural networks on the plot are trained using feature data from the inversion solution at flow conditions $M = 0.7209$, $AoA = 5.669$, $Re = 8787960$. The NN-augmented TAU run is done at flow conditions $M = M = 0.7235$, $AoA = 5.145$, $Re = 15323800$

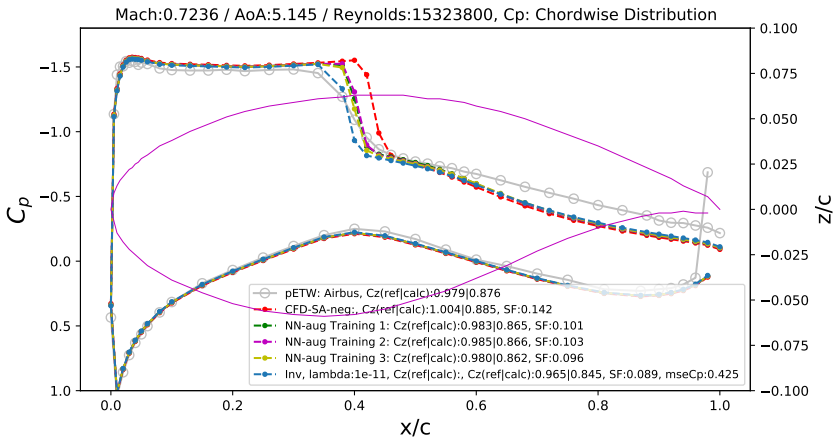


Figure 7.13: C_p distribution for the NN-augmented TAU solutions at flow conditions $M = 0.7235$, $AoA = 5.145$, $Re = 15323800$. 'ref'= reference output from TAU solver, 'calc' = calculated output from a user-defined function, 'SF' = similarity factor with respect to wind tunnel results, 'mseCP' = normalized loss function value for the inversion problem. Training 1-3 are indicated in Section 7.2.1.

The final testing case for this dataset is at the flow conditions $M = 0.7173$, $AoA = 2.604$, $Re = 2680960$. For this case, the shock is not strong enough to cause a shock-induced separation on the airfoil. Therefore, this flow case is different from the data the neural networks have been trained on such a dataset. Figure 7.14 shows that all the augmented solutions do not have a significant improvement in the resulting pressure distribution, with Training 3 performing the worst.

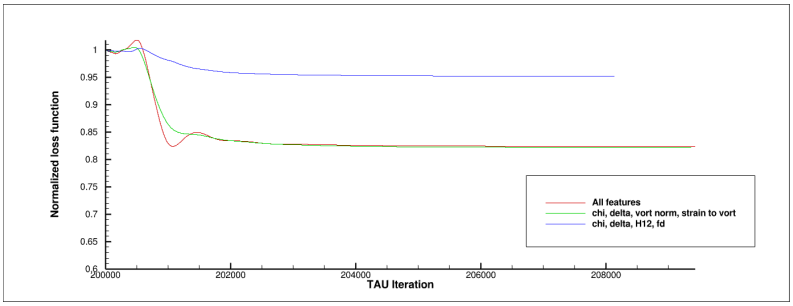


Figure 7.14: Mean squared error of C_p distribution (wrt pETW data). This residual variation with the flow solution can be viewed in TAU after augmenting the baseline SA-neg model with the trained neural network at every 500 iterations. The neural networks on the plot are trained using feature data from the inversion solution at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960. The NN-augmented TAU run is done at flow conditions M = 0.7173, AoA = 2.604, Re = 2680960

Figure 7.15 shows the pressure distribution results for the three augmentation cases. There are very marginal shifts of the NN augmented solutions near the shock location towards the pETW results. Thus, it may seem that the data from this flow case is not identified well by the trained neural networks. It is interesting to see whether the inversion solution for this flow case looks to see whether even the inversion procedure can reach the pETW results.

Figure 7.16 shows the inversion solution for the current testing flow case at various Tikhonov regularization values. Until a certain strength of regularization, the inversion solution does not show any movement towards the pETW solution. However, on decreasing the regularization further, the inversion solution suddenly overfits the pETW solution. For the current dataset used for neural network training, the Tikhonov regularization was chosen so that the shock location was correctly identified and overfitting was avoided. Thus, for the augmented RANS solution, the author expects the output to resemble more of the highly regularized case. Therefore, the augmented solution in Figure 7.15 still performs as expected, as it has been trained on a highly regularized dataset. The fact that the augmented solution has not gone worse and still resembles the baseline solution is encouraging as the testing flow phenomena is unseen.

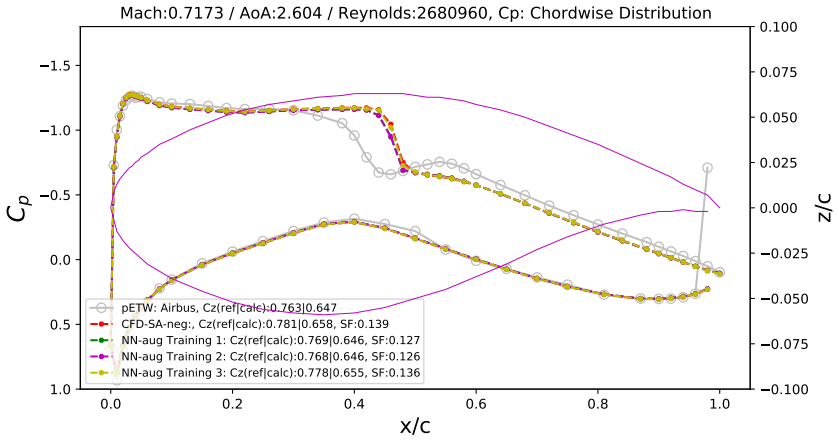


Figure 7.15: C_p distribution for the NN-augmented TAU solutions at flow conditions M = 0.7173, AoA = 2.604, Re = 2680960. 'ref'= reference output from TAU solver, 'calc' = calculated output from a user-defined function, 'SF' = similarity factor with respect to wind tunnel results, 'mseCP' = normalized loss function value for the inversion problem. Training 1-3 are indicated in Section 7.2.1.

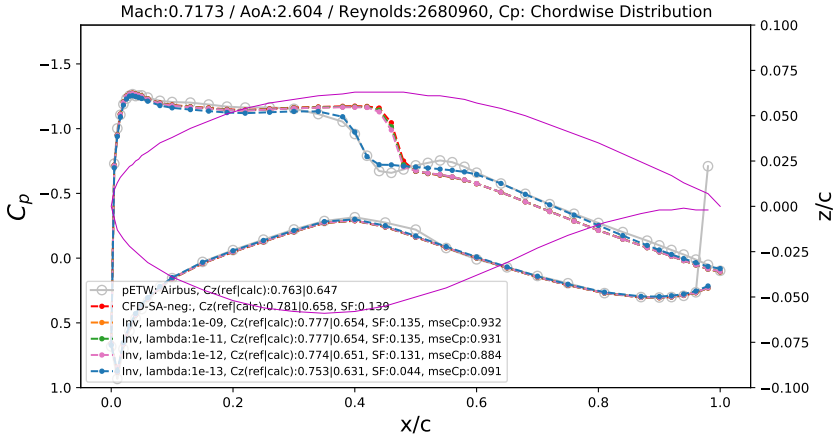
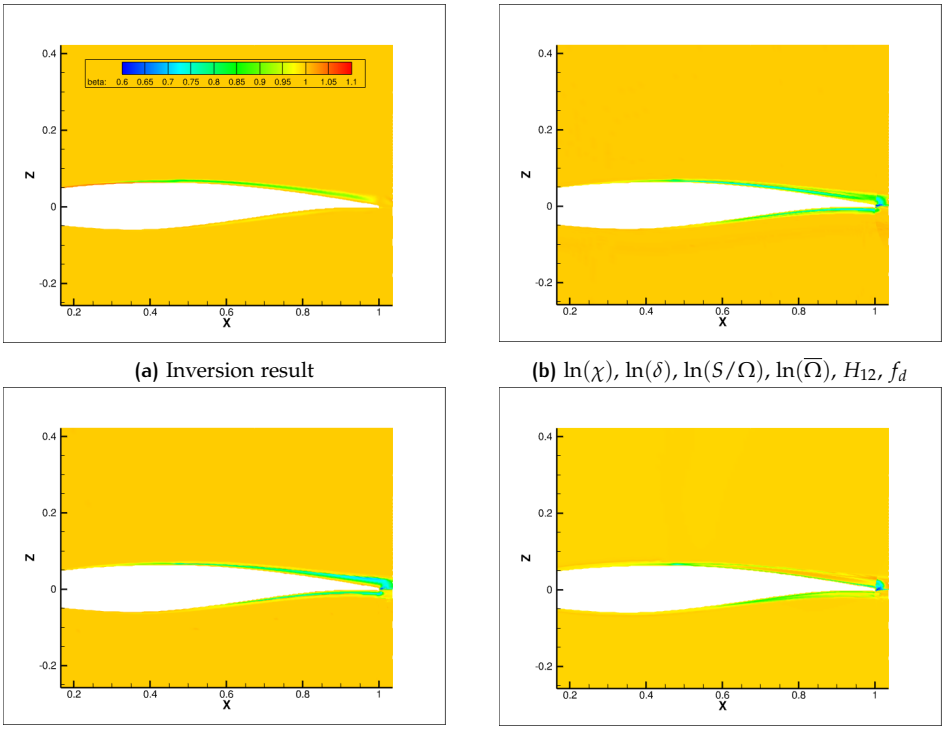


Figure 7.16: C_p distribution for inversion solutions at various λ values at flow conditions M = 0.7173, AoA = 2.604, Re = 2680960. 'ref'= reference output from TAU solver, 'calc' = calculated output from a user-defined function, 'SF' = similarity factor with respect to wind tunnel results, 'mseCP' = normalized loss function value for the inversion problem.

Figure 7.17 and Figure 7.18 show the output discrepancy fields for the augmented solutions. The comparison is done with a highly regularized inversion solution at $\lambda = 1E-12$ (pink line in Figure 7.16). The inversion solution itself does not have the recognizable shock foot production decrease, as seen in the inversion solution for other test cases. Furthermore, there are no regions of turbulence production increase on the upper surface of the airfoil, as was seen for flow cases with shock-induced separation. The augmented solution in Training 1 and Training 2 resemble the inversion solution in consideration, but Training 3 cannot identify the production decrease.



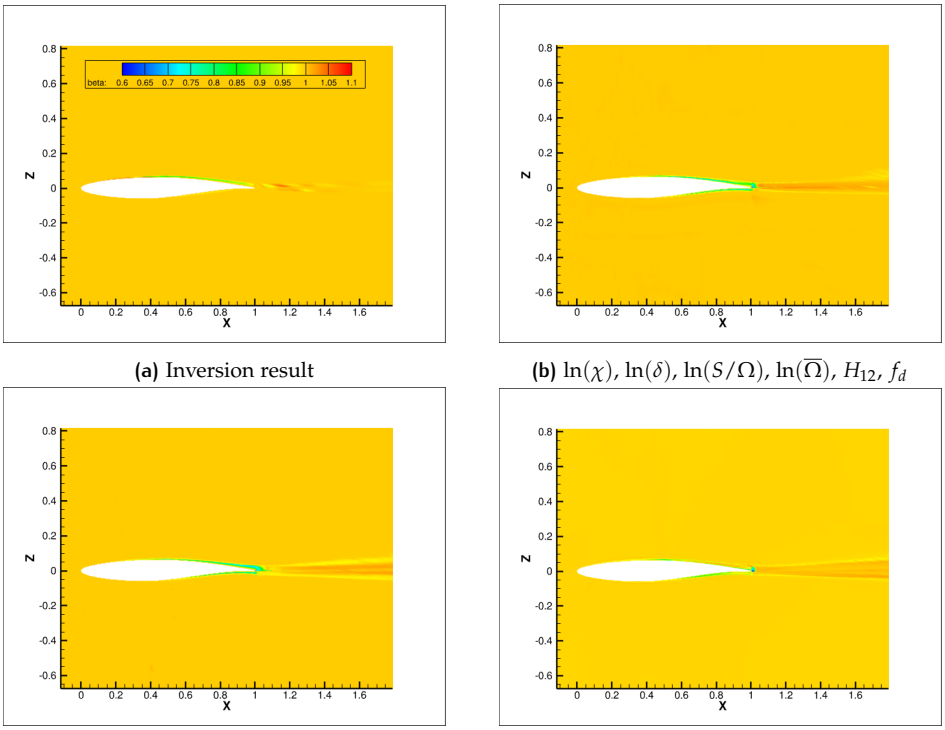
(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$, H_{12}, f_d

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, H_{12}, f_d

Figure 7.17: Results after connecting the trained neural networks with different features to TAU flow solver. Testing is done at flow conditions M = 0.7173, AoA = 2.604, Re = 2680960.



(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$, H_{12}, f_d

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, H_{12}, f_d

Figure 7.18: Far field results after connecting the trained neural networks with different features to TAU flow solver. Testing is done at flow conditions M = 0.7173, AoA = 2.604, Re = 2680960.

7.2.2 Full dataset results

This section displays the testing results for the Full dataset, which has been generated using inversion solutions at flow conditions in Table 5.1. The first expectation using a more expansive dataset is that the augmentations for this case will be more accurate and generalized. The neural network architecture and hyperparameters for the training were previously mentioned in Table 7.1. The feature subsets tested for this dataset are listed below:

- Training 1 - $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}: Using all the features from Equation 6.13, i.e. the final feature shortlist for this dataset.
- Training 2 - $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$: Selecting the top 3 features in Table 6.5.
- Training 3 - $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}: Adding one of the remaining features to the top 3 features subset.
- Training 4 - $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$: Adding the other remaining feature to the top 3 features subset.
- Training 5 - $\ln(\chi)$, $\ln(\delta)$, H_{12}: Testing a set of 3 input features based on visual correlation, avoiding oscillations seen in Figure 7.3c and Figure 7.3d.

Here, it is not easy to select features based on visual correlation as there are multiple flow cases in the dataset, and one would have to look at the contours of all the features for each flow case. However, given that all the flow cases correspond to the shock-induced separation phenomena, the decision for this dataset is based on contours in Figure 7.3. If the dataset were more expansive, incorporating more flow conditions or different geometries, this kind of decision making would not have been possible.

The time for training and their performance on the segregated test data can be viewed in Table 7.5, and the truth vs prediction scatter plots in Figure 7.20. Figure 7.19 shows the evolution of validation MSE loss with the training epochs.

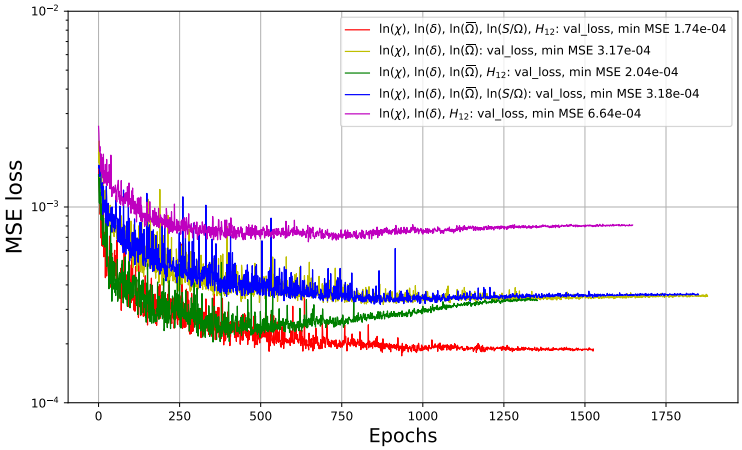
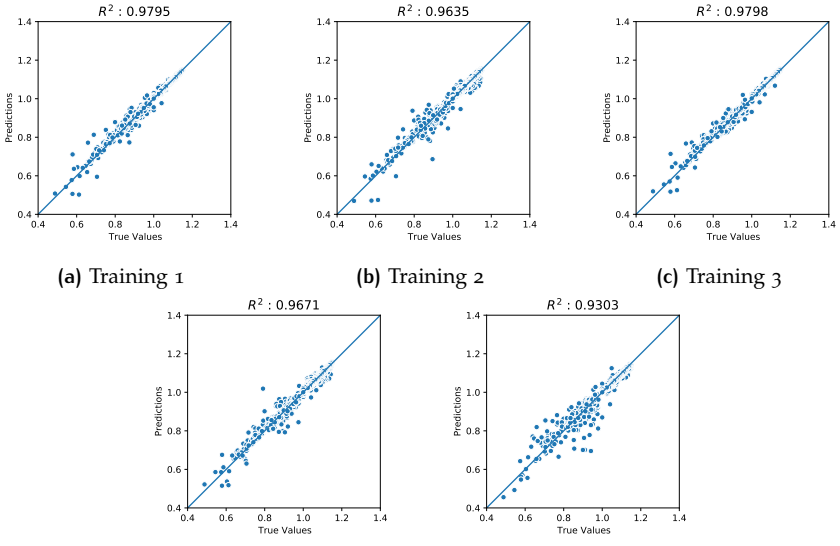


Figure 7.19: Validation losses for neural network training using different features. The feature data is processed from all the inversion solutions. Each line corresponds to one training, and the minimum validation MSE loss of the training is reported.

Features for training	Training time (min)	Test data results		
		R^2	MSE	MAE
$\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}	9.863	0.9795	1.57863e-04	5.11303e-03
$\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$	12.329	0.9635	2.81145e-04	7.35925e-03
$\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}	8.685	0.9798	1.56016e-04	6.05569e-03
$\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$	11.725	0.9671	2.53542e-04	7.09951e-03
$\ln(\chi)$, $\ln(\delta)$, H_{12}	10.470	0.9303	5.36785e-04	1.03795e-02

Table 7.5: Information about NN training with various feature subsets for the Full dataset. MSE = Mean Square Error, MAE = Mean Absolute Error. Test data is the data split before starting the training, containing 1476 data points.



(a) Training 1 **(b)** Training 2 **(c)** Training 3

(d) Training 4 **(e)** Training 5

Figure 7.20: Truth vs Predicted values plot for neural network training with different feature subsets. Training 1-5 are explained in Section 7.2.2

The time to train is approximately the same for all feature subsets, but the performance on test data is affected by the number of features supplied. Training 1 and 3 perform the best on the segregated test data and take the least time to train. The validation loss evolution for Training 3 gets worse as the training process, but the weights are restored to the best validation loss state. The minimum validation MSE for Training 3 is very close to Training 1, using all features. Another important fact is that Training 2 using only three features also performs comparatively well to the subsets containing four or five features.

The final performance of the NN-augmented SA-neg model in the TAU solver is tested first on the training data flow cases to evaluate whether it worked and then on unseen flow conditions. The results of this testing can be viewed in Section D.2. Again, this chapter shows only relevant results supporting the discussion. The testing is done on the following flow conditions:

- Testing on training data flow cases: To test whether the NN-augmented TAU works.
  1. $M = 0.7421$, $AoA = 4.456$, $Re = 6360970$
  2. $M = 0.7209$, $AoA = 5.669$, $Re = 8787960$
  3. $M = 0.7235$, $AoA = 5.145$, $Re = 15323800$

- Testing on unknown flow cases: To test the generalisation capability of the NN-augmented SA-neg model.
  1. $M = 0.7206$, $AoA = 5.737$, $Re = 6331480$
  2. $M = 0.7421$, $AoA = 4.420$, $Re = 8760680$
  3. $M = 0.7173$, $AoA = 2.604$, $Re = 2680960$
  4. $M = 0.7397$, $AoA = 1.403$, $Re = 13257500$
  5. $M = 0.7110$, $AoA = 5.145$, $Re = 15363000$

Figure 7.21 shows where the training and testing flow cases lie with respect to each other. Figure 7.22 shows the flow solution of the unknown flow cases using the baseline SA-neg model. In Figure 7.22a and Figure 7.22b, the shock is strong enough to cause separation near the shock foot. However, for the rest, the shock-induced separation is not seen.

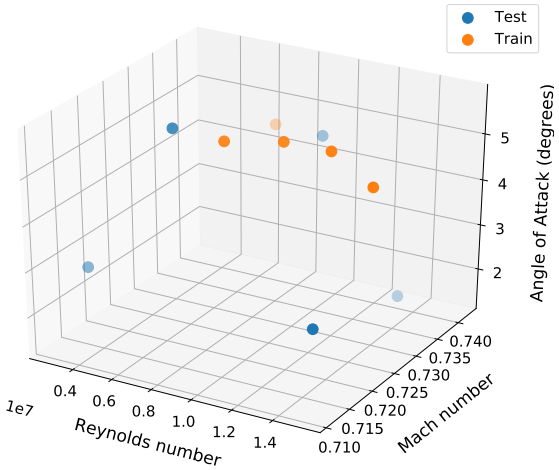
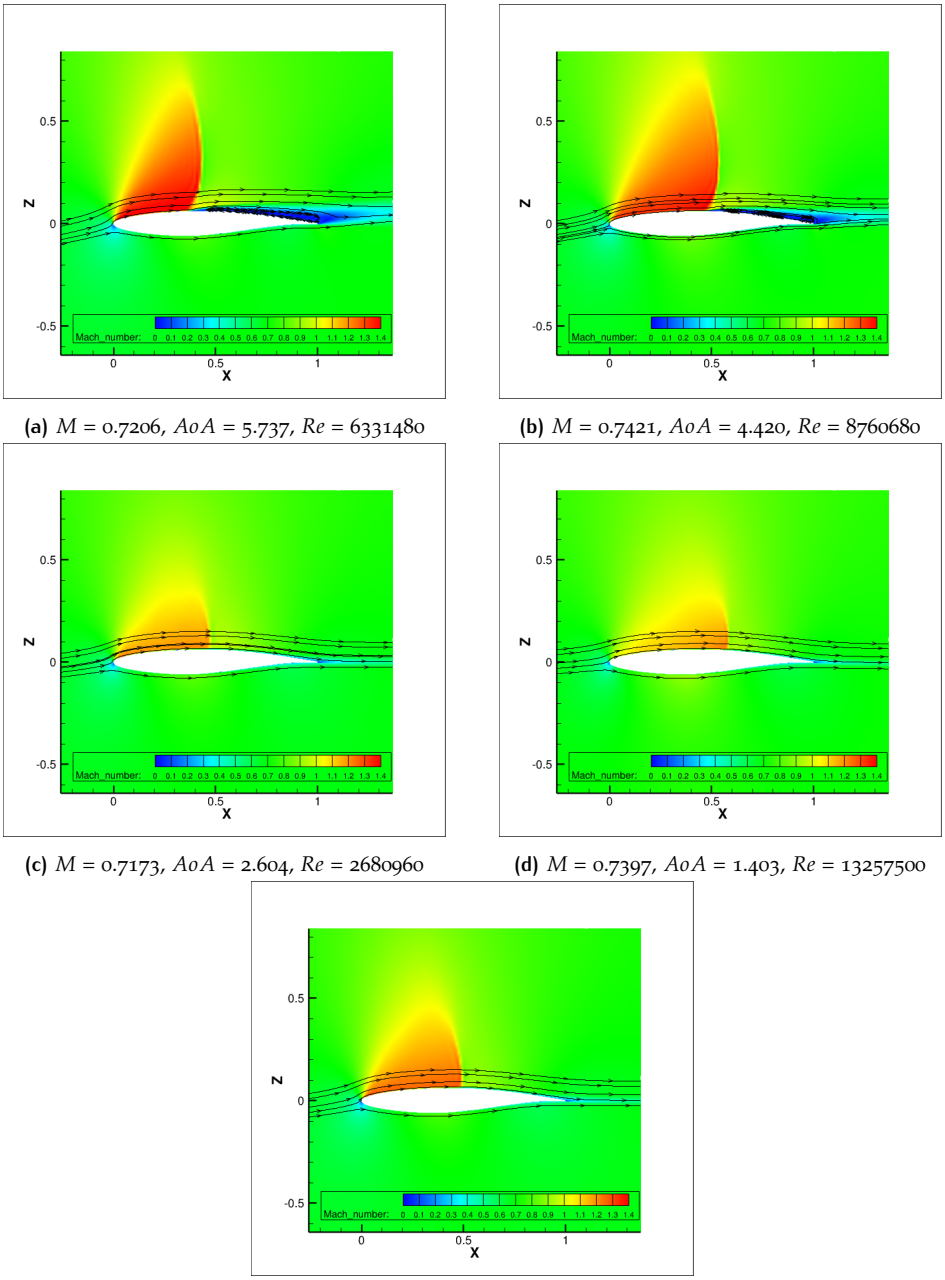


Figure 7.21: Scatter plot displaying flow conditions for the flow cases used for training the neural networks with Full dataset, and the testing flow conditions. Testing is done at unseen flow conditions to assess the extrapolation capability of the neural network augmentation.



(a) $M = 0.7206$, $AoA = 5.737$, $Re = 6331480$

(b) $M = 0.7421$, $AoA = 4.420$, $Re = 8760680$

(c) $M = 0.7173$, $AoA = 2.604$, $Re = 2680960$

(d) $M = 0.7397$, $AoA = 1.403$, $Re = 13257500$

(e) $M = 0.7110$, $AoA = 5.145$, $Re = 15363000$

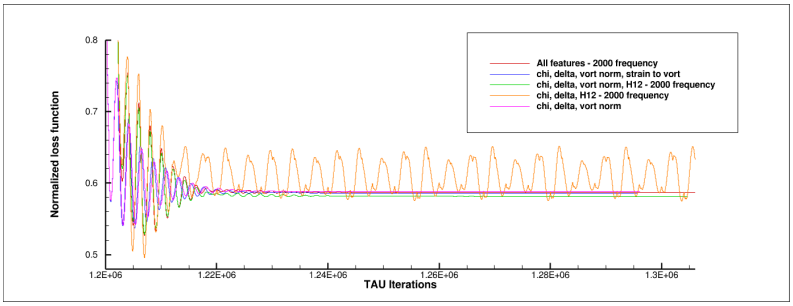
Figure 7.22: Mach number contours of the testing flow cases using the baseline SA-neg model in TAU solver.

Testing on training data flow cases

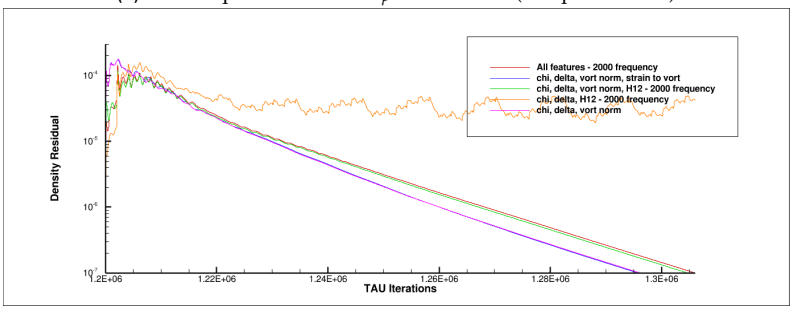
NN-augmented RANS is tested for three of the five flow cases whose flow inversion solution generated the training data. The results for all these cases can be found in Section D.2.1. Here, the discussion will follow using the results from $M = 0.7209$, $AoA = 5.669$, $Re = 8787960$ as all three cases showed similar behaviour in the testing results.

Figure 7.23 displays the residuals for the neural network augmentation for all the chosen feature subsets. Initially, the iteration frequency at which the discrepancy field is predicted was kept the same as before, i.e. 500. However, the simulation would not converge to the desired density residual criteria for some feature subsets and keep oscillating. To fix this, the augmented solution was rerun at an iteration

frequency of 2000 flow iterations for the neural network prediction. This change improved the convergence characteristics for most NN-augmented flow calculations, an example of which can be seen in Figure 7.24. However, for Training 5 with the features $\ln(\chi)$, $\ln(\delta)$, H_{12} convergence was still not achieved, but increase the frequency did provide some improvement which can be seen in Figure 7.25.

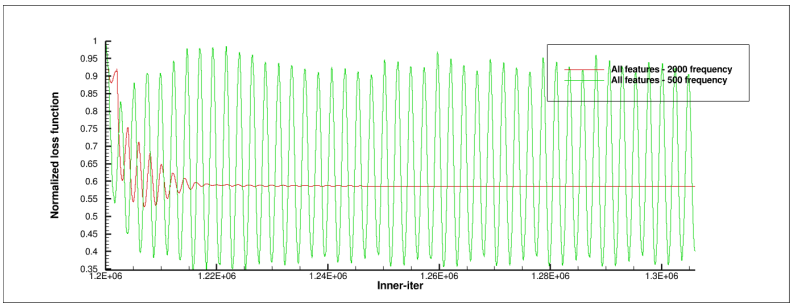


(a) Mean squared error of C_p distribution (wrt pETW data)

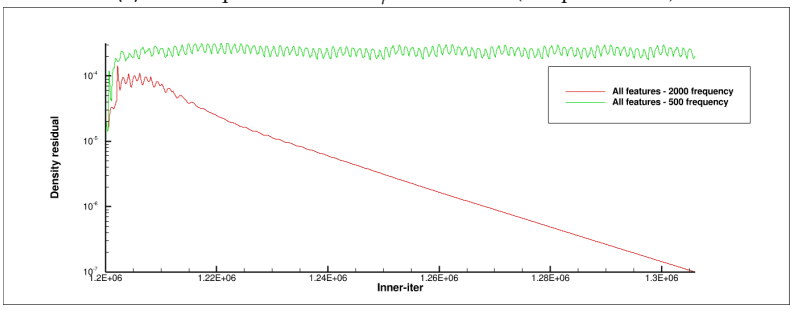


(b) Density residual

Figure 7.23: Residual variation with the flow solution in TAU after augmenting the baseline SA-neg model with the trained neural network. Testing is done at flow conditions $M = 0.7209$, $AoA = 5.669$, $Re = 8787960$.

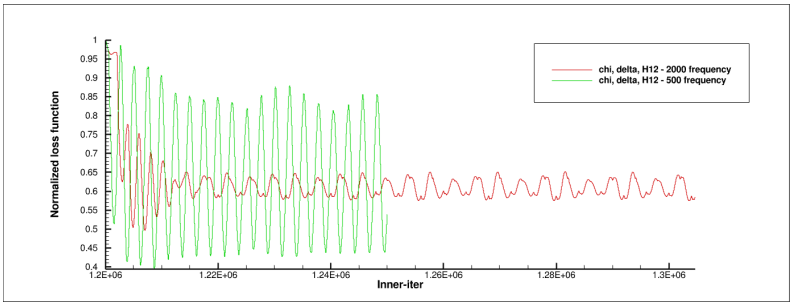


(a) Mean squared error of C_p distribution (wrt pETW data)

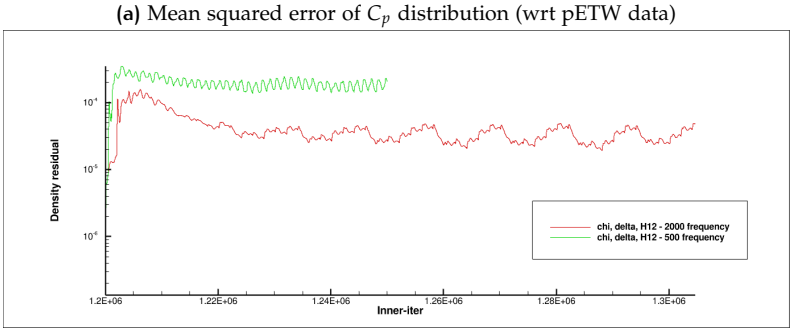


(b) Density residual

Figure 7.24: Residual comparison on changing the frequency of plugging the trained neural network. Features used: $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}. Testing is done at flow conditions $M = 0.7209$, $AoA = 5.669$, $Re = 8787960$.



(a) Mean squared error of C_p distribution (wrt pETW data)



(b) Density residual

Figure 7.25: Residual comparison on changing the frequency of plugging the trained neural network. Features used: $\ln(\chi)$, $\ln(\delta)$, H_{12}. Testing is done at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960.

All the feature subsets are able to identify the shock location well, as seen in Figure 7.26. Predictions using the selected feature subsets vary slightly from each other (Figure 7.27). The predicted $\beta(x)$ fields can be seen in Figure 7.28 and Figure 7.29. In Figure 7.28e, the magnitude of the oscillations in $\beta(x)$ values (green dots) is higher than other cases. This spurious behaviour in the discrepancy field prediction may be one of the reasons that the simulation does not converge at a prediction iteration frequency of 500 iterations. For Training 5 in Figure 7.28f, the shock foot predictions do not resemble the inversion solution as well as the other cases.

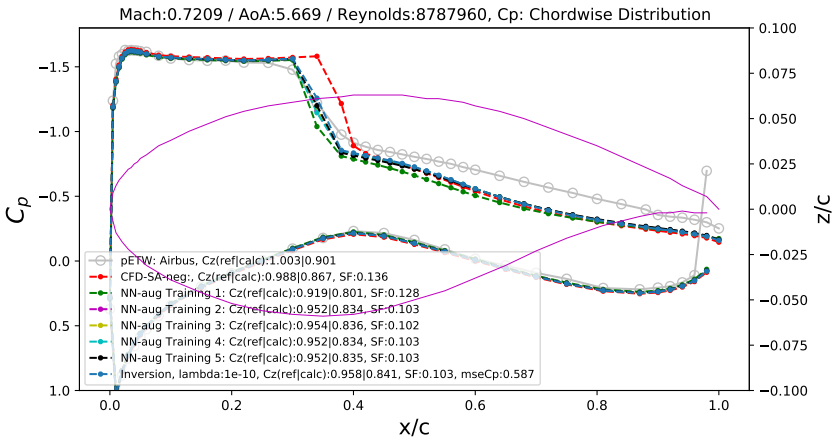
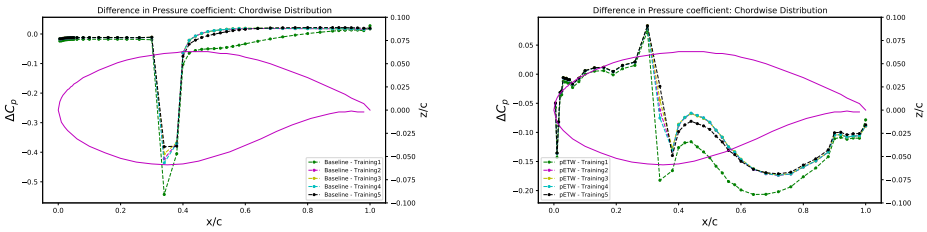


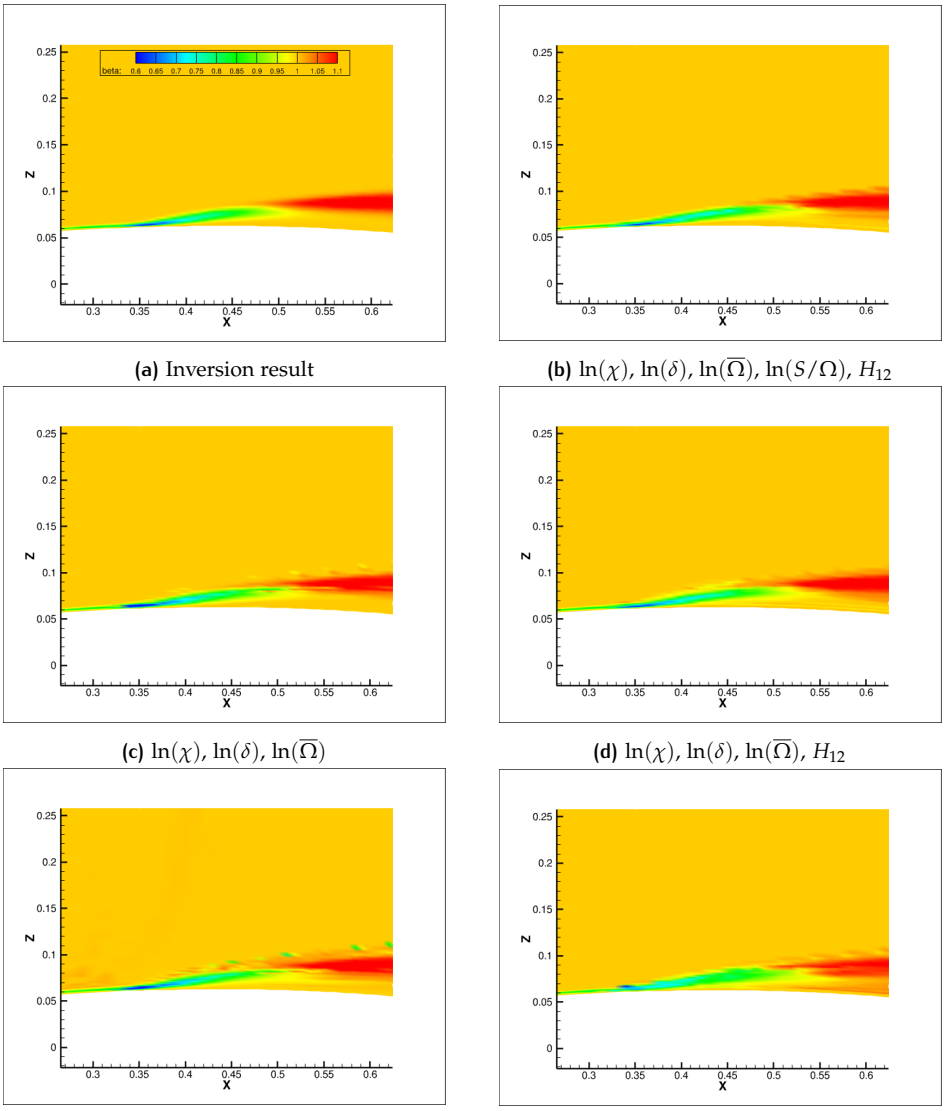
Figure 7.26: C_p distribution for the NN-augmented TAU solutions. Testing is done at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960. Training 1-5 are indicated in Section 7.2.2.



(a) Difference with Baseline solution

(b) Difference with pETW result

Figure 7.27: ΔC_p distribution for the NN-augmented TAU solution. Testing is done at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960. Training 1-5 are indicated in Section 7.2.2.



(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

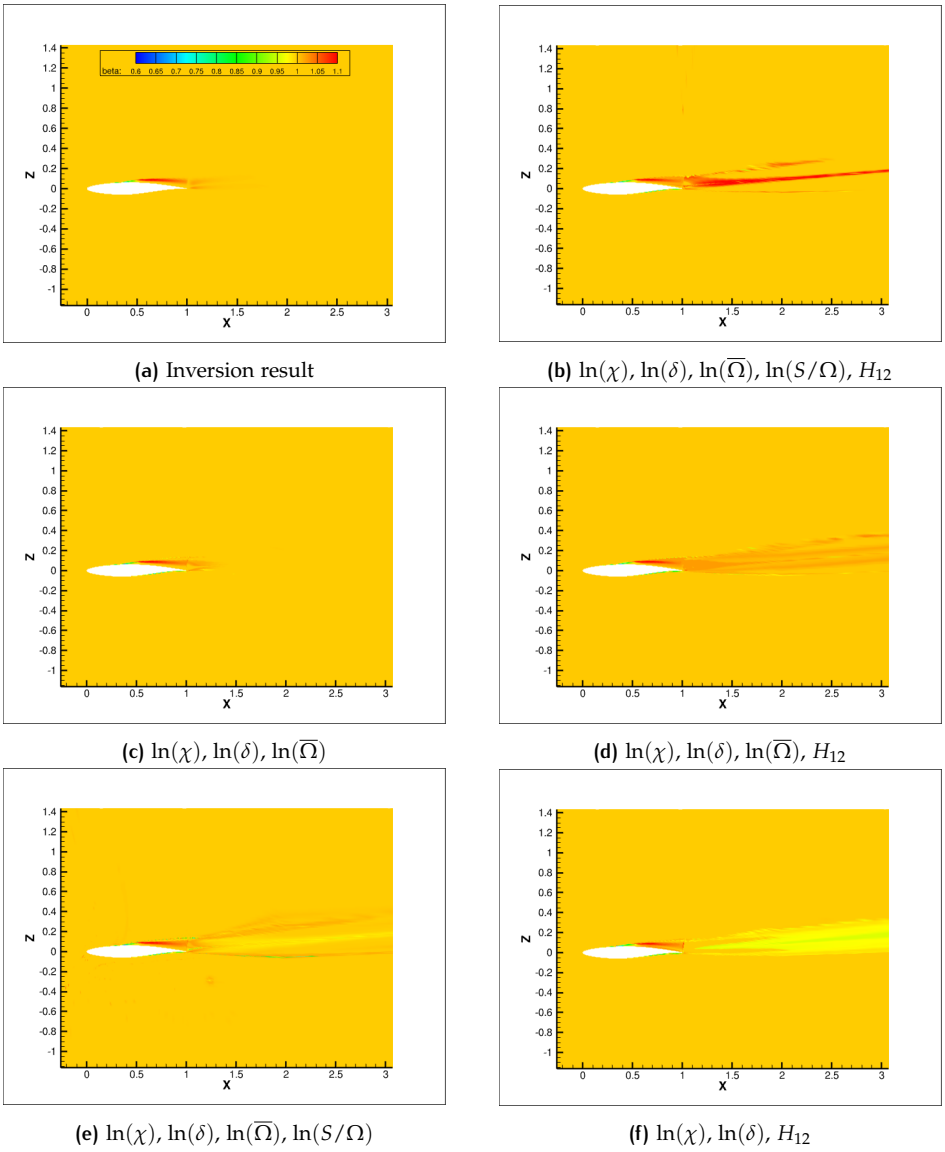
(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

(e) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

(f) $\ln(\chi)$, $\ln(\delta)$, H_{12}

Figure 7.28: Results at shock foot after connecting the trained neural networks with different features to TAU flow solver. Testing is done at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960.



(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

(e) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

(f) $\ln(\chi)$, $\ln(\delta)$, H_{12}

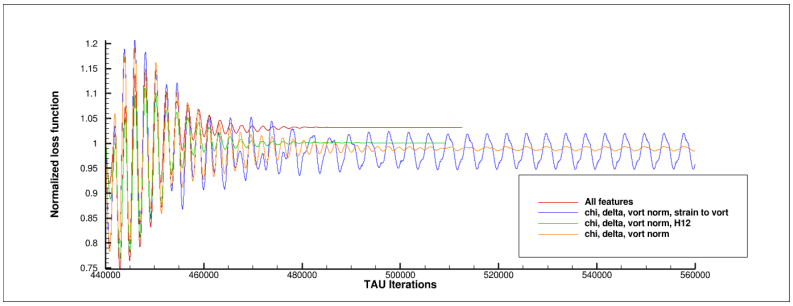
Figure 7.29: Results at shock foot after connecting the trained neural networks with different features to TAU flow solver. Testing is done at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960.

Another interesting behaviour is seen in Figure 7.29c, where the $\beta(x)$ prediction for Training 2 only changes values close to the airfoil, and $\beta = 1$ everywhere else, much like the inversion result in Figure 7.29a. This prediction is the first case where there are no predictions in the flow behind the trailing edge, as observed in all other subfigures of Figure 7.29. These unwanted trailing edge predictions do not affect the pressure coefficient distribution on the surface of the airfoil, but it is an added advantage if it is possible to avoid them, to match closely to the inversion solution.

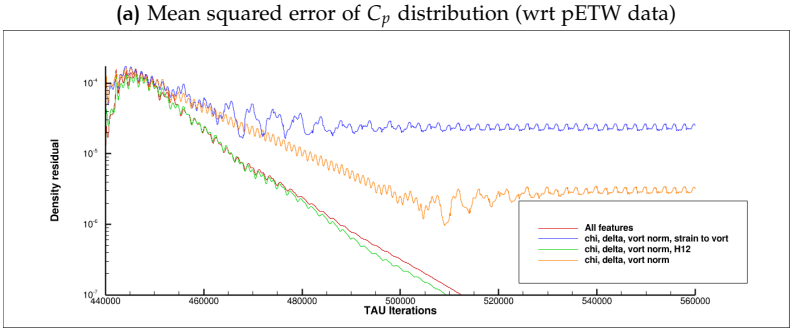
Testing on unknown flow cases

Neural networks trained on the Full dataset are now tested on the unknown flow cases. The results for all the cases can be viewed in Section D.2.2. Looking at the results of the Full dataset on the training flow case tests, it is decided that iteration frequency for plugging the neural network to TAU is increased to 2000 for all the cases in this section. Nothing further is done if the augmented calculation does not converge; however, a study about how this frequency affects the simulation would have been interesting. Inversion solutions are not available for all the test cases in this section; thus, the $\beta(x)$ results can not be compared to a reference field.

The first test case is for the flow conditions $M = 0.7206$, $AoA = 5.737$, $Re = 6331480$, where a shock-induced separation is expected to occur. Only Training 1 and Training 3 converge, as seen in Figure 7.30b. This fact is also confirmed in Figure 7.33b and Figure 7.33d, where for Training 2 and Training 4, the discrepancy field appears to be immature due to lack of convergence of the solution. The MSE error for pETW data is still high for all the feature subsets (Figure 7.30a), which may suggest that all the augmentations do not work for this flow case. However, the pressure distribution in Figure 7.31 suggests that there is an improvement in identifying the shock location for all the feature subsets from the baseline SA-neg result.



(a) Mean squared error of C_p distribution (wrt pETW data)



(b) Density residual

Figure 7.30: Residual variation with the flow solution in TAU after augmenting the baseline SA-neg model with the trained neural network after every 2000 iterations. Testing is done at flow conditions $M = 0.7206$, $AoA = 5.737$, $Re = 6331480$.

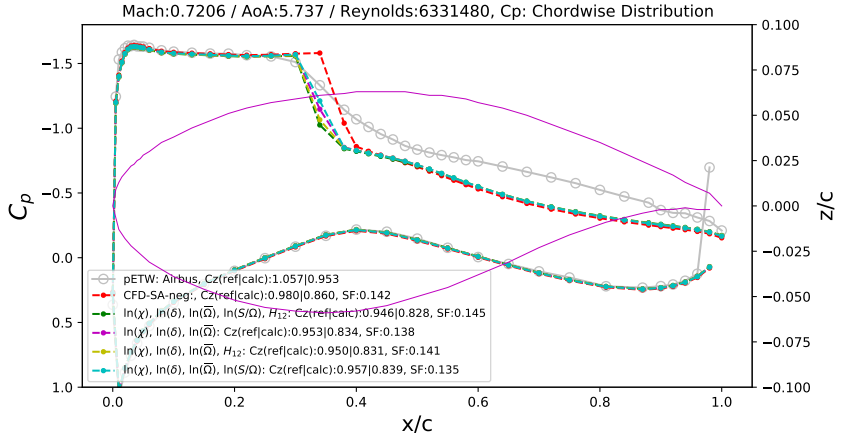
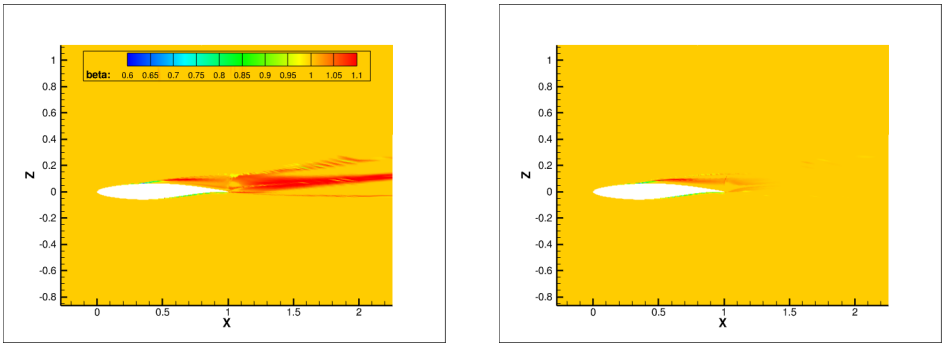
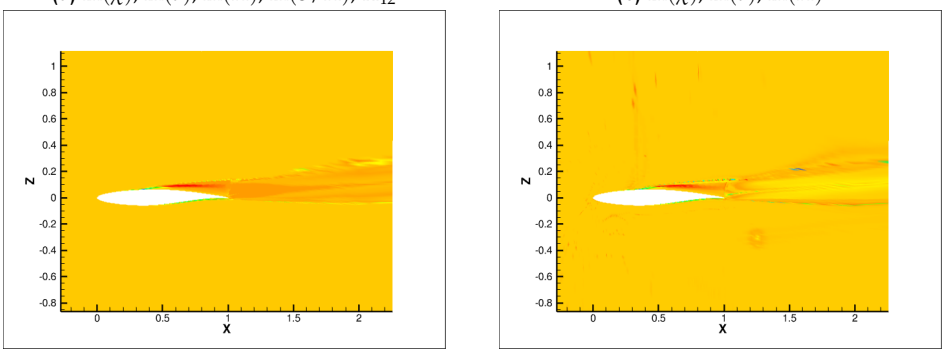


Figure 7.31: C_p distribution for the NN-augmented TAU solutions. Testing is done at flow conditions $M = 0.7206$, $AoA = 5.737$, $Re = 6331480$.

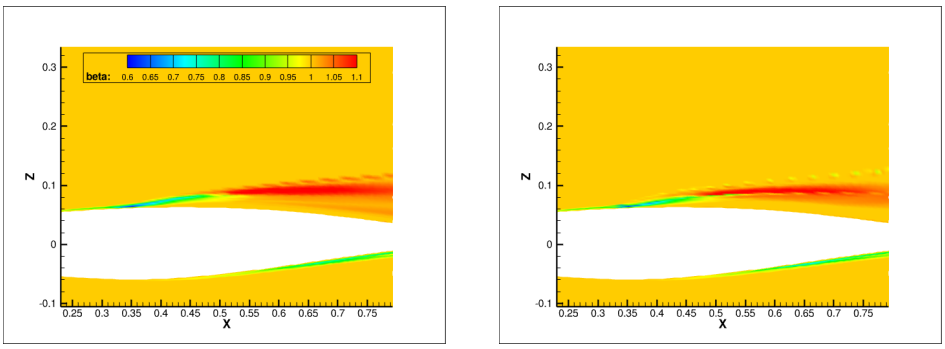


(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12} **(b)** $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

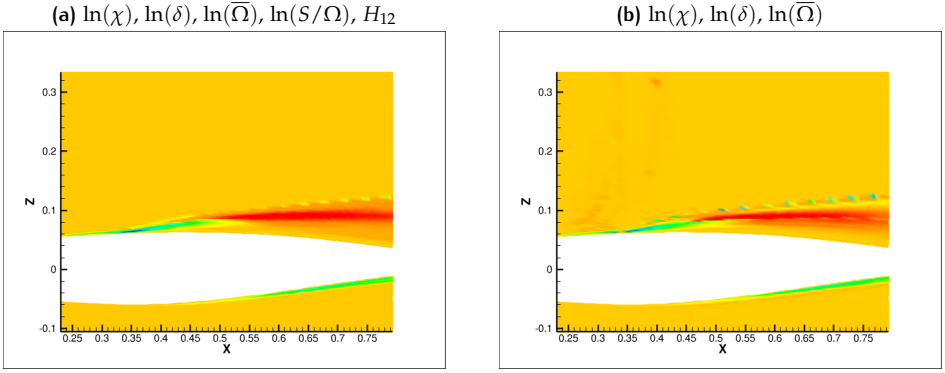


(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12} **(d)** $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

Figure 7.32: Far field results after connecting the trained neural networks with different features to TAU flow solver. Testing is done at flow conditions M = 0.7206, AoA = 5.737, Re = 6331480.



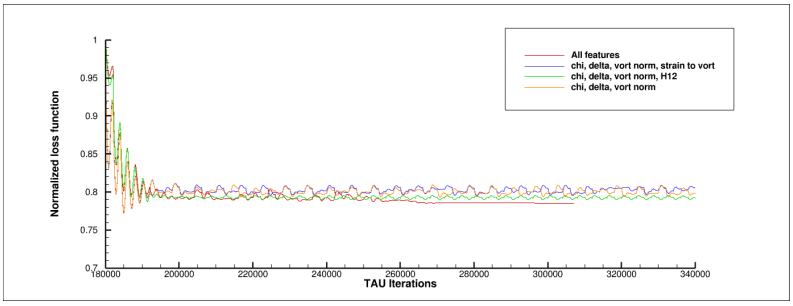
(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12} **(b)** $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$



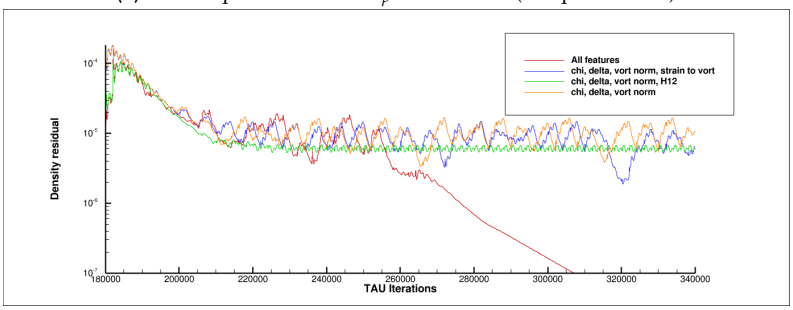
(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12} **(d)** $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

Figure 7.33: Zoom in results after connecting the trained neural networks with different features to TAU flow solver. Testing is done at flow conditions M = 0.7206, AoA = 5.737, Re = 6331480.

The second test case is $M = 0.7421$, $AoA = 4.420$, $Re = 8760680$, where again shock-induced separation is expected to occur. The NN-augmented solution only converges for Training 1 (Figure 7.34b). The shock location identification is better for all the augmented solutions compared to the baseline results, as seen in Figure 7.35. The pressure distribution in the flow after the shock location is not affected at all for the augmented results. In the discrepancy fields displayed in Figure 7.36, only the field for Training 1 shows proper maturity in the solution. Additionally, the resultant $\beta(x)$ also looks similar to what has been observed until now for shock-induced separation.



(a) Mean squared error of C_p distribution (wrt pETW data)



(b) Density residual

Figure 7.34: Residual variation with the flow solution in TAU after augmenting the baseline SA-neg model with the trained neural network after every 2000 iterations. Testing is done at flow conditions $M = 0.7421$, $AoA = 4.420$, $Re = 8760680$.

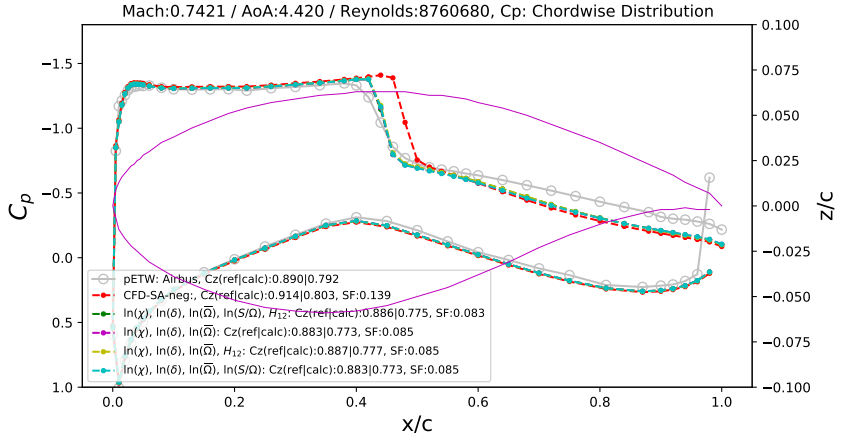
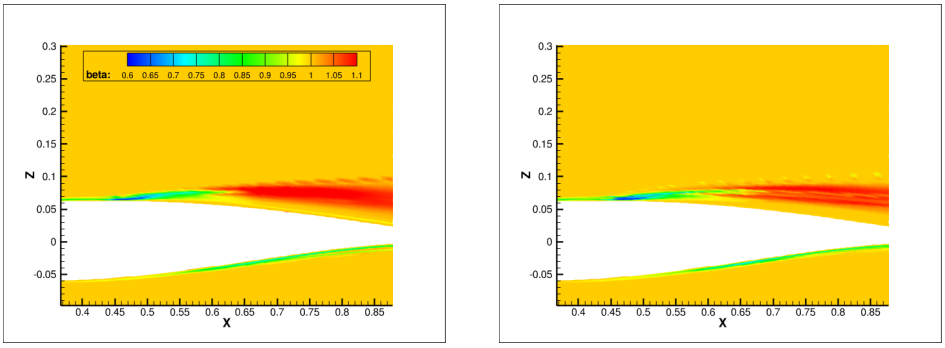


Figure 7.35: C_p distribution for the NN-augmented TAU solutions. Testing is done at flow conditions $M = 0.7421$, $AoA = 4.420$, $Re = 8760680$.



(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

Figure 7.36: Zoom in results after connecting the trained neural networks with different features to TAU flow solver. Testing is done at flow conditions M = 0.7421, AoA = 4.420, Re = 8760680.

For the remaining three test cases, it was seen in Figure 7.22 that the shock is not strong enough to cause a separation on the airfoil. Thus, following from the Re = 9 million dataset results, one would expect that the NN-augmented results will not be too different from the baseline results. This can be confirmed from Figure 7.37, Figure 7.39 and Figure 7.40. There were no problems in the convergence of the augmented solutions for these flow cases, which can be confirmed by viewing the full results in Section D.2.2.

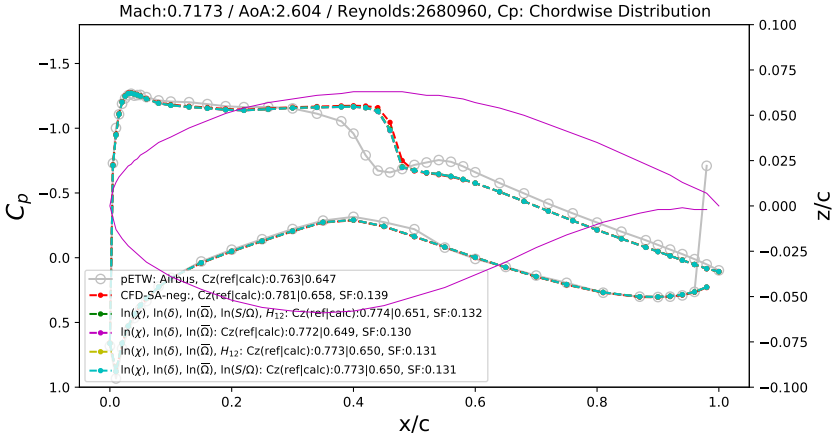
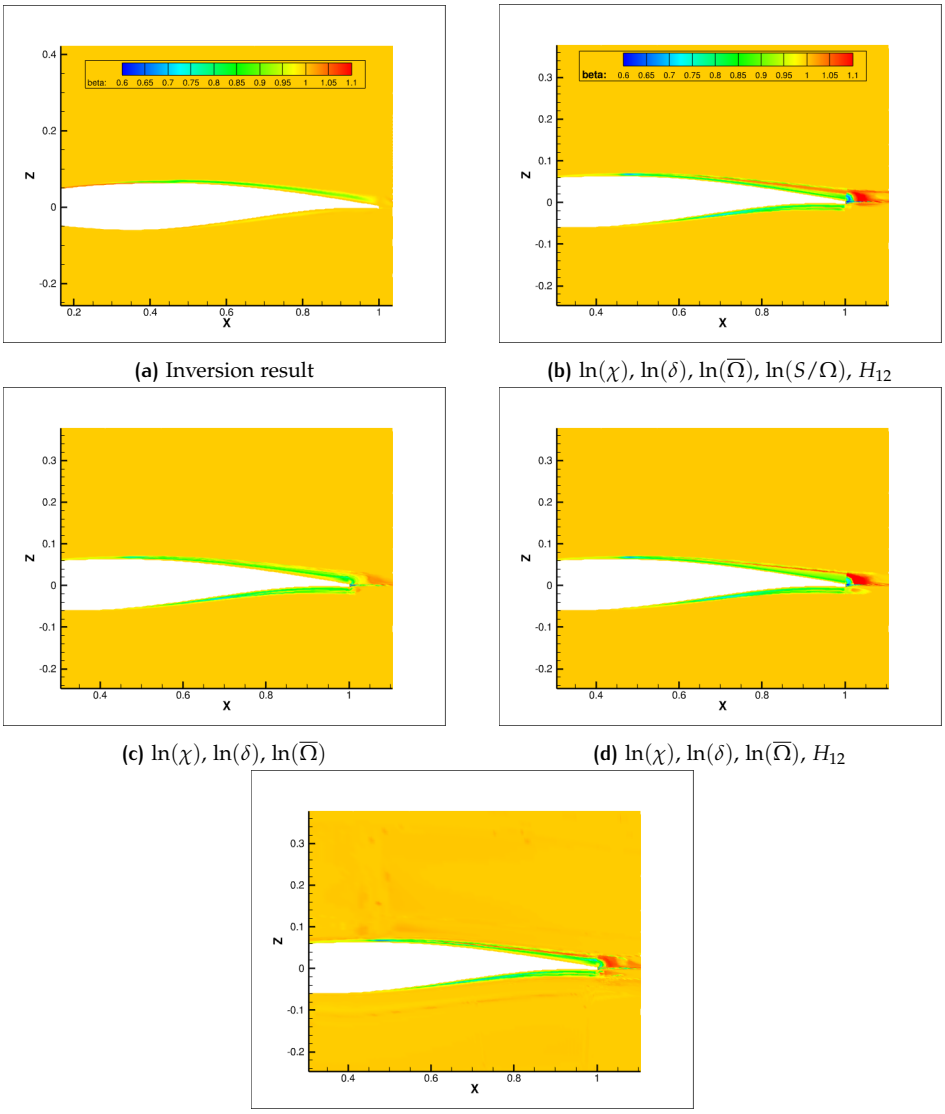


Figure 7.37: C_p distribution for the NN-augmented TAU solutions. Testing is done at flow conditions M = 0.7173, AoA = 2.604, Re = 2680960.



(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

(e) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

Figure 7.38: Results after connecting the trained neural networks with different features to TAU flow solver. Testing is done at flow conditions M = 0.7173, AoA = 2.604, Re = 2680960.

Inversion results are available for the flow case M = 0.7173, AoA = 2.604, Re = 2680960, which is the only unknown flow case tested for both the Re = 9 million dataset and Full dataset. Figure 7.38 shows that the $\beta(x)$ fields for the augmented solutions matches with the inversion solution (strongly regularized) for this case. The difference between predictions between the Re = 9 million dataset (Figure 7.17) and Full dataset (Figure 7.38) is the production increase seen in the latter. However, this has no implications on the resulting pressure distribution. The $\beta(x)$ fields for the flow cases M = 0.7397, AoA = 1.403, Re = 13257500 and M = 0.7110, AoA = 5.145, Re = 15363000 also look similar because the NN-augmentation has no effect for these flow cases. The results for the remaining flow cases can be viewed in Section D.2.2.

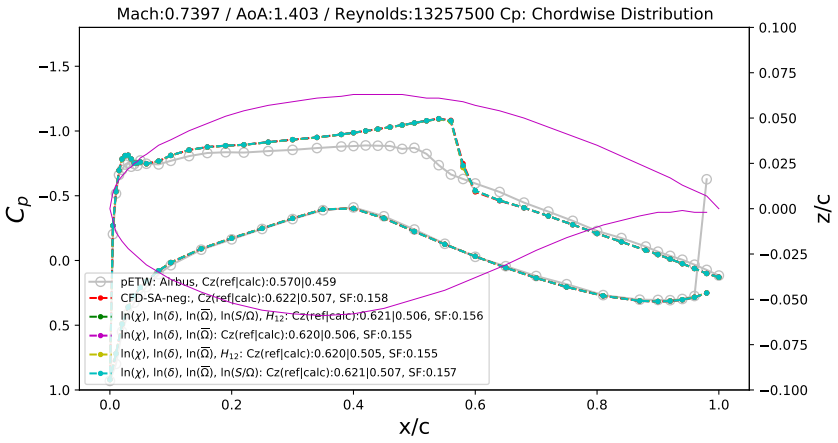


Figure 7.39: C_p distribution for the NN-augmented TAU solutions. Testing is done at flow conditions M = 0.7397, AoA = 1.403, Re = 13257500.

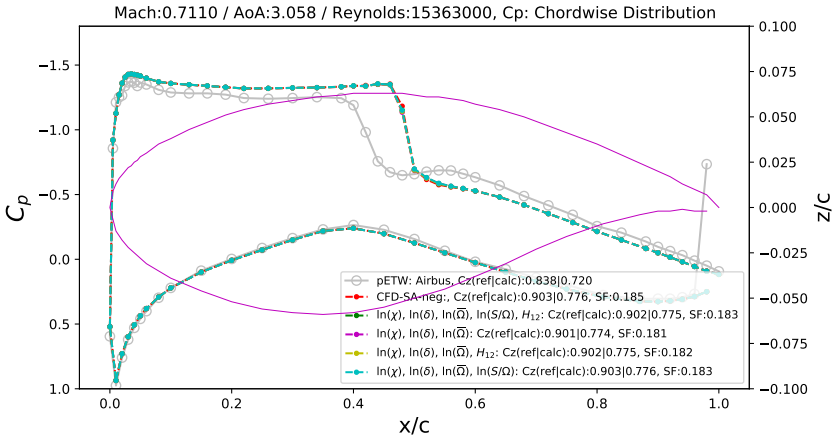


Figure 7.40: C_p distribution for the NN-augmented TAU solutions. Testing is done at flow conditions M = 0.7110, AoA = 5.145, Re = 15363000.

7.3 REFLECTION & RECOMMENDATIONS

This chapter explained how the Machine Learning (ML) part was implemented in the current FIML implementation. Two datasets were used in the study, one generated using an inversion solution only from one flow case (Re = 9 million dataset) and the other using inversion solutions from five (Full dataset). All the flow cases displayed the shock-induced separation phenomena; thus, the eventual ML model training is expected to perform well on such a case. Neural networks were the ML algorithm of choice, whose architecture and hyperparameters were selected using a surrogate-based hyperparameter optimization routine. The results of choosing the architecture using this approach may not be the best as it is influenced by the user's choice of the input range of the design variables, but it is definitely better than a naive approach of choosing the neural network hyperparameters by trial and error.

The selected neural networks were then trained using various combinations of input features from the final feature shortlist after the feature engineering pipeline in the hope of further reducing the number of required features. These trained neural networks were then plugged with the TAU flow solver to augment the SA-

neg turbulence model, with the calculation starting from the converged baseline solution. The NN-augmented was tested first on unseen flow conditions to test the extrapolation capability of the augmentation. The primary observation was that the NN-augmented results in a corrected shock location even for an unknown flow condition causing a shock wave strong enough to cause shock-induced separation. This observation is consistent with the data provided to the neural networks, which were selected from inversion solutions regularized just enough to capture the shock location well and not overfit the post-shock, uncertain, turbulent flow.

This was observed for neural networks trained on both datasets, therefore having more data in the Full dataset was not particularly beneficial to correctly identify the shock-induced separation phenomena. It was observed that for the Re = 9 million dataset, at least four features are required to have a good β field prediction which converges well when connected to the TAU solver, but for the Full dataset, three input features were just enough. Choosing the features based on a visual correlation with the output was also a viable strategy, which could boost both the performance and flow solution output of the NN-augmented RANS run. Additionally, visual elements of the chosen input features could be observed in the resulting β fields.

For the flow cases not expected to have a shock-induced separation, it was seen that the NN-augmentation had little to no effect on the baseline results. The hope was that the method of choosing flow features detailed in Chapter 6 would overcome the fact that data for other flow phenomena is not provided to the neural network, but that was not the case. However, it is encouraging that the NN-augmentation did not make the results worse starting from the baseline solution.

During the current ML implementation, there were multiple avenues of further research not pursued due to limited time at the author's end. These have been provided as recommendations below to aid further researchers delving into this topic:

- Incorporation of more flow cases from the pETW database in the training data: This suggestion follows from the expectation that if the neural network observes data from different flow phenomena, like shock waves with no separation on the airfoil, it will be able to identify the same phenomena at a different unknown flow condition. This addition will also lead to the development of a more generalized model.
- Testing other data compositions and its influence on results: The current training data was generated by selecting all "Non-ones" or points with $\beta \notin (0.98, 1.02)$ and then "Ones", i.e. points with $\beta \in [0.98, 1.02]$ were randomly chosen, with the number of "Ones" was chosen to be 20% of "Non-ones". The range (0.98, 1.02) and percentage 20% were selected with little knowledge of how other compositions would perform due to less time to test them extensively, and a detailed analysis on this would provide valuable insight.
- Testing on cases outside the database: Tests should have been done on flow cases from other high-fidelity data sources to verify whether the augmented solution is general enough to be applied out of the box. This may include testing other airfoil profiles with similar flow conditions or randomly chosen flow conditions outside the database.
- Fine-tuning the neural network training process: Some choices while training the neural network were made on an informed opinion, but an evidence approach may provide better validation. This includes the choice of the loss function (Mean Square Error), activation function ('ReLU'), optimizer (Adam) and many more.

8 CONCLUSIONS & FUTURE WORK

8.1 CONCLUSION

The conclusion for this project is best done by answering the research questions formulated at the start of this document.

"How can the Field Inversion and Machine Learning (FIML) approach be applied to augment the negative Spalart-Allmaras (SA-neg) turbulence model for shock-induced separation on a 2D transonic airfoil?"

This work employed a data-driven approach based on the Field Inversion and Machine Learning (FIML) framework explained in Section 3.3, implemented in DLR's software ecosystem [Jäckel, 2020] for the negative Spalart-Allmaras turbulence model. A database generated by Airbus in the pilot facility of the ETW (pETW) for the RAE2822 airfoil was exploited to perform this data-driven approach. Cases in the database were identified with significant deviations between measured (wind tunnel) and computed (CFD results) pressure distribution, which were used to perform the Field Inversion procedure. The Field Inversion procedure uses a gradient-based optimization approach, the gradients for which are obtained using a discrete adjoint approach in TAU [Dwight and Brezillon, 2006]. The solution from this procedure was processed using a feature selection and engineering process exclusive to this work to generate the training data for the Machine Learning part of the FIML procedure. This work's focus was mainly on the applicability of the FIML approach to shock-induced flow separation, and a general augmented RANS turbulence model could not be implemented in the limited period for the MSc thesis.

The sub-questions have been answered below:

- *Is the experimental aerodynamic database of high-fidelity wind-tunnel measurements dataset relevant and representative of the flow cases on which the augmented turbulence model will be applied?*

  The pETW data provided by Airbus for the RAE 2822 airfoil is an extensive database covering a large range of Mach number, Reynolds number and angles of attack, making it the most extensive database for the transonic flow around RAE2822 airfoil, making it a relevant data source that can be used for this study. The database is representative of this study's goal as flow cases with shock-induced separation, the focus of the study could be easily found across a range of Reynolds number values. This database is a significant upgrade over the legacy data by AGARD [Cook et al., 1979] which has been used for multiple aerodynamic numerical code validation activities. Furthermore, Airbus provided a CFD database, which aided in formulating a viable test case selection strategy for the current FIML implementation. The assumptions behind the experiment for the pETW database and the limitations of the CFD database were kept in mind while formulating this strategy. The only qualm with this database could be its lack of maturity in applications for data-driven turbulence modelling approaches, as this is the first study in the author's knowledge that uses this database.

- *Does a unique discrepancy field exists which can be inferred using a deterministic inversion approach? Does this unique field correspond to the global minimum of the discrepancy between the baseline model and high-fidelity data? Is it possible to reach this solution with the current tools and computational resources at hand?*

  The choice of deterministic inversion approach used for this study was primarily based on the ease of implementation in the DLR's software ecosystem. A gradient-based optimizer employing the Steepest Gradient approach was implemented to solve the Field Inversion problem. For a given restart solution and Tikhonov regularization value, a unique discrepancy field solution could be reached for the chosen convergence criteria. However, the global minimum solution was not reached. The L-curve approach [Hansen and O'Leary, 1993] failed in the current implementation, and other remedies like restarting from a random solution either failed or were not given enough time to mature. Furthermore, it was not even sure that the global minimum should be reached as it may have meant that the inversion solution fits too closely to the experimental results, hence reducing its generalization ability for an unseen flow case during the Machine Learning phase.

  The final inversion solution chosen for the analysis was based on reasoning around physical phenomena related to the selected flow cases. The regularization values were chosen so that the shock location improvement is captured well, but it does not overfit the flow trailing the shock location. This is the best that could be implemented with the current tools and computational resources at hand. A gradient-free method like the Ensemble Kalman filter (EnKF) could have pointed us to the global minimum, but it was simply not implemented due to the time needed.

- *Will machine learning algorithms be able to find the patterns in the discrepancies obtained from inversion solutions? Are neural networks the best algorithm for the learning process in this case? What is the best architecture/ set of hyper-parameters for the neural networks to be used for learning?*

  From a literature survey of various FIML implementations, it was seen that machine learning algorithms are more than able to generalize the discrepancy from inversion solutions. The successful testing of flow cases with shock-induced separation in this study is a testament that the intended patterns in the training data can be generalized to unseen flow cases. However, the training data needs to be carefully prepared from the inversion solution, keeping in mind the intended patterns to be taught to the ML algorithm and the needs of the ML algorithm itself.

  Neural networks are the best choice for the ML algorithm in terms of its capability to learn complex non-linear function approximations. Additionally, the ease of implementation with currently available Python libraries and its integration with DLR software make it a feasible option for the current study. A possible con is the "black-box" nature of how this approximation is learnt, which would not be the case if this function approximation could be presented as a polynomial using a candidate of library functions as done by Schmelzer et al. [2020]. The best set of hyper-parameters for the neural networks is chosen using the surrogate model based hyperparameter optimization routine implemented by [Sabater et al., 2021] in the DLR-SMARTy [Görtz et al., 2013] framework. Thus, an efficient, systematic strategy was used to round down to the final architecture.

- *What is the appropriate set of features required for the learning process? How many features are necessary? Should the features specific to the test case (shock-induced*

separation) be introduced in the learning process?

An initial shortlist of 13 features was formed based on an extensive data-driven RANS turbulence modelling and FIML literature survey. Special care was taken while choosing these features; the features are motivated by physical phenomena (shock-induced separation), adhere to turbulence modelling requirements, and are amenable to neural networks by appropriate feature preprocessing procedures. A feature selection and- engineering process has been detailed in this work to determine the appropriate set of features and the number of features required. This process employs various techniques such as log transformations and scaling operations, Spearman correlation and Sequential Feature Selection (SFS) algorithm [Raschka and Mirjalili, 2017] with bidirectional elimination. It gives an output of a final list of features for machine learning for a given dataset. Furthermore, attempts to reduce this final list were made by testing various subsets of this final list of features to verify the predictive ability.

- *Are the results of this learning process interpretable? Is there a causal relationship between the data, the discrepancy, and the corresponding prediction?*

  The training data provided to the neural network learning originates from the inversion solution for flow cases where shock-induced separation phenomena occur. The discrepancy field results for the NN-augmented RANS model seen in Chapter 7 generally show the following behaviour for shock-induced separation: a turbulence production decrease near the shock foot, followed by a separated turbulence production increase in the downstream flow until the trailing edge (for instance in Figure 7.11). This behaviour of the discrepancy fields is in line with the expectations of the physical phenomena, making the learning process interpretable. This discrepancy field results in an upstream shift of the shock location, generally agreeing with the pETW data, in the resulting pressure distribution. Thus, there is a causal relationship between the training data, the discrepancy fields, and the corresponding prediction. The predictions do not affect the flow after the shock, which was intended during the inversion procedure. Furthermore, the NN-augmented results for a flow case not displaying shock-induced separation do not vary significantly from the baseline RANS model results, which implies that the NN-augmentation of the model will only respond to flow phenomena it has seen; which points to a causal relationship.

- *Does this ML-augmentation provide better results than the baseline model? Can it predict test cases with design points unrelated to those in the training database?*

  The current ML-augmentation provides better results than the baseline model by correctly identifying the shock location in a shock-induced separation case. It is confirmed by tests in Chapter 7 that the ML-augmentation works for unseen flow conditions at training time as long as the shock-induced separation is expected to happen. For other flow cases, there is little to no improvement to the baseline model. Incorporating more training data with different flow phenomena may remedy this problem, leading to a more general augmented RANS turbulence model.

8.2 SCOPE FOR FUTURE WORK

The goal of any data-driven augmentation of a turbulence model is to be a general model augmentation applicable to any flow case. To improve the current FIML

implementation to work towards the said goal, the following suggestions can be implemented at various steps of the project:

- **FIML approach**:
  - Using more tightly coupled approaches: The current FIML approach is the classic FIML approach given by Parish and Duraisamy [2016] which does ensure model-consistent learning as discussed in Chapter 3. However, more tightly coupled approaches like Embedded FIML and Direct FIML [Holland, 2019], and Learning and Inference assisted by Feature-space Engineering (LIFE) [Srivastava and Duraisamy, 2021] can be used to move towards a more general and robust model.
  - Zonal prediction approaches: An ML algorithm that identifies the zones of turbulence production changes as a classification problem and then predicts the correct values of discrepancy as a regression problem is an idea that can be implemented. A similar idea was implemented by Matai and Durbin [2019], where the discrepancy field was classified into various clusters, and then a decision tree algorithm chose the input features to be used in those clusters.
- **Field Inversion**:
  - Work towards incorporating transition location in the optimization problem for field inversion by trying the recommendations provided in Section 5.4.
  - Carefully choosing the restart β field for the field inversion procedure by following the suggestions in Section 5.4.
  - Using a gradient-based optimizer other than SGD, which can escape the local minima better. Alternatively, a gradient-free optimizer can be used to be more assured of reaching the global minimum.
- **Feature Engineering**:
  - Adding more features to the initial shortlist: The initial feature shortlist could be expanded by using features based on streamline curvature [Wang et al., 2017; Volpiani et al., 2021], wall-distance based Reynolds number [Ling and Templeton, 2015; Wang et al., 2017] and many more.
  - Testing other feature transformations: Feature transformations employed in literature (for instance, by Ling and Templeton [2015]), or using other mathematical operators (for instance, $\ln(1 + x_i)$, where x_i is the feature) can be checked whether they provide any notable gains compared to the current strategy.
  - Giving more consideration from a commercial solver standpoint: The DLR TAU flow solver makes available input features that are not generally available in a commercial flow solver. Additionally, the current study uses some wall-distance based features which would not work for an unstructured grid solver as they are non-local. Thus, consider these factors for the trained ML model to be used out of the box in any solver.
- **Machine Learning**
  - Inclusion of more generalized training and test cases: More flow cases can be added to the training data to head towards a general model, and testing on a range of test cases with varying geometry, flow cases can validate the generalization capability. More details are provided in Section 7.3.
  - Other ML algorithms: Many ML algorithms have been used in FIML literature with varying degree of success in terms of ease of implementation and accuracy. These include Random Forests, boosting algorithm like

AdaBoost, Gaussian processes, etc. and a comparative study of their performance with the current neural network implementation would have added value to the work.

– Use of explainable function approximation: The ML algorithm could be replaced by a more explainable function. Examples of this are: sparse symbolic regression [Schmelzer et al., 2020], smooth RBF function approximations [Jäckel, 2022].
– Extensive treatment of invariance: The current study uses scalar features to work around the invariance of the resulting data-driven turbulence model. However, other approaches like tensor basis neural networks proposed by Ling et al. [2016] or tensor basis random forests implemented by Kaandorp and Dwight [2018] could have been employed

BIBLIOGRAPHY

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Babinsky, H. and Harvey, J. K. (2011). *Shock wave-boundary-layer interactions*, volume 32. Cambridge University Press.

Baldwin, B. and Lomax, H. (1978). Thin-layer approximation and algebraic model for separated turbulentflows. In *16th aerospace sciences meeting*, page 257.

Brunton, S. L., Noack, B. R., and Koumoutsakos, P. (2019). Machine learning for fluid mechanics. *Annu. Rev. Fluid Mech. 2020*, 52:477–508.

Chollet, F. (2018). *Deep learning with Python*. Simon and Schuster.

Clauser, F. H. (1954). Turbulent boundary layers in adverse pressure gradients. *Journal of the Aeronautical Sciences*, 21(2):91–108.

Cook, P., McDonald, M., and Firmin, M. (1979). Aerofoil rae 2822: Pressure distribution and boundary layer and wake measurements. agard ar 138. *Research and Technology Organi-sation, Neuilly-sur-Seine*.

Davidon, W. C. (1991). Variable metric method for minimization. *SIAM Journal on Optimization*, 1(1):1–17.

Duraisamy, K. (2020). Perspectives on machine learning-augmented reynolds-averaged and large eddy simulation models of turbulence.

Duraisamy, K. and Durbin, P. (2014). Transition modeling using data driven approaches. In *Proceedings of the Summer Program*, page 427.

Duraisamy, K., Iaccarino, G., and Xiao, H. (2019). Annual review of fluid mechanics turbulence modeling in the age of data. *Annu. Rev. Fluid Mech*, 51:357–377.

Duraisamy, K., Zhang, Z. J., and Singh, A. P. (2015). New approaches in turbulence and transition modeling using data-driven techniques. In *53rd AIAA Aerospace Sciences Meeting*, page 1284.

Dwight, R. P. and Brezillon, J. (2006). Effect of approximations of the discrete adjoint on gradient-based optimization. *AIAA Journal*, 44(12):3022–3031.

Ferrero, A., Iollo, A., and Larocca, F. (2020). Field inversion for data-augmented RANS modelling in turbomachinery flows. *Computers and Fluids*, 201.

Ferri, F. J., Pudil, P., Hatef, M., and Kittler, J. (1994). Comparative study of techniques for large-scale feature selection. In *Machine Intelligence and Pattern Recognition*, volume 16, pages 403–413. Elsevier.

Fletcher, R. (2013). *Practical methods of optimization*. John Wiley & Sons.

Fletcher, R. and Reeves, C. M. (1964). Function minimization by conjugate gradients. *The computer journal*, 7(2):149–154.

Forrester, A. I. and Keane, A. J. (2009). Recent advances in surrogate-based optimization. *Progress in aerospace sciences*, 45(1-3):50–79.

Garbaruk, A., Shur, M., Strelets, M., and Spalart, P. R. (2003). Numerical study of wind-tunnel walls effects on transonic airfoil flow. *AIAA journal*, 41(6):1046–1054.

Giles, M. B. and Pierce, N. A. (2000). An introduction to the adjoint approach to design. *Flow, turbulence and combustion*, 65(3):393–415.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR.

Görtz, S., Zimmermann, R., and Han, Z.-H. (2013). Variable-fidelity and reduced-order models for aero data for loads predictions. In *Computational Flight Testing*, pages 99–112. Springer.

Guyon, I. and De, A. M. (2003). An introduction to variable and feature selection andré elisseeff.

Hansen, P. C. and O'Leary, D. P. (1993). The use of the l-curve in the regularization of discrete ill-posed problems. *SIAM journal on scientific computing*, 14(6):1487–1503.

Hilbert, D. and David, H. (1993). *Theory of algebraic invariants*. Cambridge university press.

Holland, J. R. (2019). *Integrated Field Inversion and Machine Learning with Embedded Neural Network Training for Turbulence Modeling*. PhD thesis, University of Maryland, College Park.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Iglesias, M. A. (2016). A regularizing iterative ensemble kalman method for pde-constrained inverse problems. *Inverse Problems*, 32(2):025002.

Iglesias, M. A., Law, K. J., and Stuart, A. M. (2013). Ensemble kalman methods for inverse problems. *Inverse Problems*, 29(4):045001.

Jones, W. and Launder, B. E. (1972). The prediction of laminarization with a two-equation model of turbulence. *International journal of heat and mass transfer*, 15(2):301–314.

Jäckel, F. (2020). Sensitivity analysis of discrepancy terms introduced in turbulence models using field inversion.

Jäckel, F. (2022). A closed-form correction for the Spalart-Allmaras turbulence model for separated flows. In *accepted for presentation and publication on the AIAA SciTech2022 Forum, Jan. 2022*.

Kaandorp, M. and Dwight, R. (2018). Stochastic random forests with invariance for rans turbulence modelling. *arXiv preprint arXiv:1810.08794*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Köhler, F., Munz, J., and Schäfer, M. (2020). Data-driven augmentation of rans turbulence models for improved prediction of separation in wall-bounded flows. In *AIAA Scitech 2020 Forum*, page 1586.

Lau, M. M. and Hann Lim, K. (2018). Review of adaptive activation function in deep neural network. In *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, pages 686–690.

Ling, J., Jones, R., and Templeton, J. (2016). Machine learning strategies for systems with invariance properties. *Journal of Computational Physics*, 318:22–35.

Ling, J. and Templeton, J. (2015). Evaluation of machine learning algorithms for prediction of regions of high reynolds averaged navier stokes uncertainty. *Physics of Fluids*, 27.

Mani, M., Babcock, D. A., Winkler, C. M., and Spalart, P. R. (2013). Predictions of a supersonic turbulent flow in a square duct. *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition 2013*.

Matai, R. and Durbin, P. A. (2019). Zonal eddy viscosity models based on machine learning. *Flow, Turbulence and Combustion*, 103:93–109.

Medida, S. (2014). *Correlation-based transition modeling for external aerodynamic flows*. PhD thesis, University of Maryland, College Park.

Mellor, G. L. (1966). The effects of pressure gradients on turbulent flow near a smooth wall. *Journal of Fluid Mechanics*, 24(2):255–274.

Moré, J. J. (1978). The levenberg-marquardt algorithm: implementation and theory. In *Numerical analysis*, pages 105–116. Springer.

Obayashi, S. and Tsukahara, T. (1997). Comparison of optimization algorithms for aerodynamic shape design. *AIAA journal*, 35(8):1413–1415.

Papalambros, P. Y. and Wilde, D. J. (2000). *Principles of optimal design: modeling and computation*. Cambridge university press.

Parish, E. J. and Duraisamy, K. (2016). A paradigm for data-driven predictive modeling using field inversion and machine learning. *Journal of Computational Physics*, 305:758–774.

Patel, V. C. (1973). A unified view of the law of the wall using mixing-length theory. *Aeronautical Quarterly*, 24(1):55–70.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Peter, J. E. and Dwight, R. P. (2010). Numerical sensitivity analysis for aerodynamic optimization: A survey of approaches. *Computers & Fluids*, 39(3):373–391.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151.

Raschka, S. (2018). Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack. *The Journal of Open Source Software*, 3(24).

Raschka, S. and Mirjalili, V. (2017). *Python machine learning : machine learning and deep learning with Python, scikit-learn, and TensorFlow*. Packt Publishing.

Reimer, L. (2015). The FlowSimulator—a software framework for CFD-related multidisciplinary simulations.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Rumsey, C., Smith, B., and Huang, G. (2010). Description of a website resource for turbulence modeling verification and validation. In *40th Fluid Dynamics Conference and Exhibit*, page 4742.

Saad, Y. and Schultz, M. H. (1986). Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869.

Sabater, C., Bekemeyer, P., and Görtz, S. (2020). Efficient bilevel surrogate approach for optimization under uncertainty of shock control bumps. *AIAA Journal*, 58(12):5228–5242.

Sabater, C., Stürmer, P., and Bekemeyer, P. (2021). Fast predictions of aircraft aerodynamics using deep learning techniques. In *AIAA Aviation 2021 Forum*, page 2549.

Schewe, G., Mai, H., and Dietz, G. (2003). Nonlinear effects in transonic flutter with emphasis on manifestations of limit cycle oscillations. *Journal of Fluids and Structures*, 18(1):3–22.

Schmelzer, M., Dwight, R. P., and Cinnella, P. (2020). Discovery of algebraic reynolds-stress models using sparse symbolic regression. *Flow, Turbulence and Combustion*, 104(2):579–603.

Schwamborn, D., Gerhold, T., and Heinrich, R. (2006). The DLR TAU-code: recent applications in research and industry.

Singh, A. P. (2018). *A framework to improve turbulence models using full-field inversion and machine learning*. PhD thesis, The University of Michigan.

Singh, A. P., Duraisamy, K., and Pan, S. (2017a). *Characterizing and Improving Predictive Accuracy in Shock-Turbulent Boundary Layer Interactions Using Data-driven Models*.

Singh, A. P., Medida, S., and Duraisamy, K. (2017b). Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils. *AIAA Journal*, 55:2215–2227.

Skinner, S. N. and Zare-Behtash, H. (2018). State-of-the-art in aerodynamic shape optimisation methods. *Applied Soft Computing*, 62:933–962.

Spalart, P. and Allmaras, S. (1992). A one-equation turbulence model for aerodynamic flows. In *30th aerospace sciences meeting and exhibit*, page 439.

Spalart, P. R. (2000). Strategies for turbulence modelling and simulations. *International journal of heat and fluid flow*, 21(3):252–263.

Spalart, P. R. (2015). Philosophies and fallacies in turbulence modeling.

Spalart, P. R., Deck, S., Shur, M. L., Squires, K. D., Strelets, M. K., and Travin, A. (2006). A new version of detached-eddy simulation, resistant to ambiguous grid densities. *Theoretical and Computational Fluid Dynamics*, 20:181–195.

Spalart, Allmaras, S. R. and Johnson, F. T. (2012). Modifications and clarifications for the implementation of the spalart-allmaras turbulence model. In *Seventh international conference on computational fluid dynamics (ICCFD7)*, volume 1902. Big Island, HI.

Srivastava, V. and Duraisamy, K. (2021). Generalizable physics-constrained modeling using learning and inference assisted by feature space engineering.

Tan, H. H. and Lim, K. H. (2019). Vanishing gradient mitigation with deep learning neural network optimization. In *2019 7th International Conference on Smart Computing Communications (ICSCC)*, pages 1–4.

Tarantola, A. (2021). Inverse problem. From *MathWorld*–A Wolfram Web Resource, created by Eric W. Weisstein.

Tikhonov, A. N. and Arsenin, V. Y. (1977). Solutions of ill-posed problems. *New York*, 1(30):487.

Tracey, B., Duraisamy, K., and Alonso, J. (2013). Application of supervised learning to quantify uncertainties in turbulence and combustion modeling. In *51st AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition*, page 259.

van Korlaar, A. (2019). Field inversion and machine learning in turbulence modeling.

Volpiani, P. S., Meyer, M., Franceschini, L., Dandois, J., Renac, F., Martin, E., Marquet, O., and Sipp, D. (2021). Machine learning-augmented turbulence modeling for rans simulations of massively separated flows. *Physical Review Fluids*, 6.

Wang, J. X., Wu, J. L., and Xiao, H. (2017). Physics-informed machine learning approach for reconstructing reynolds stress modeling discrepancies based on dns data. *Physical Review Fluids*, 2.

Weatheritt, J. and Sandberg, R. (2017). The development of algebraic stress models using a novel evolutionary algorithm. *International Journal of Heat and Fluid Flow*, 68:298–318.

Wilcox, D. C. (1988). Reassessment of the scale-determining equation for advanced turbulence models. *AIAA journal*, 26(11):1299–1310.

Wu, J. L., Xiao, H., and Paterson, E. (2018). Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework. *Physical Review Fluids*, 7.

Yang, M. and Xiao, Z. (2020). Improving the k- - -artransition model by the field inversion and machine learning framework. *Physics of Fluids*, 32.

Zingg, D. W., Nemec, M., and Pulliam, T. H. (2008). A comparative evaluation of genetic and gradient-based algorithms applied to aerodynamic optimization. *European Journal of Computational Mechanics/Revue Européenne de Mécanique Numérique*, 17(1-2):103–126.

A APPENDIX: THEORETICAL BACKGROUND

A.1 DETAILS: SPALARAT–ALLMARAS MODEL AND NEGATIVE SA VARIANT

Original model by [Spalart and Allmaras, 1992]. Trip term added in 1994 version. The description is given in the website https://turbmodels.larc.nasa.gov/spalart.html or here. The description about the website is given in the paper by Rumsey et al..

The Reynolds stresses are defined using the Boussinesq eddy viscosity assumption, and the eddy viscosity (ν_t) is defined in terms of a SA working variable $\tilde{\nu}$ in the following manner:

$$\nu_t = \frac{\mu_t}{\rho} = \tilde{\nu} f_{v1}, \quad f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3}, \quad \chi \equiv \frac{\tilde{\nu}}{\nu}, \tag{A.1}$$

where ρ is the density, $\nu = \frac{\mu}{\rho}$ is the kinematic viscosity. The SA working variable $\tilde{\nu}$ follows the transport equation:

$$\frac{D\tilde{\nu}}{Dt} = P - D + T + \frac{1}{\sigma}\left[\nabla.((\nu + \tilde{\nu})\nabla\tilde{\nu}) + c_{b2}\nabla(\tilde{\nu})^2\right], \tag{A.2}$$

where P, D and T are the production, destruction and trip terms respectively. These are defined as:

$$P = c_{b1}(1 - f_{t2})\widetilde{\Omega}\tilde{\nu}, \quad D = \left(c_{w1} f_w - \frac{c_{b1}}{\kappa^2} f_{t2}\right)\left[\frac{\tilde{\nu}}{d}\right]^2, \quad T = f_{t1}(\Delta u)^2, \tag{A.3}$$

where $\widetilde{\Omega}$ is the modified vorticity. This is derived from $\Omega = \sqrt{2\Omega_{ij}\Omega_{ij}}$ i.e. the magnitude of vorticity and d being the distance of the closest wall point (referred to as wall distance) in the following mannner:

$$\widetilde{\Omega} = \Omega + \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}, \quad f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}}. \tag{A.4}$$

The wall function f_w is defined as:

$$f_w = g\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6}, \quad g = r + c_{w2}(r^6 - r), \quad r = \min\left(\frac{\tilde{\nu}}{\kappa^2 d^2 \widetilde{\Omega}}, r_{lim}\right). \tag{A.5}$$

Trip and laminar suppression terms are

$$f_{t1} = c_{t1} g_t \exp\left(-c_{t2}\frac{\omega_t}{\Delta u^2}\left[d^2 + g_t^2 d_t^2\right]\right), \quad f_{t2} = c_{t3}\exp\left(-c_{t4}\chi^2\right)$$

with $g_t = \min(0.1, \Delta u/\omega_t \Delta x)$, where d_t is distance to the trip point, ω_t is the vorticity at the trip, Δu is the difference in velocity relative the trip point, and Δx is streamwise grid spacing at the trip. The constants are $c_{b1} = 0.1355, \sigma = 2/3, c_{b2} = 0.622, \kappa = 0.41, c_{w1} = c_{b1}/\kappa^2 + (1 + c_{b2})/\sigma, c_{w2} = 0.3$, $c_{w3} = 2, c_{v1} = 7.1, c_{t1} = 1, c_{t2} = 2, c_{t3} = 1.2, c_{t4} = 0.5$, and $r_{1\text{lim}} = 10$. Turbulent heat transfer obeys a turbulent Prandtl number equal to 0.9. Boundary conditions for $\tilde{\nu}$ are

$$\text{no-slip wall:} \quad \tilde{\nu} = 0 \quad \text{symmetry plane:} \ \frac{\partial \tilde{\nu}}{\partial n} = 0$$

A.1.1 Preventing Negative Values of Modified Vorticity $\widetilde{\Omega}$

In physically relevant situations, the modified vorticity $\widetilde{\Omega}$ should always be positive with a value that never falls below 0.3Ω, where Ω is the vorticity magnitude. However, discretely this is not always the case. It is possible for $\widetilde{\Omega}$ to become zero or negative due to the fact that f_{v2} is itself negative over a range of χ. Negative $\widetilde{\Omega}$ in turn disrupts other SA correlation functions. We present a modified form of $\widetilde{\Omega}$ that is identical to the original for $\widetilde{\Omega} > 0.3\Omega$, but remains positive for all nonzero Ω and is C^1 continuous:

$$\bar{\Omega} = \frac{\tilde{\nu}}{\kappa^2 d^2} f_{v2}$$

$$\widetilde{\Omega} = \begin{cases} \Omega + \bar{\Omega} & : \quad \bar{\Omega} \geq -c_{v2}\Omega \\ \Omega + \frac{\Omega\left(c_{v2}^2\Omega + c_{v3}\bar{\Omega}\right)}{(c_{v3} - 2c_{v2})\Omega - \bar{\Omega}} & : \quad \bar{\Omega} < -c_{v2}\Omega \end{cases}$$

with $c_{v2} = 0.7$ and $c_{v3} = 0.9$. The modified function is plotted in Fig. 1. The constant c_{v2} controls the patch point; value and derivative with respect to Ω are matched at $\Omega = (1 - c_{v2})\,\Omega$. The constant c_{v3} controls the asymptote,

$$\widetilde{\Omega} \to (1 - c_{v3})\,\Omega \quad \text{as} \quad \bar{\Omega}/\Omega \to -\infty$$

A.1.2 Negative model

We formulate a continuation of SA into the realm of negative $\tilde{\nu}$ solutions to deal with situations of undershoots. Although an analytic continuation of SA, its primary purpose is to address issues with underresolved grids and non-physical transient states in discrete settings. The negative SA model is proposed with the following properties:

- original (positive) SA is unchanged for $\tilde{\nu} \geq 0$
- negative $\tilde{\nu}$ produces zero eddy viscosity
- functions in the PDE are C^1 continuous with respect to $\tilde{\nu}$ at $\tilde{\nu} = 0$
- negative SA is energy stable
- the analytic solution is non-negative given non-negative boundary conditions

Consider a negative SA model of the form,

$$\frac{D\tilde{\nu}}{Dt} = P_n - D_n + \frac{1}{\sigma}\nabla \cdot \left[\left(\nu + \tilde{\nu} f_n\right) \nabla\tilde{\nu}\right] + \frac{c_{b2}}{\sigma}(\nabla\tilde{\nu})^2$$

where P_n is production, D_n is wall destruction and $f_n(\chi)$ is a modification to the diffusion coefficient. For C^1 continuity at $\tilde{\nu} = 0$, we require

$$P_n|_0 = D_n|_0 = 0, \quad \left.\frac{\partial P_n}{\partial\tilde{\nu}}\right|_0 = c_{b1}\,(1 - c_{t3})\,\Omega, \quad \left.\frac{\partial D_n}{\partial\tilde{\nu}}\right|_0 = 0$$

$$f_n(0) = 1, \quad \left.\frac{\partial f_n}{\partial\chi}\right|_0 = 0$$

When $\tilde{\nu}$ is negative, the eddy viscosity is set to zero, and $\tilde{\nu}$ itself becomes a passive scalar.

The usual requirements for energy stability then give the constraints,

$$P_n - D_n \geq 0, \quad 1 + \chi\,(f_n - c_{b2}) \geq 0$$

The resulting steady PDE is,

$$P_n - D_n + \frac{1}{\sigma}\left(\nu + \tilde{\nu} f_n\right)\nabla^2\tilde{\nu} = 0$$

If the diffusion coefficient mimics the positive model behavior for large $|\tilde{\nu}|$, then f_n should asymptote to -1. Maximizing the region over which the diffusion coefficient turns from $\nu + \tilde{\nu}$ for positive to $\nu + |\tilde{\nu}|$ for large negative, we arrive at,

$$f_n = \frac{c_{n1} + \chi^3}{c_{n1} - \chi^3}$$

with $c_{n1} = 16$. The diffusion coefficient $\nu + \tilde{\nu} f_n$ is everywhere positive as shown in Fig. 2. A negative diffusion coefficient first occurs with $c_{n1} \approx 16.46$, which limits the magnitude of this parameter.

Although Eq. 20 places a constraint on the combined production and destruction terms, we define individually production to be positive and destruction to be negative,

$$P_n = c_{b1} (1 - c_{t3}) \Omega \tilde{\nu}, \quad D_n = -c_{w1} \left[\frac{\tilde{\nu}}{d} \right]^2$$

Note that P_n is defined in terms of vorticity Ω rather than modified vorticity $\widetilde{\Omega}$ as in the positive model. Also note the sign change on D_n compared to the positive model.

B APPENDIX: FIELD INVERSION

B.1 FIELD INVERSION RESULTS FOR THE SELECTED CASES

Reynolds number	Mach number	Angle of attack (degrees)	Chosen λ	Initial $\mathcal{L}$	Relative regularization
6,360,970	0.74208	4.4563	2.00E-11	1.14E-08	1.75E-03
8,787,960	0.72089	5.6690	1.00E-11	8.41E-09	1.19E-03
10,939,600	0.72401	5.6541	1.00E-10	7.95E-09	1.26E-02
13,181,400	0.72418	5.6504	1.00E-11	4.42E-09	2.26E-03
15,323,800	0.72355	5.1450	1.00E-10	7.85E-09	1.27E-02

Table B.1: Summary: Selected inversion flow cases used for Machine Learning data and their selected Tikhonov regularization values

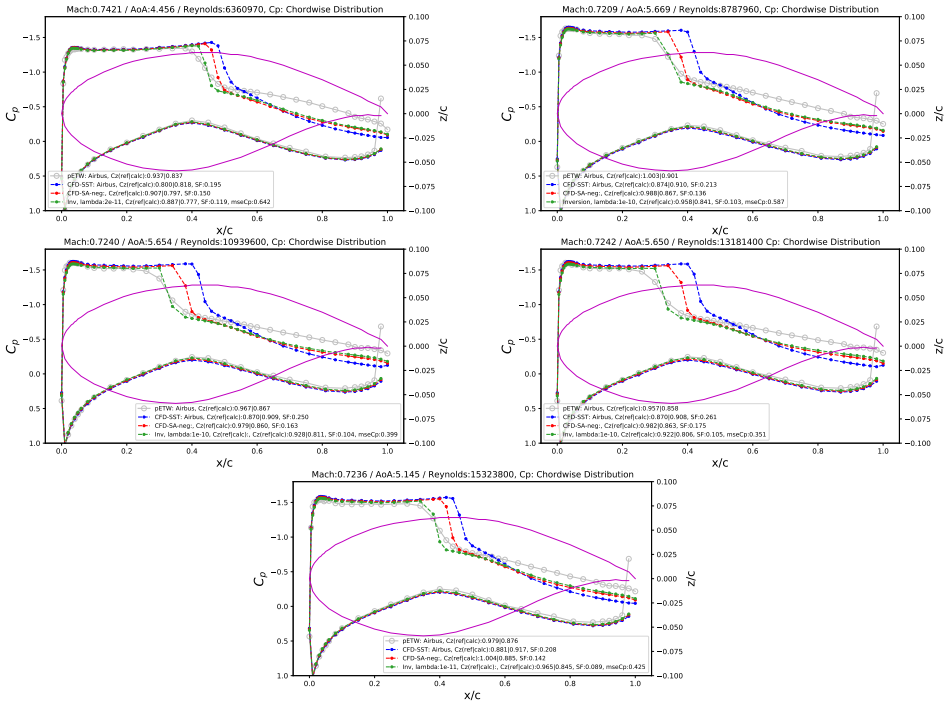
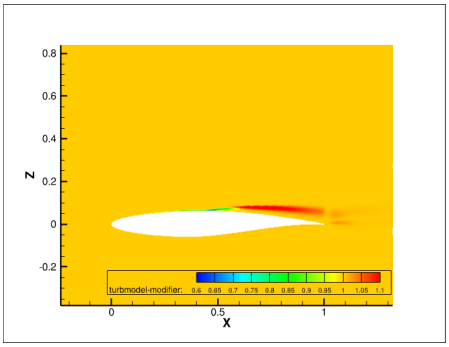
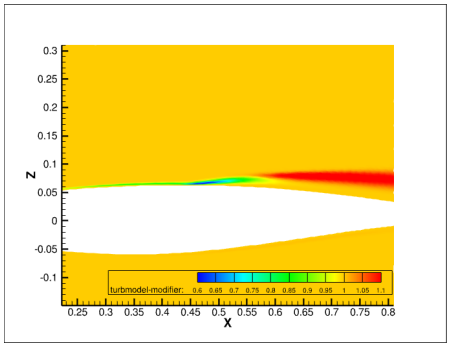


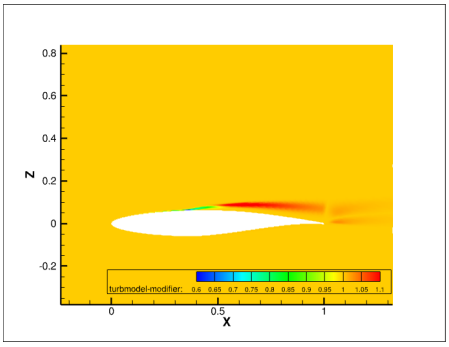
Figure B.1: C_p distribution for the selected inversion solution at flow conditions reported in Table B.1. The flow conditions are reported on the top of the subfigures. The last sensor on the lower surface of the airfoil in the pETW results (Grey line) is faulty, and results were not used for calculation of 'Cz' and 'mseCP'.
Red line: Baseline solution with SA-neg, Blue line: Baseline solution with 2-eqn SST model, Grey line: pETW wind tunnel result, Green line: Inversion solution. 'Cz': Lift coefficient, 'ref'= reference output from TAU solver, 'calc' = calculated output from a user-defined function, 'SF' = similarity factor with respect to wind tunnel results (pETW), 'mseCP' = normalized loss function value for the inversion problem.



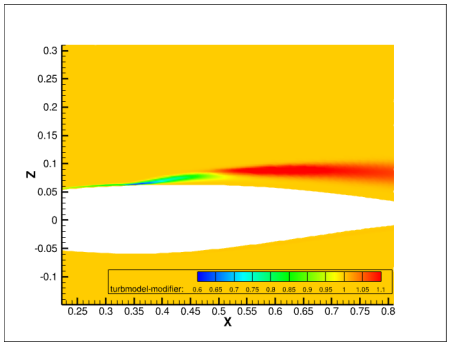
(a) Full field: Re = 6,360,970



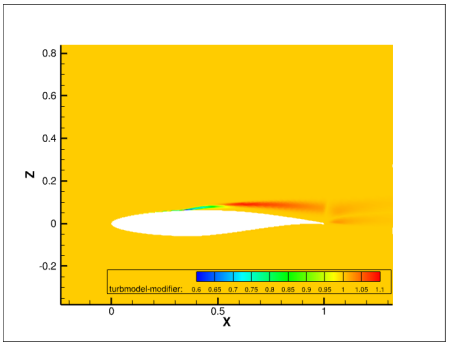
(b) Shock foot: Re = 6,360,970



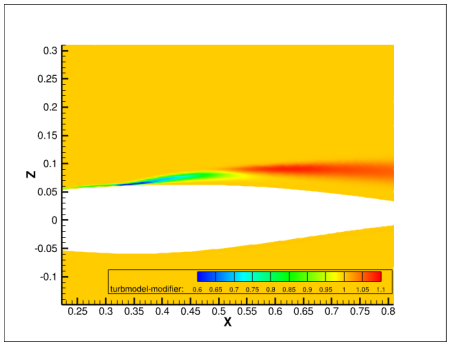
(c) Full field: Re = 8,787,960



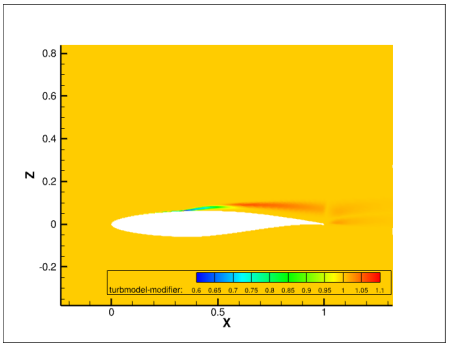
(d) Shock foot: Re = 8,787,960



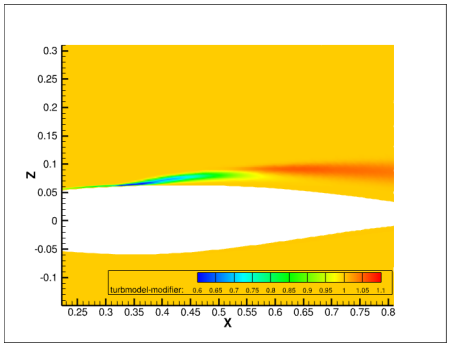
(e) Full field: Re = 10,939,600



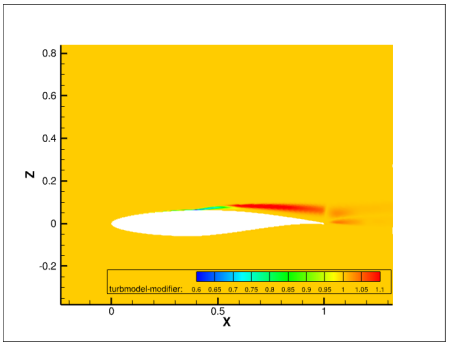
(f) Shock foot: Re = 10,939,600



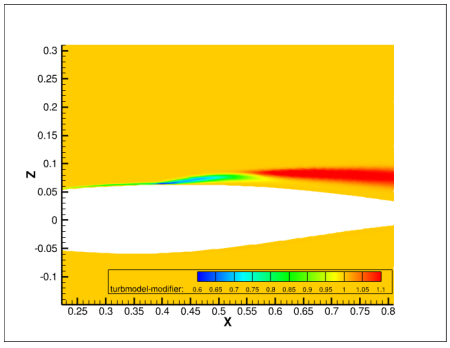
(g) Full field: Re = 13,181,400



(h) Shock foot: Re = 13,181,400

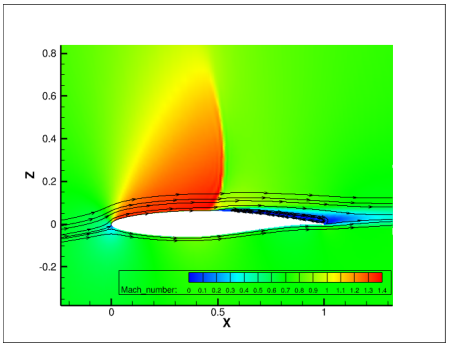


(i) Full field: Re = 15,323,800

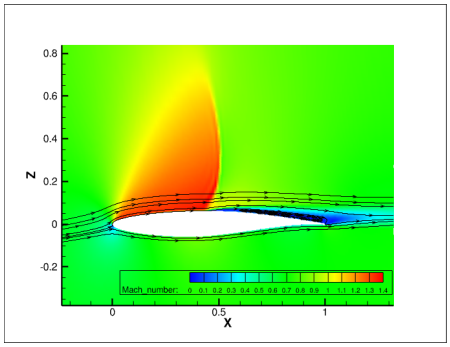


(j) Shock foot: Re = 15,323,800

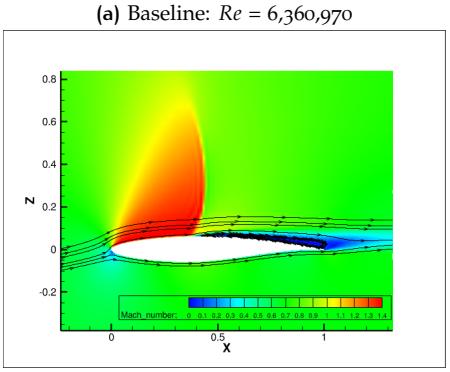
Figure B.2: $\beta(x)$ field for the selected inversion solutions at flow conditions provided in Table B.1. Only Reynolds number has been reported in the subfigure captions.



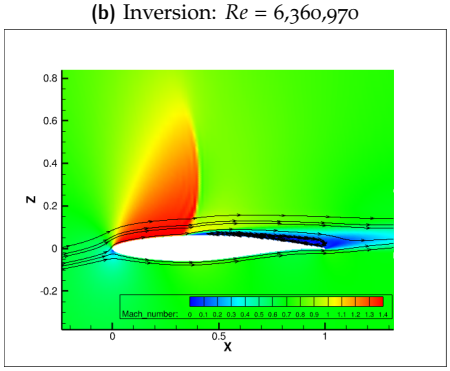
(a) Baseline: Re = 6,360,970



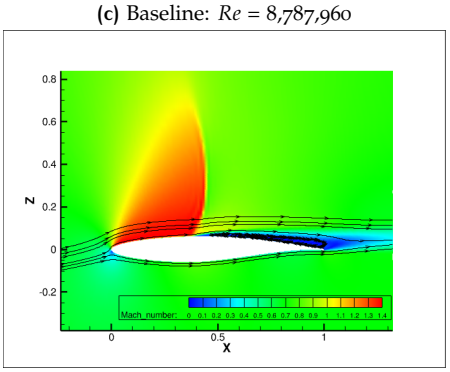
(b) Inversion: Re = 6,360,970



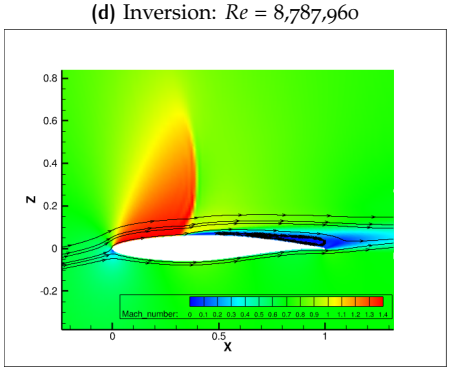
(c) Baseline: Re = 8,787,960



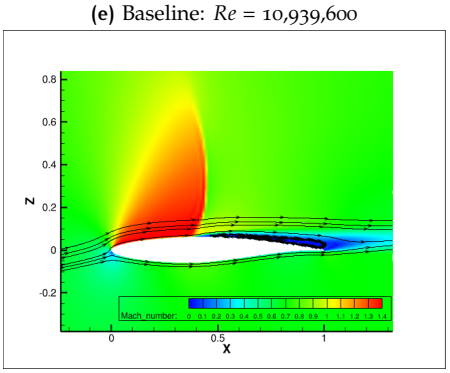
(d) Inversion: Re = 8,787,960



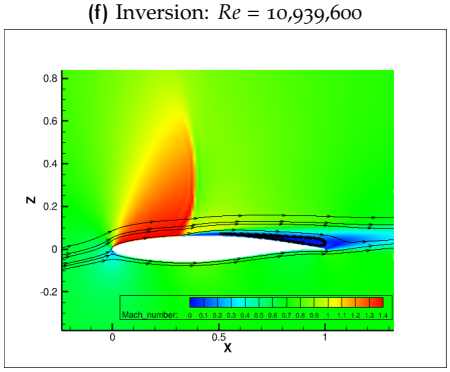
(e) Baseline: Re = 10,939,600



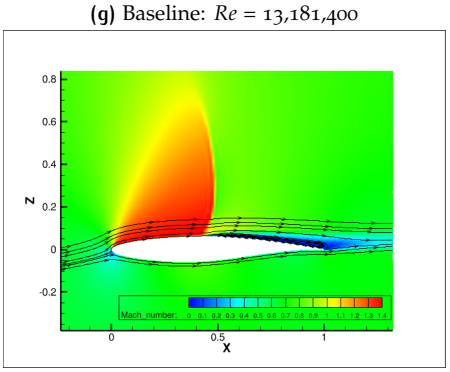
(f) Inversion: Re = 10,939,600



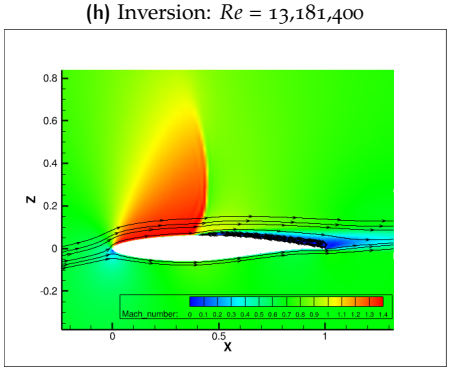
(g) Baseline: Re = 13,181,400



(h) Inversion: Re = 13,181,400



(i) Baseline: Re = 15,323,800



(j) Inversion: Re = 15,323,800

Figure B.3: Comparsion of the Mach number contours for the baseline vs. inversion solution at flow conditions reported in Table B.1. Only Reynolds number has been reported in the subfigure captions.

C APPENDIX: FEATURE ENGINEERING

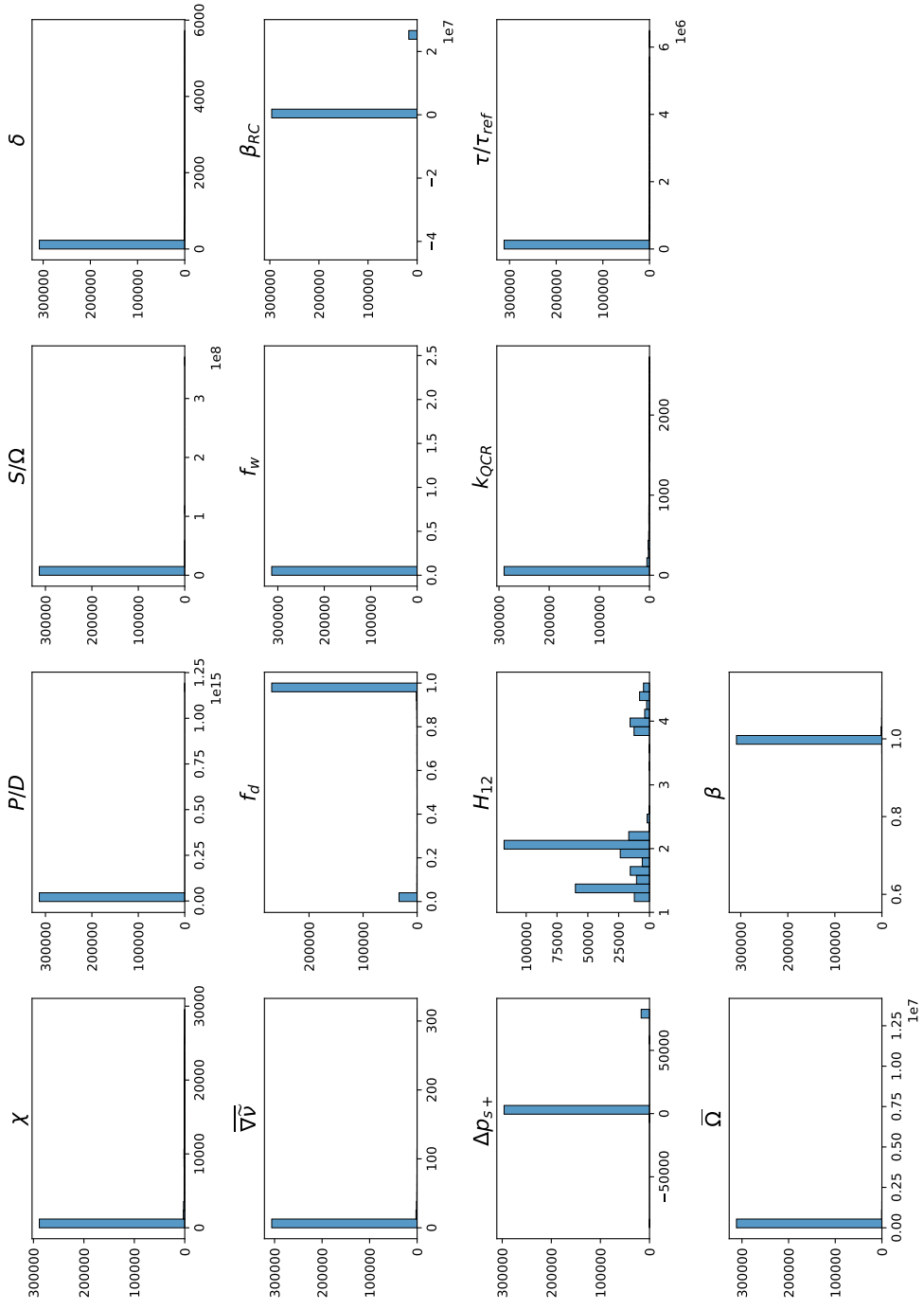


Figure C.1: Histogram for all features without any transformations for the feature data from the inversion solution at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960.

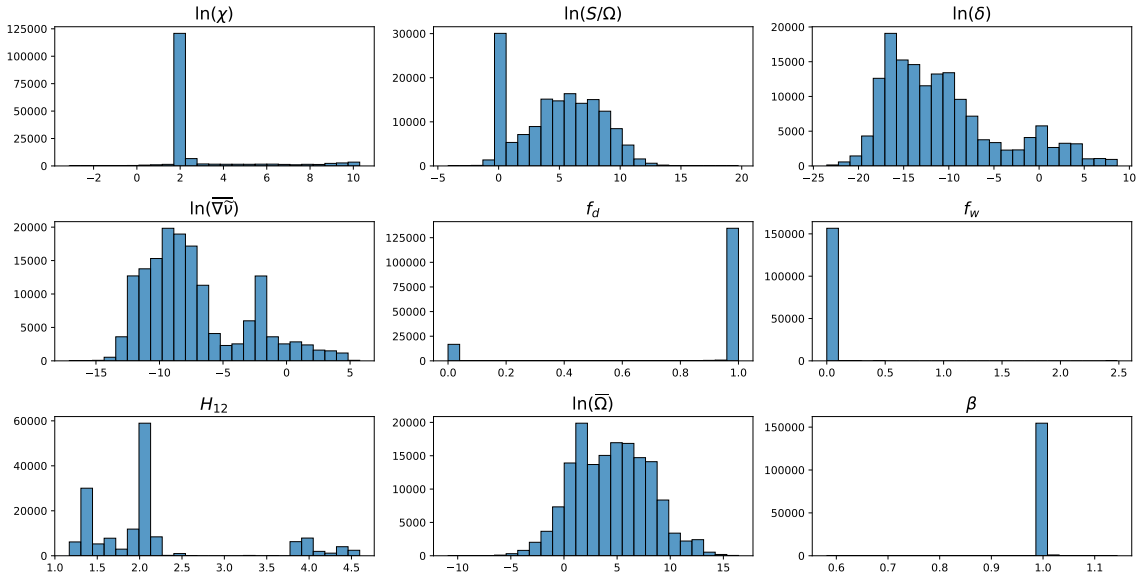


Figure C.2: Histogram of the remaining input features after applying the $\ln(q_\beta)$ transformation for the feature data from the inversion solution at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960

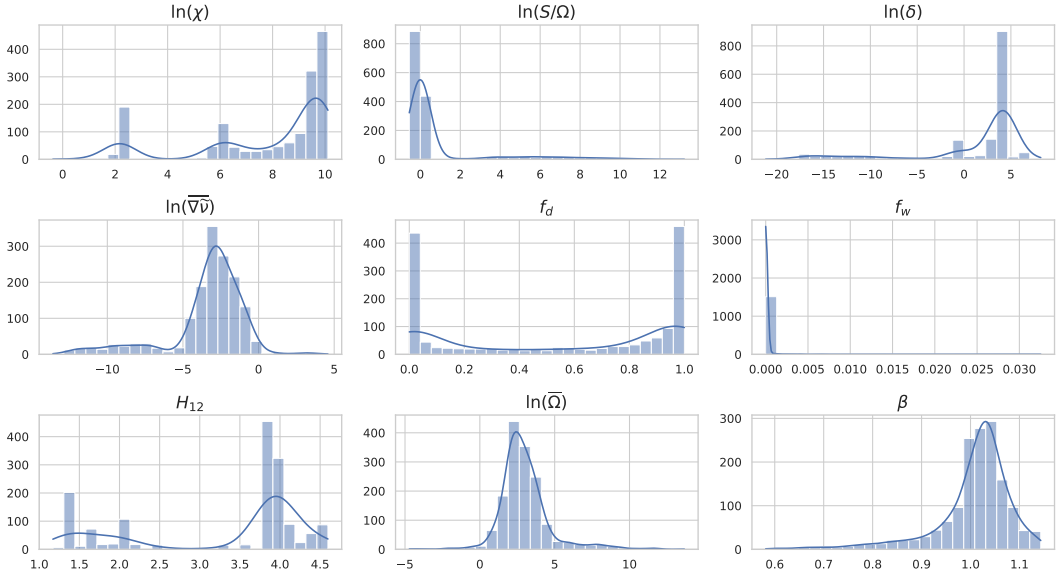


Figure C.3: Histogram for the input features in the training data generated after applying the filtering protocol for all $\beta \notin [0.98, 1.02]$ from the inversion solution at flow conditions M = 0.7209, AoA = 5.669, Re = 8787960.

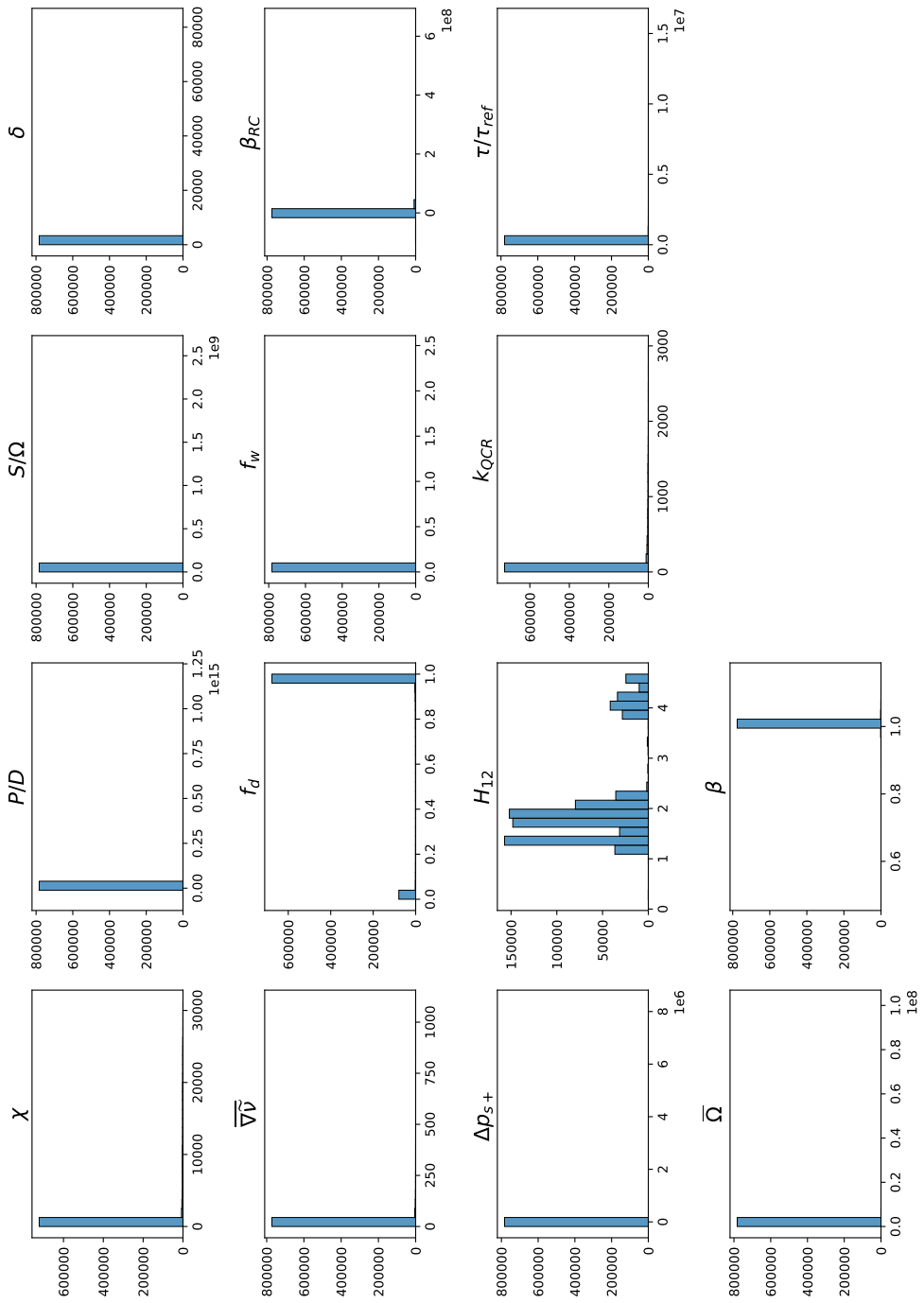


Figure C.4: Histogram for all features without any transformations for the feature data from all the inversion solutions at flow conditions in Table 5.1

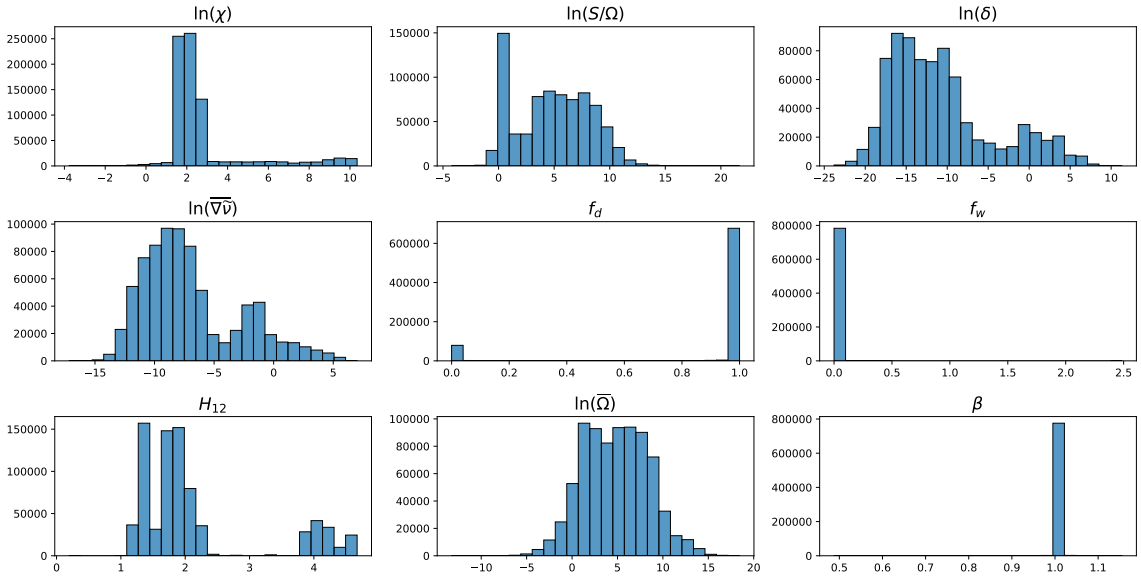


Figure C.5: Histogram of the remaining input features after applying the $\ln(q_\beta)$ transformation for the feature data from all the inversion solutions at flow conditions in Table 5.1

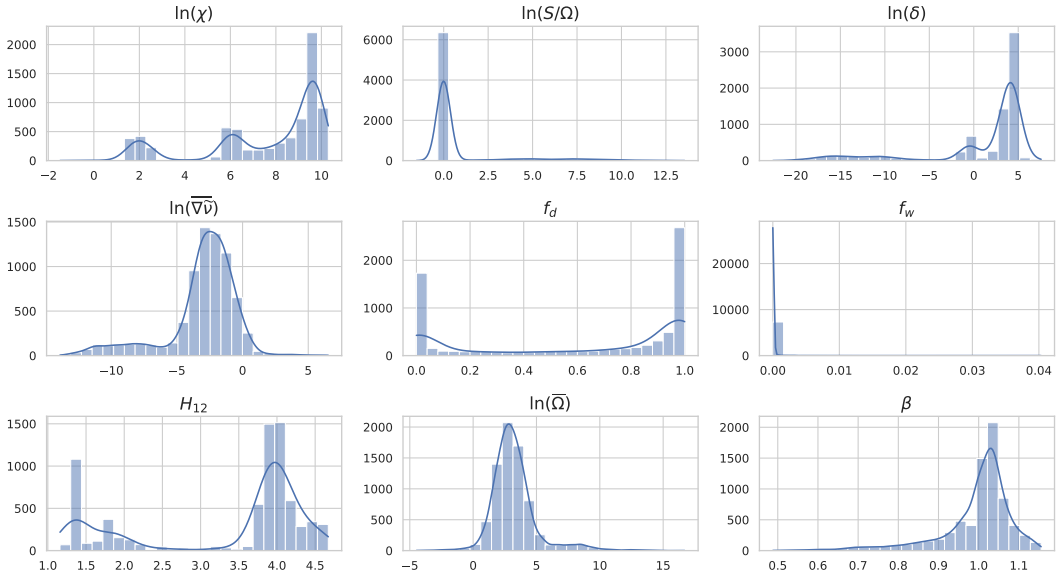


Figure C.6: Histogram for the input features in the training data generated after applying the filtering protocol for all $\beta \notin [0.98, 1.02]$ from all the inversion solutions at flow conditions in Table 5.1

D APPENDIX: MACHINE LEARNING

The neural networks used in this study were trained on flow features generated from two datasets:

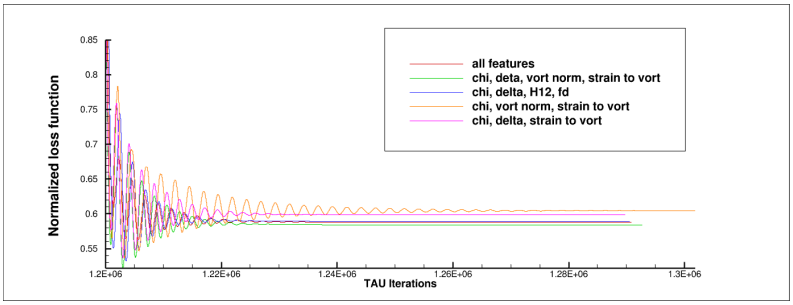
- Re = 9 million dataset: Training data generated from inversion solution at M = 0.7209, AoA = 5.669, Re = 8787960.
- Full dataset: Training data generated from all inversion solutions detailed in Table B.1 collated in one database.

The following sections display the results of connecting the neural networks trained on these datasets to the TAU flow solver.

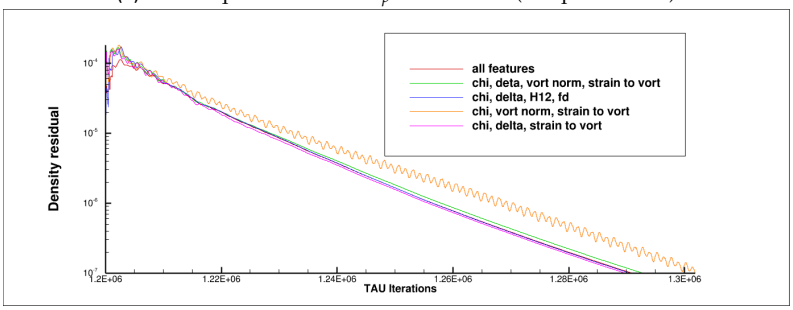
D.1 TRAINING ON RE = 9 MILLION DATASET

D.1.1 Testing on training data flow cases

M = ***0.7209***, AoA = ***5.669***, Re = ***8787960***



(a) Mean squared error of C_p distribution (wrt pETW data)



(b) Density residual

Figure D.1: Residual variation with the flow solution in TAU after augmenting the baseline SA-neg model with the trained neural network at every 500 iterations.

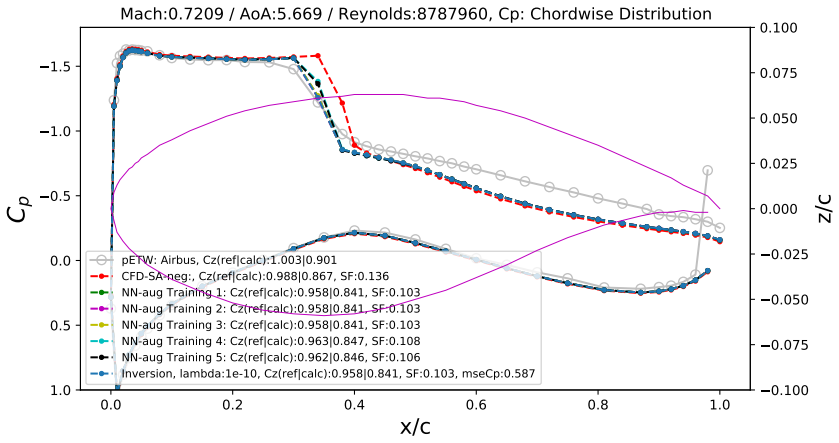
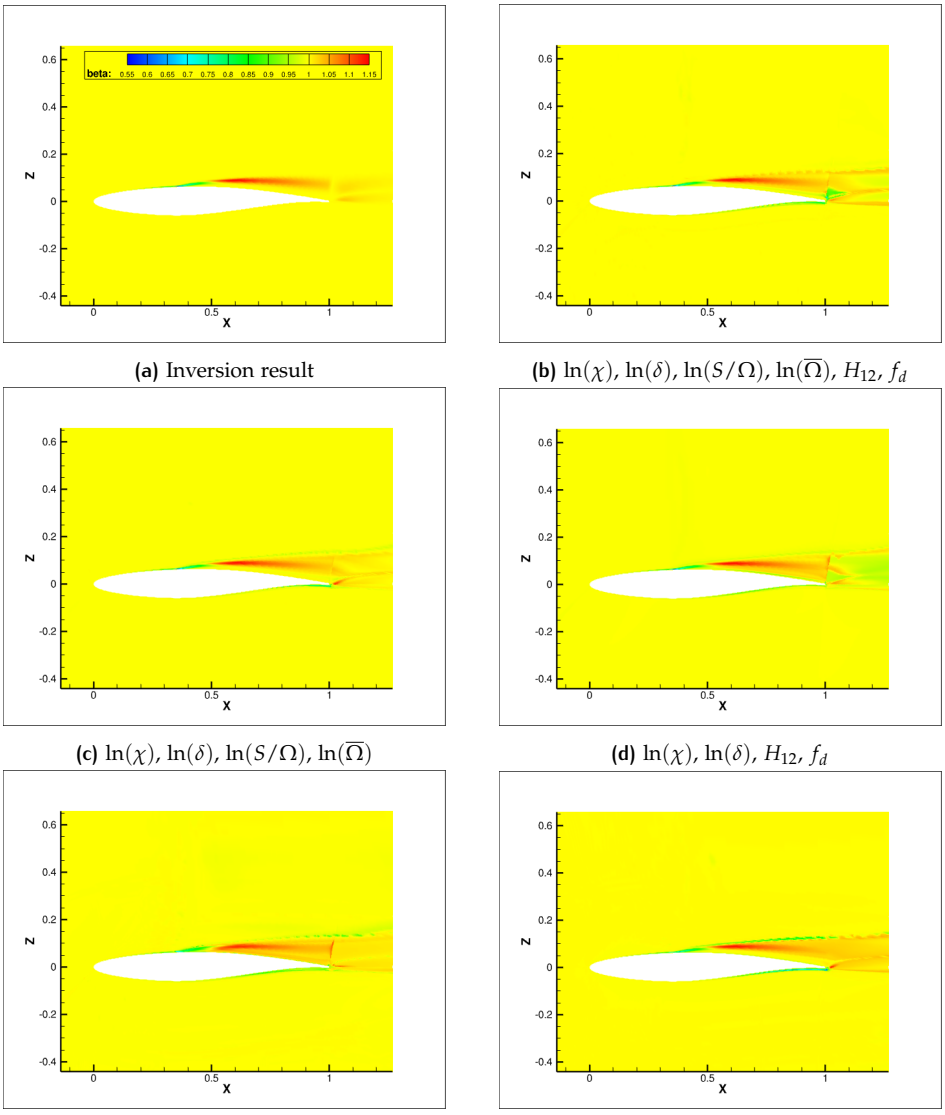


Figure D.2: C_p distribution for the NN-augmented TAU solutions. Training 1-5 are indicated in Section 7.2.1.



(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$, H_{12}, f_d

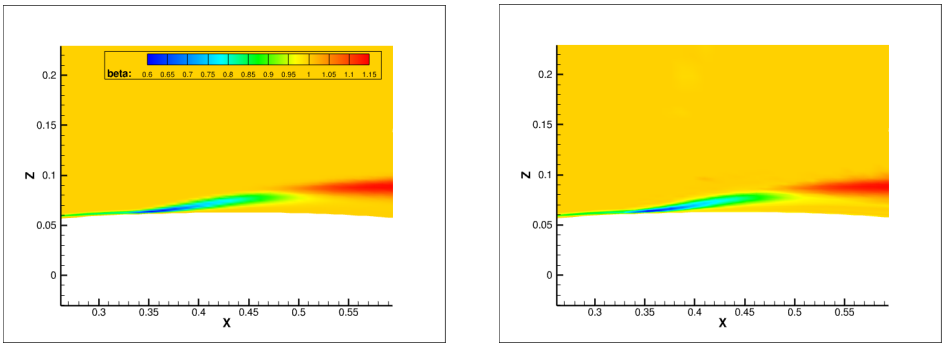
(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, H_{12}, f_d

(e) $\ln(\chi)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$

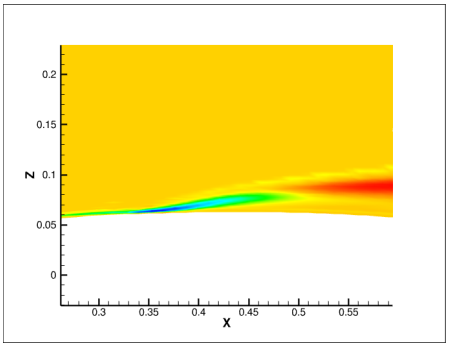
(f) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$

Figure D.3: Results after connecting the trained neural networks with different features to TAU flow solver.

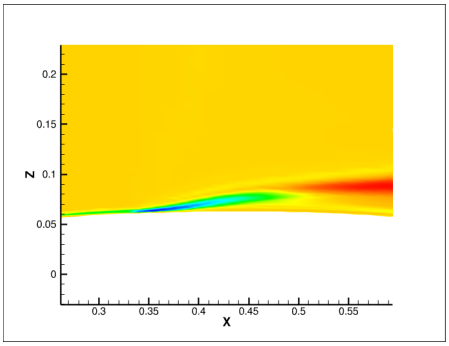


(a) Inversion result

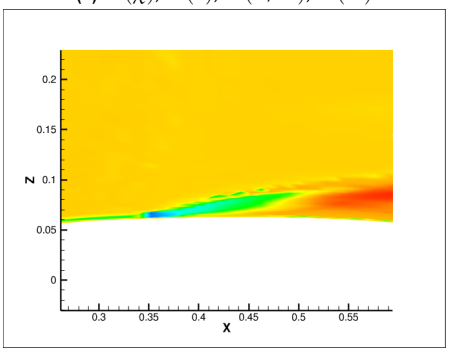
(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$, H_{12}, f_d



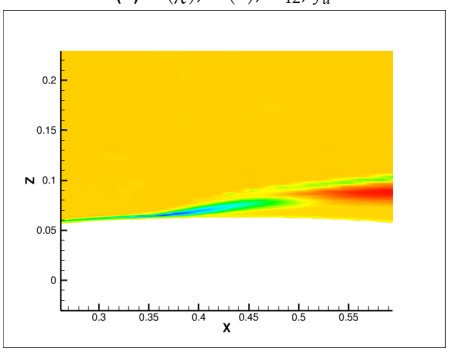
(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$



(d) $\ln(\chi)$, $\ln(\delta)$, H_{12}, f_d



(e) $\ln(\chi)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$

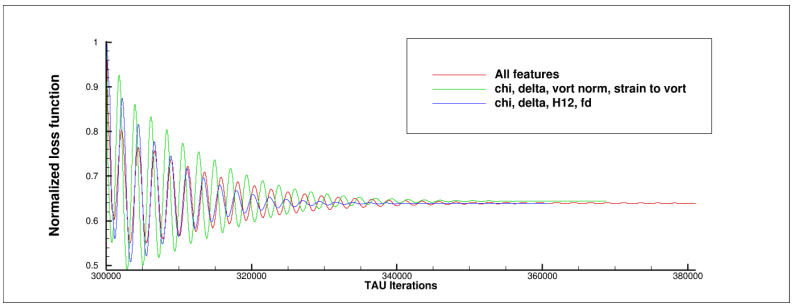


(f) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$

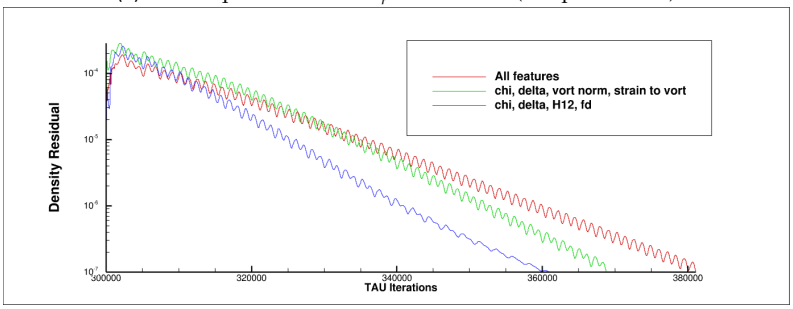
Figure D.4: Results near shock foot after connecting the trained neural networks with different features to TAU flow solver.

D.1.2 Testing on unknown flow cases

$M = 0.7421$, $AoA = 4.456$, $Re = 6360970$



(a) Mean squared error of C_p distribution (wrt pETW data)



(b) Density residual

Figure D.5: Residual variation with the flow solution in TAU after augmenting the baseline SA-neg model with the trained neural network at every 500 iterations.

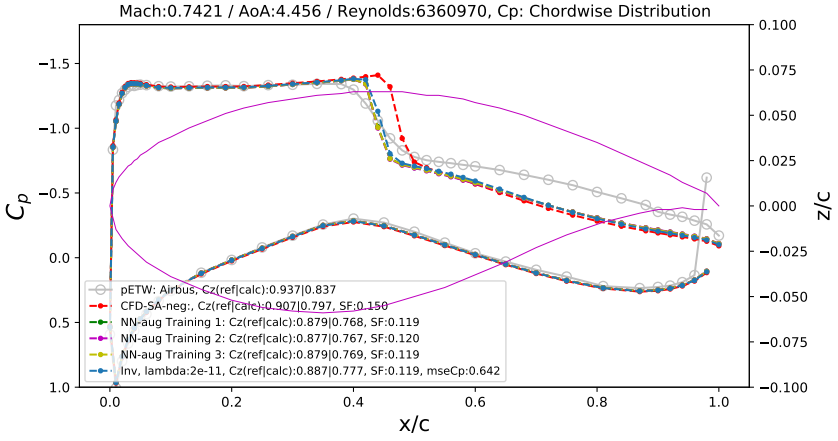
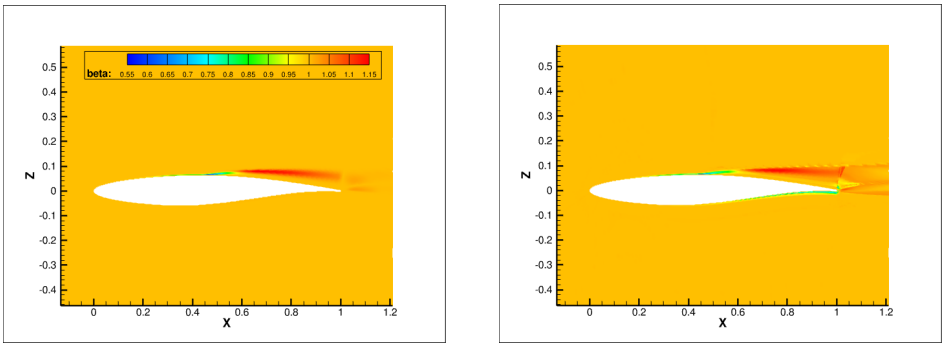


Figure D.6: C_p distribution for the NN-augmented TAU solutions. Training 1-3 are indicated in Section 7.2.1.



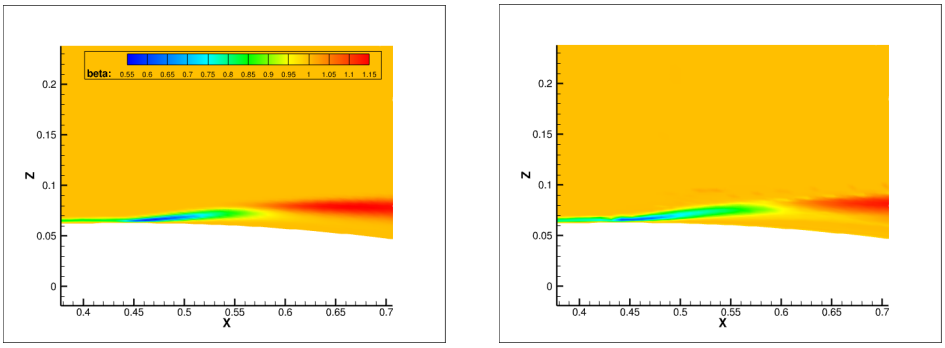
(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$, H_{12}, f_d

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, H_{12}, f_d

Figure D.7: Results after connecting the trained neural networks with different features to TAU flow solver.



(a) Inversion result

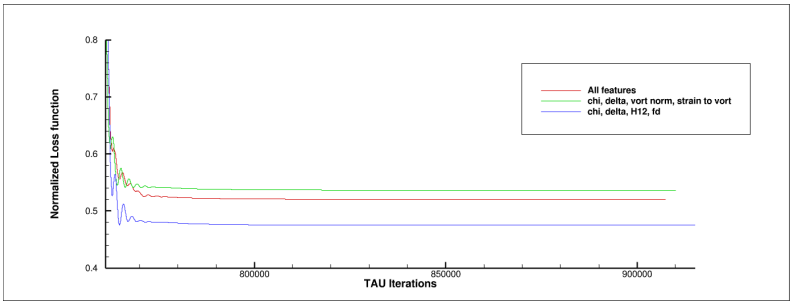
(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$, H_{12}, f_d

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$

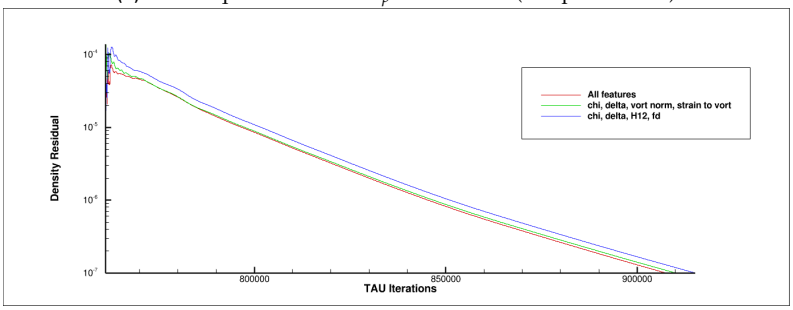
(d) $\ln(\chi)$, $\ln(\delta)$, H_{12}, f_d

Figure D.8: Results near shock foot after connecting the trained neural networks with different features to TAU flow solver.

$M = 0.7235$, $AoA = 5.145$, $Re = 15323800$



(a) Mean squared error of C_p distribution (wrt pETW data)



(b) Density residual

Figure D.9: Residual variation with the flow solution in TAU after augmenting the baseline SA-neg model with the trained neural network at every 500 iterations.

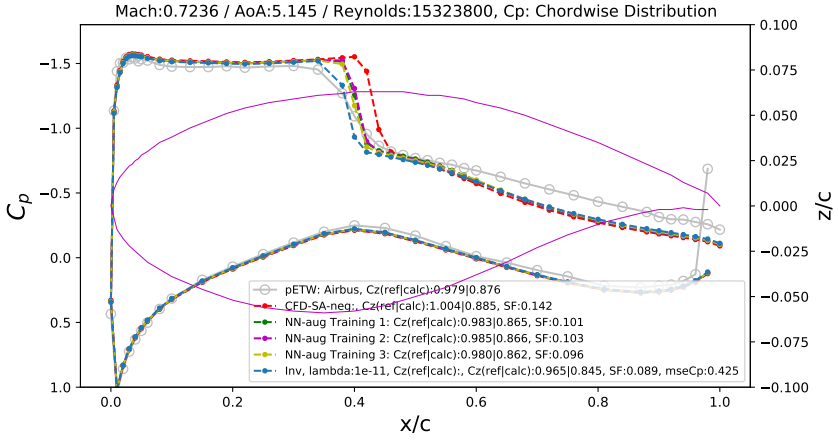
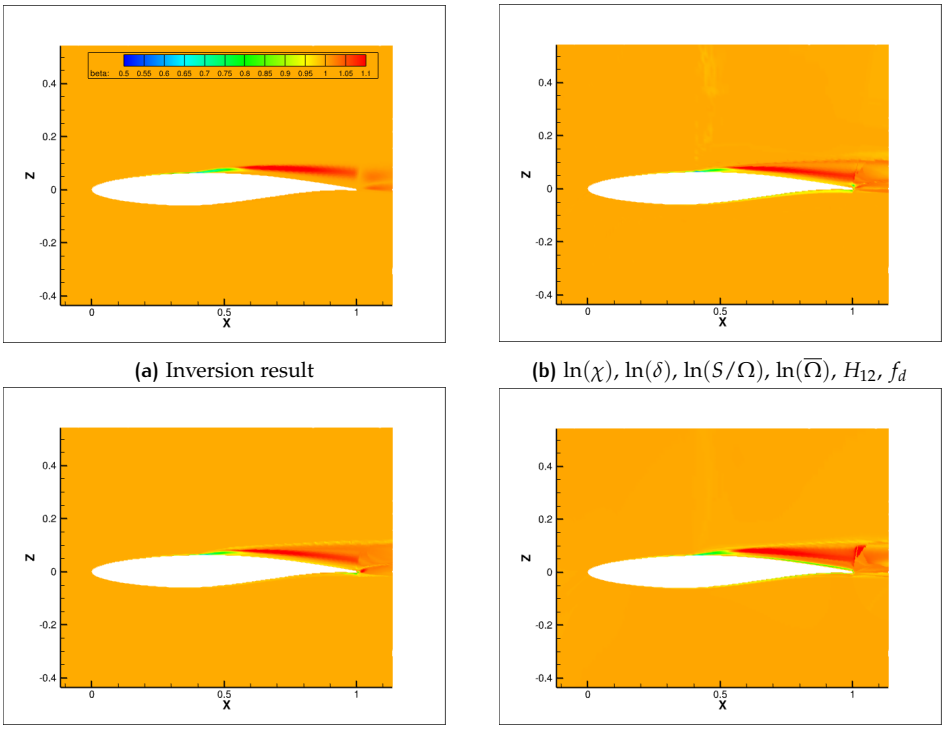


Figure D.10: C_p distribution for the NN-augmented TAU solutions. Training 1-3 are indicated in Section 7.2.1.



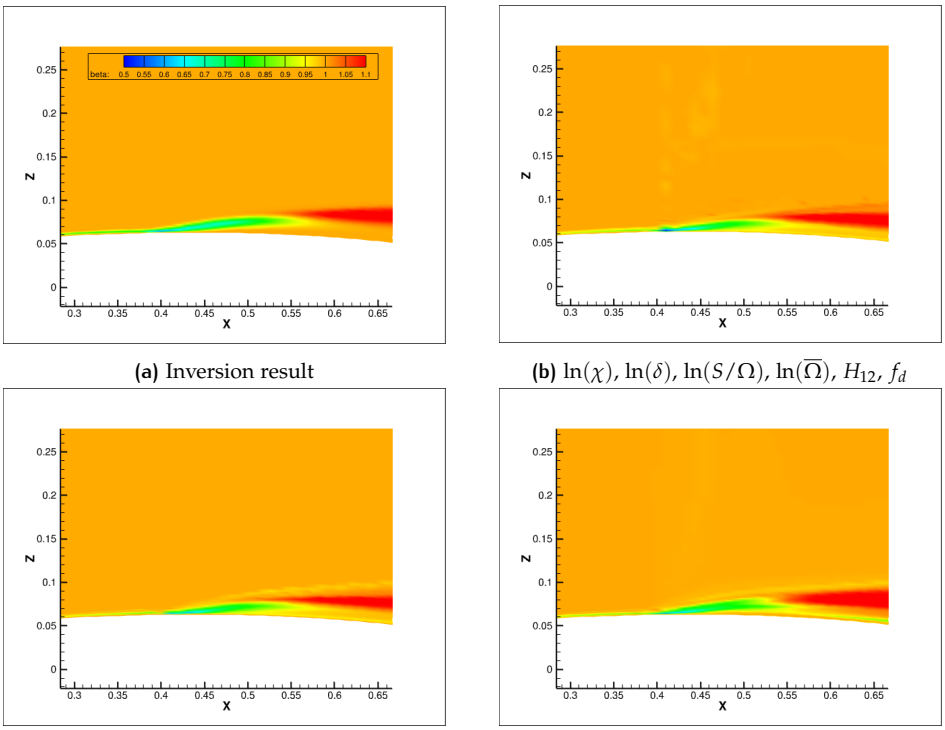
(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$, H_{12}, f_d

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, H_{12}, f_d

Figure D.11: Results after connecting the trained neural networks with different features to TAU flow solver.



(a) Inversion result

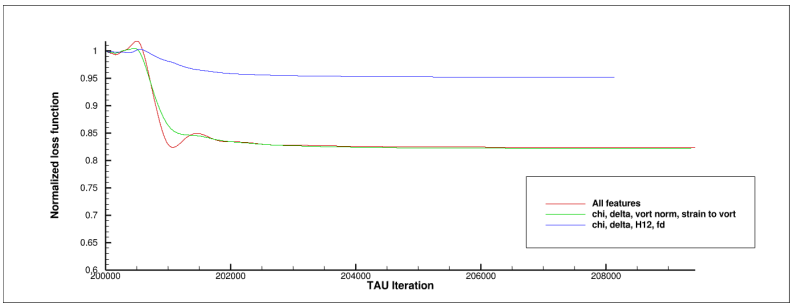
(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$, H_{12}, f_d

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$

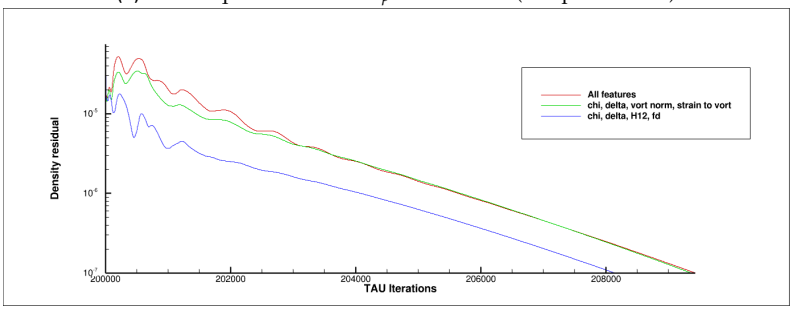
(d) $\ln(\chi)$, $\ln(\delta)$, H_{12}, f_d

Figure D.12: Results near shock foot after connecting the trained neural networks with different features to TAU flow solver.

$M = 0.7173$, $AoA = 2.604$, $Re = 2680960$



(a) Mean squared error of C_p distribution (wrt pETW data)



(b) Density residual

Figure D.13: Residuals: Neural network augmentation at every 500 iterations.

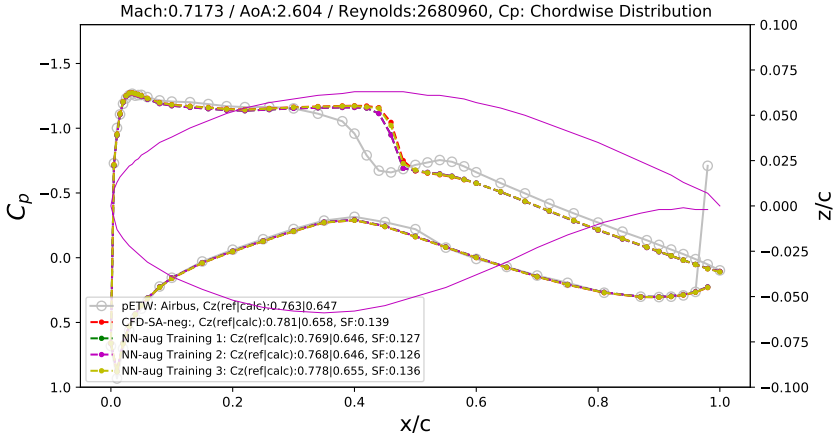
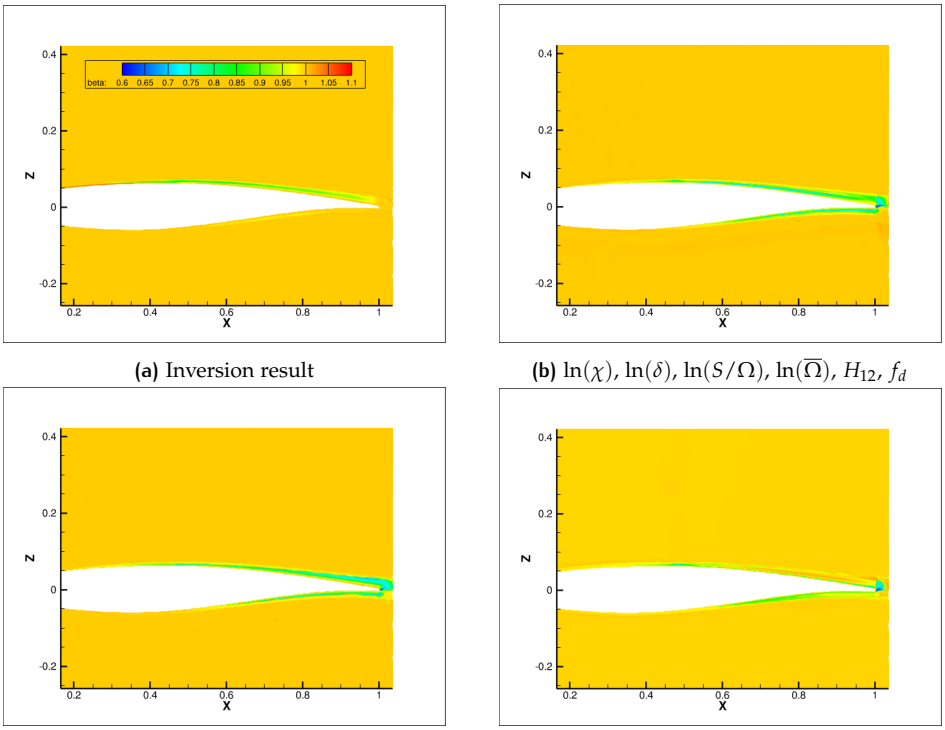


Figure D.14: C_p distribution for the NN-augmented TAU solutions. Training 1-3 are indicated in Section 7.2.1.



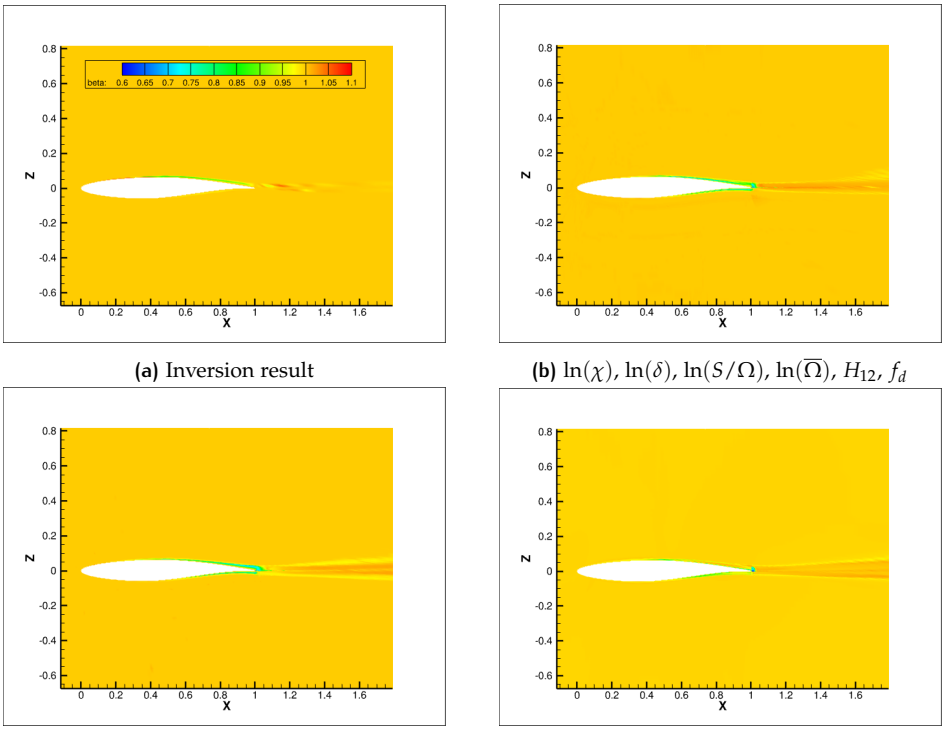
(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$, H_{12}, f_d

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, H_{12}, f_d

Figure D.15: Results after connecting the trained neural networks with different features to TAU flow solver.



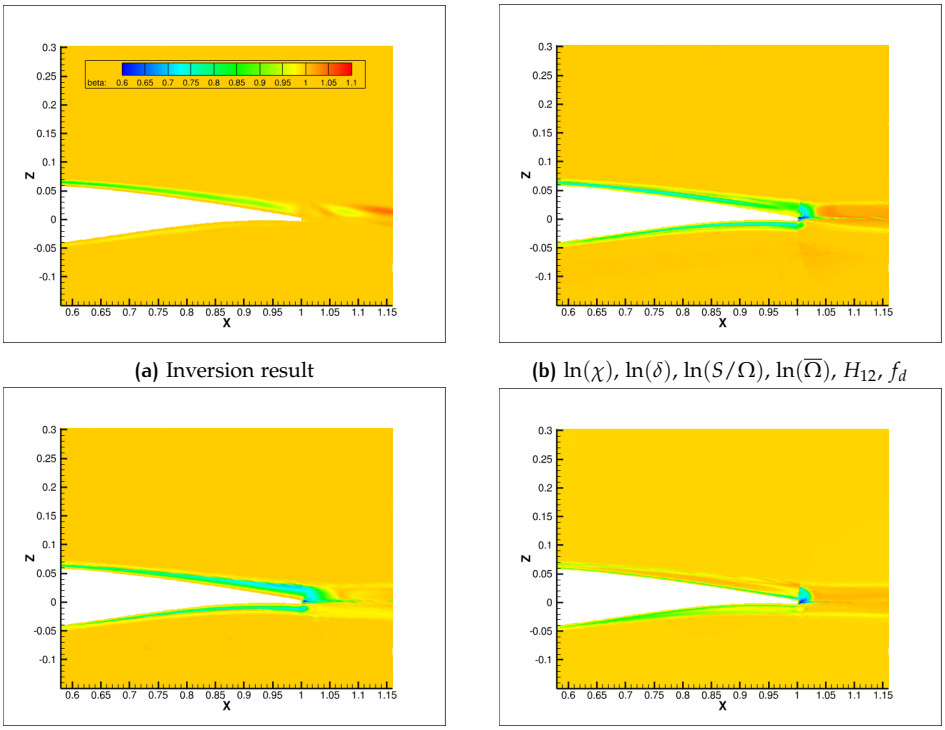
(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$, H_{12}, f_d

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, H_{12}, f_d

Figure D.16: Far field results after connecting the trained neural networks with different features to TAU flow solver.



(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$, H_{12}, f_d

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(S/\Omega)$, $\ln(\overline{\Omega})$

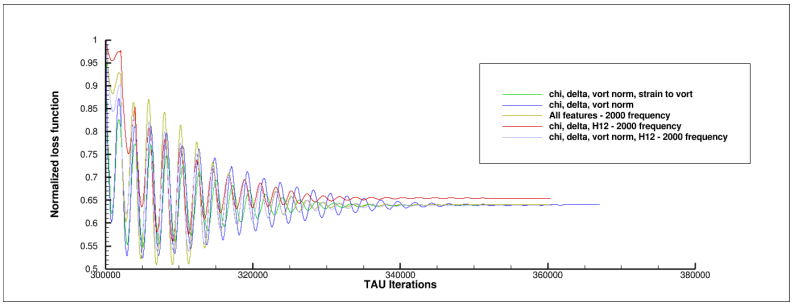
(d) $\ln(\chi)$, $\ln(\delta)$, H_{12}, f_d

Figure D.17: Results near trailing edge after connecting the trained neural networks with different features to TAU flow solver.

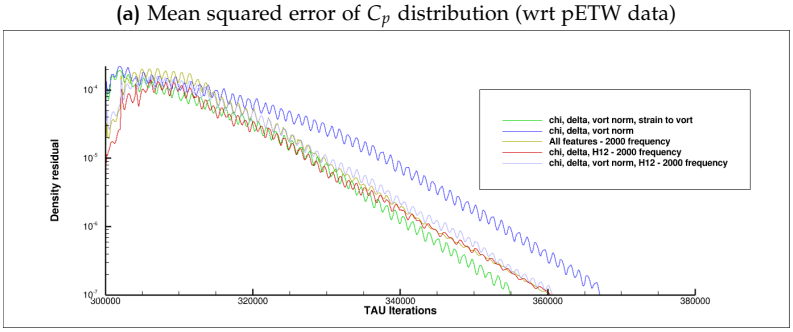
D.2 TRAINING ON FULL DATASET

D.2.1 Testing on training data flow cases

$M = 0.7421$, $AoA = 4.456$, $Re = 6360970$

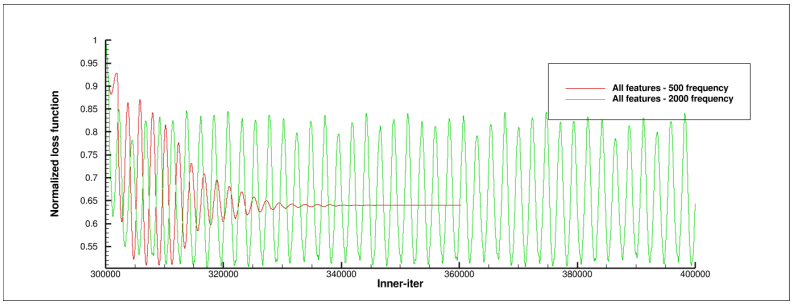


(a) Mean squared error of C_p distribution (wrt pETW data)

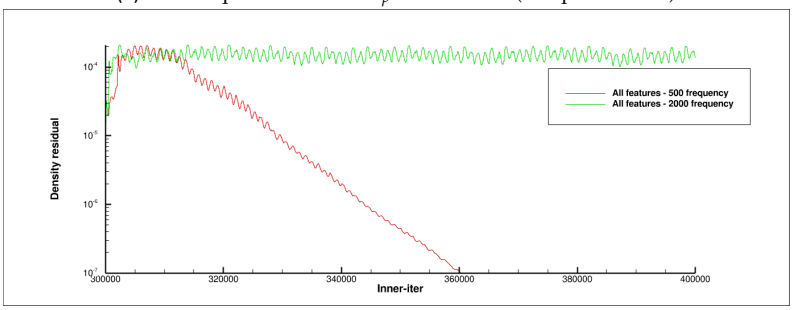


(b) Density residual

Figure D.18: Residual variation with the flow solution in TAU after augmenting the baseline SA-neg model with the trained neural network.

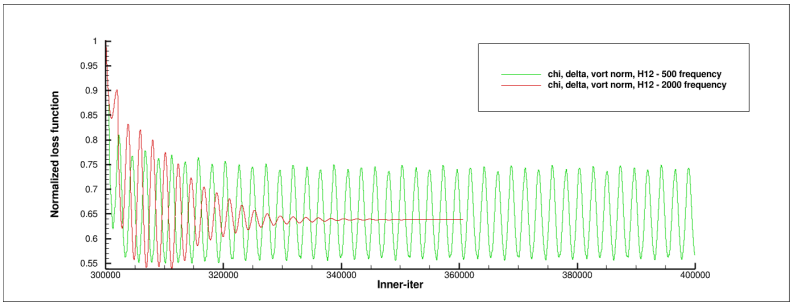


(a) Mean squared error of C_p distribution (wrt pETW data)

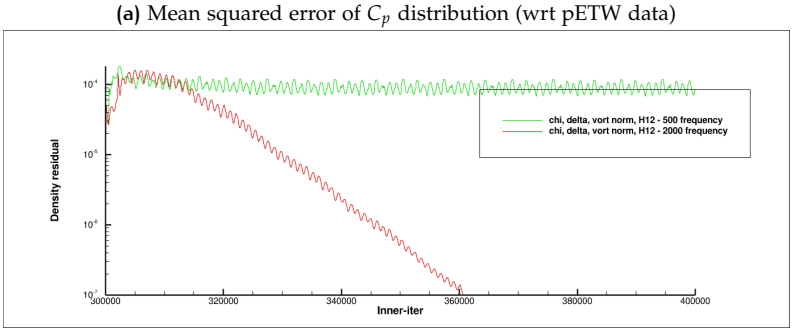


(b) Density residual

Figure D.19: Residual comparison on changing the frequency of plugging the trained neural network. Features used: $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

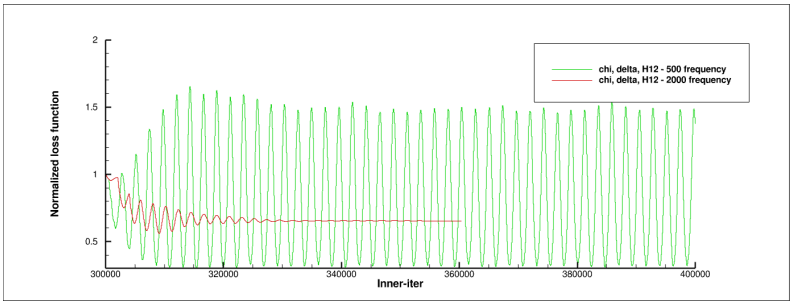


(a) Mean squared error of C_p distribution (wrt pETW data)

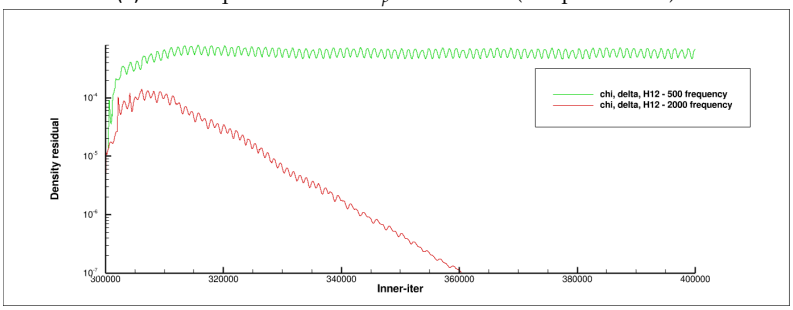


(b) Density residual

Figure D.20: Residual comparison on changing the frequency of plugging the trained neural network. Features used: $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}



(a) Mean squared error of C_p distribution (wrt pETW data)



(b) Density residual

Figure D.21: Residual comparison on changing the frequency of plugging the trained neural network. Features used: $\ln(\chi)$, $\ln(\delta)$, H_{12}

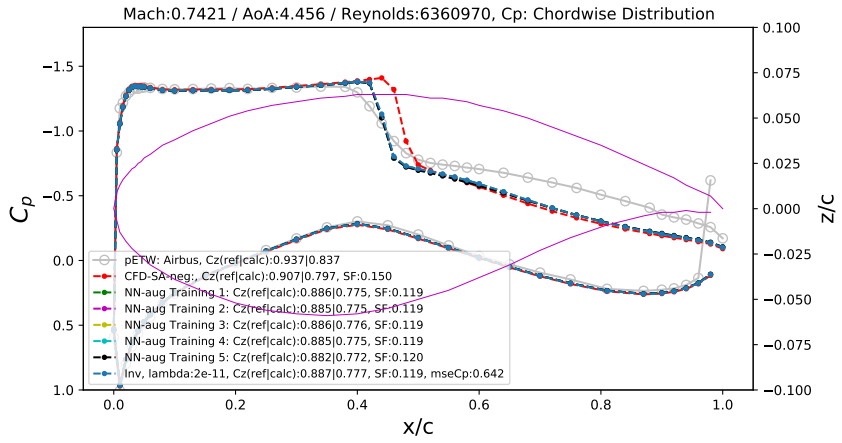
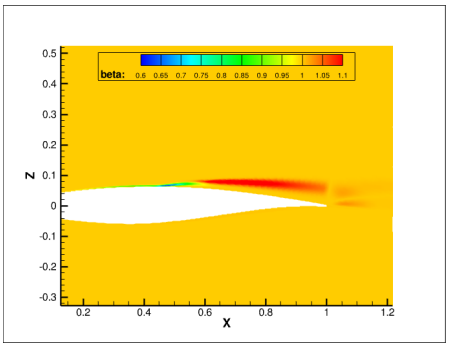
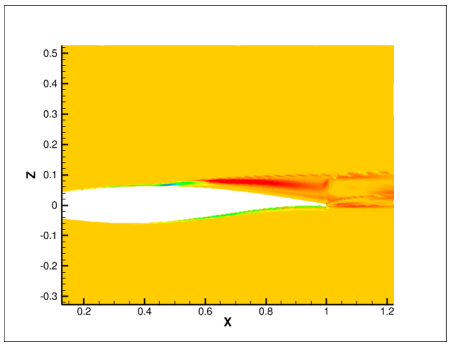


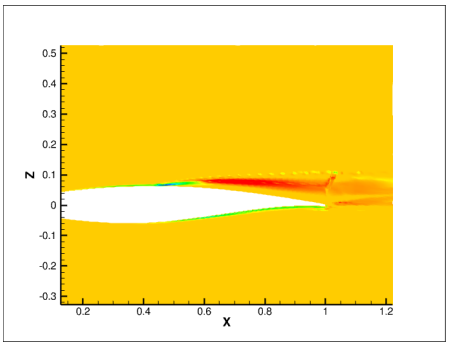
Figure D.22: C_p distribution for the NN-augmented TAU solutions. Training 1-5 are indicated in Section 7.2.2.



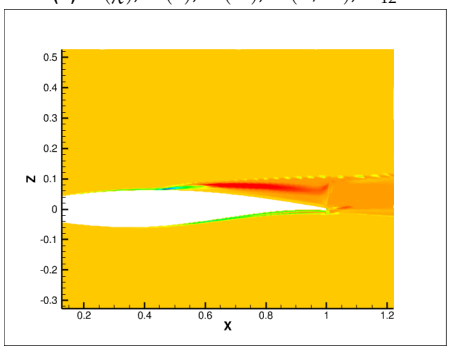
(a) Inversion result



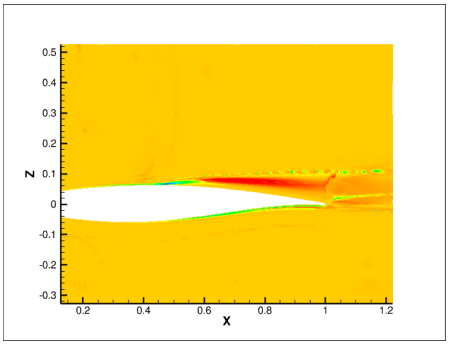
(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}



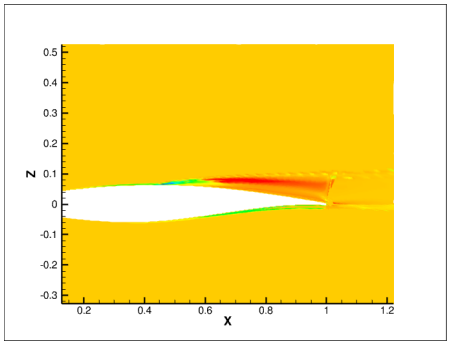
(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$



(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

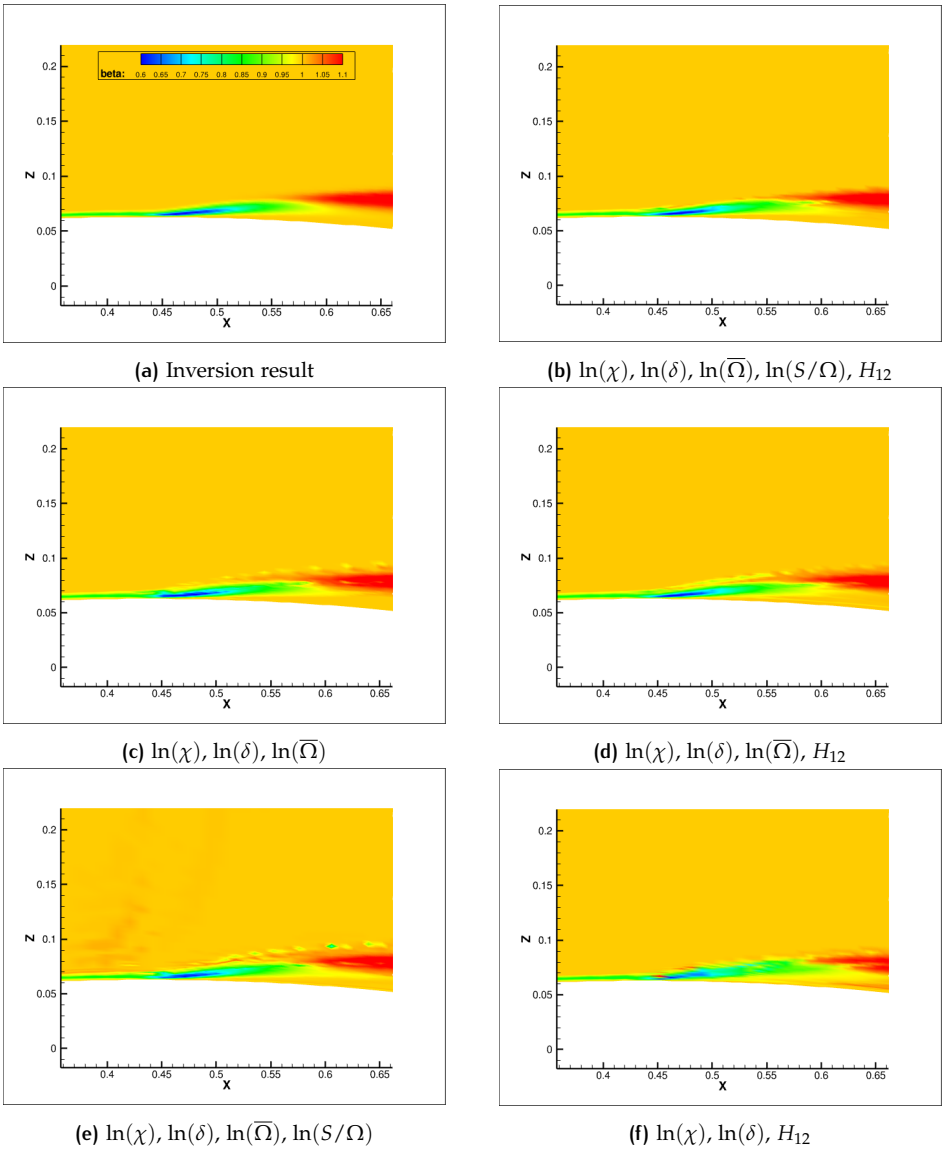


(e) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$



(f) $\ln(\chi)$, $\ln(\delta)$, H_{12}

Figure D.23: Results after connecting the trained neural networks with different features to TAU flow solver.



(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

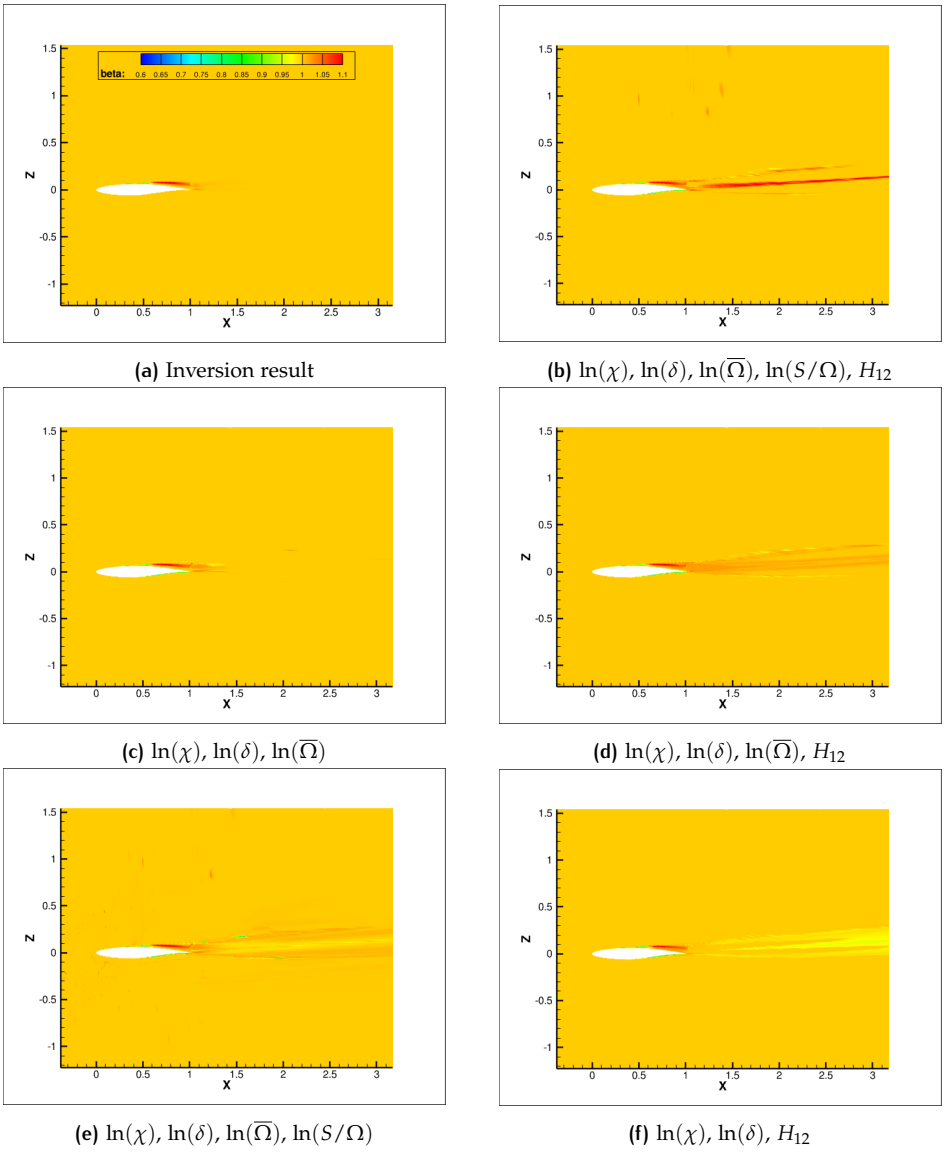
(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

(e) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

(f) $\ln(\chi)$, $\ln(\delta)$, H_{12}

Figure D.24: Results at shock foot after connecting the trained neural networks with different features to TAU flow solver.



(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

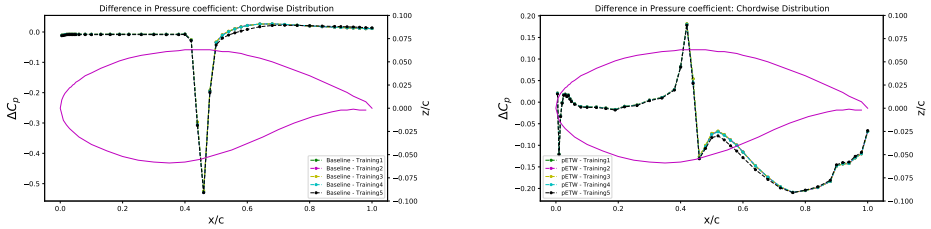
(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

(e) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

(f) $\ln(\chi)$, $\ln(\delta)$, H_{12}

Figure D.25: Results at shock foot after connecting the trained neural networks with different features to TAU flow solver.

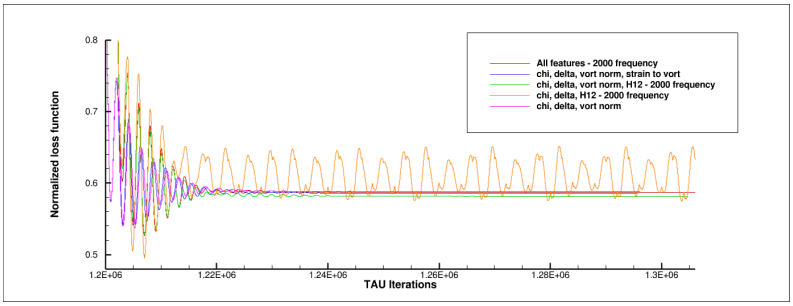


(a) Difference with Baseline solution

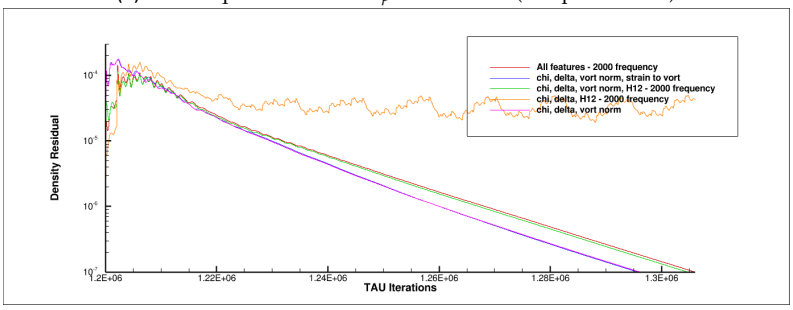
(b) Difference with pETW result

Figure D.26: ΔC_p distribution for the NN-augmented TAU solution. Training 1-5 are indicated in Section 7.2.2.

$M = 0.7209$, $AoA = 5.669$, $Re = 8787960$

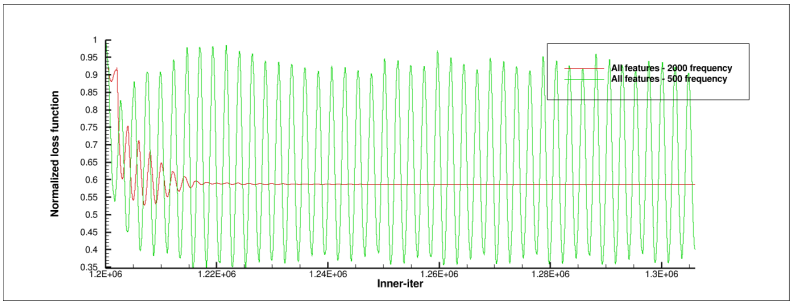


(a) Mean squared error of C_p distribution (wrt pETW data)

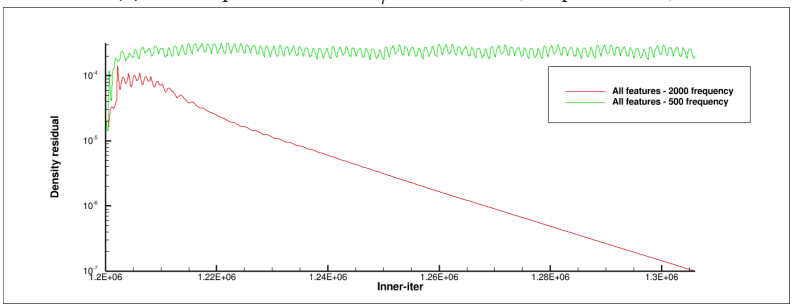


(b) Density residual

Figure D.27: Residual variation with the flow solution in TAU after augmenting the baseline SA-neg model with the trained neural network.

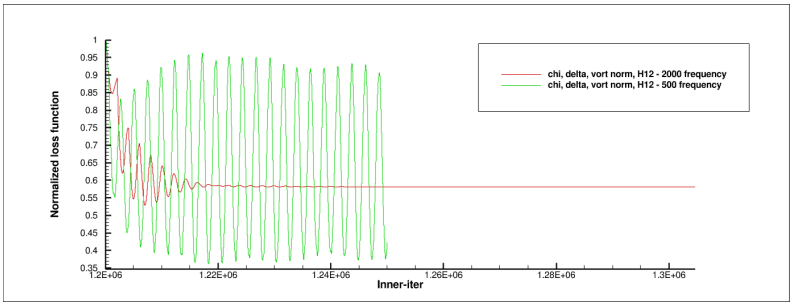


(a) Mean squared error of C_p distribution (wrt pETW data)

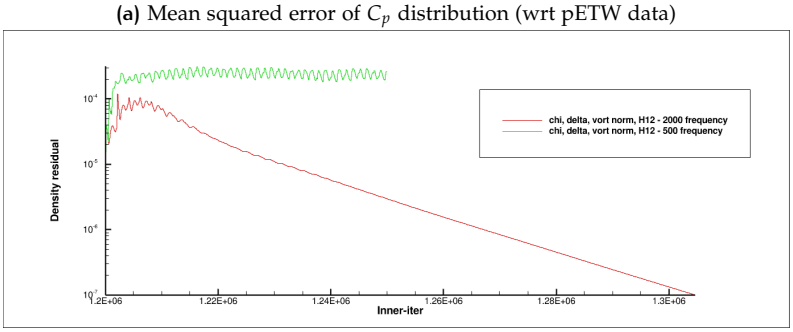


(b) Density residual

Figure D.28: Residual comparison on changing the frequency of plugging the trained neural network. Features used: $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

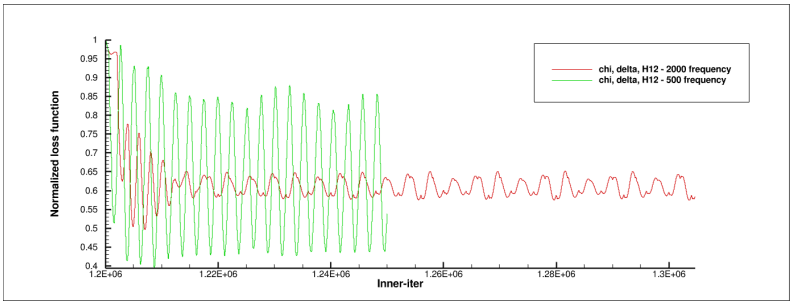


(a) Mean squared error of C_p distribution (wrt pETW data)

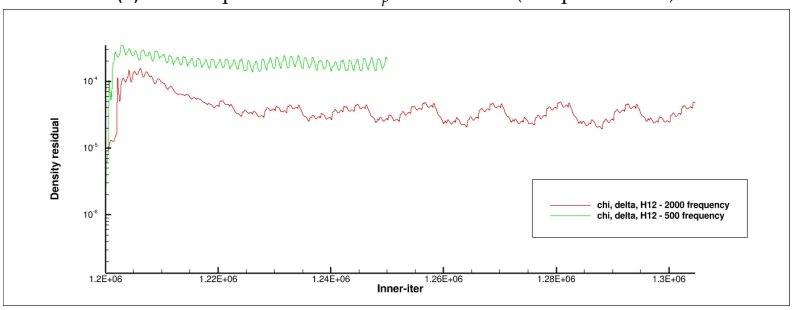


(b) Density residual

Figure D.29: Residual comparison on changing the frequency of plugging the trained neural network. Features used: $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}



(a) Mean squared error of C_p distribution (wrt pETW data)



(b) Density residual

Figure D.30: Residual comparison on changing the frequency of plugging the trained neural network. Features used: $\ln(\chi)$, $\ln(\delta)$, H_{12}

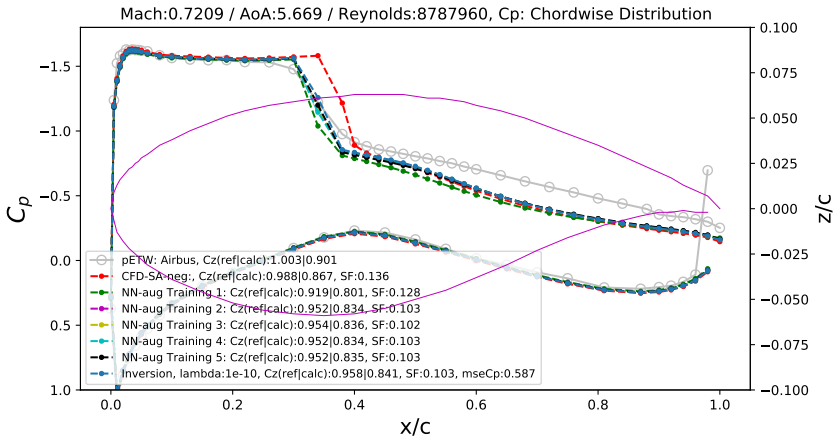
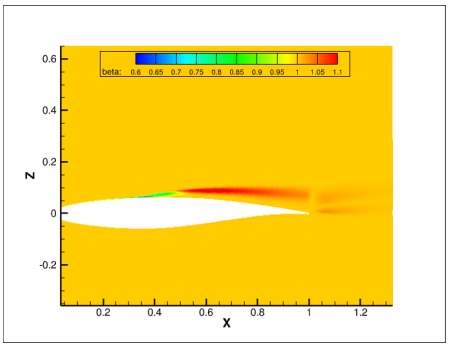
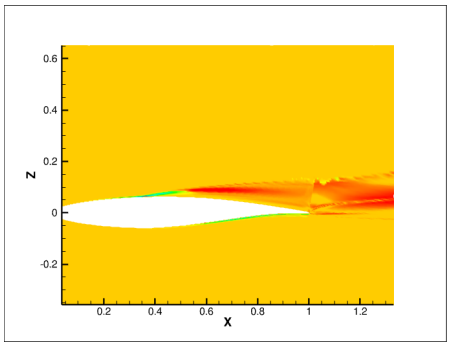


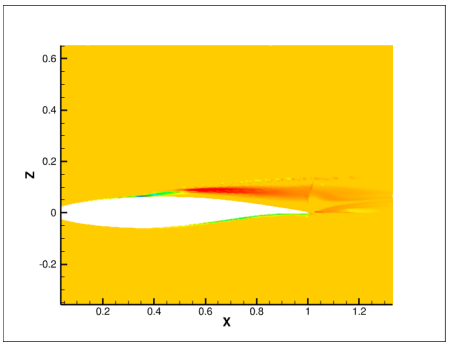
Figure D.31: C_p distribution for the NN-augmented TAU solutions. Training 1-5 are indicated in Section 7.2.2.



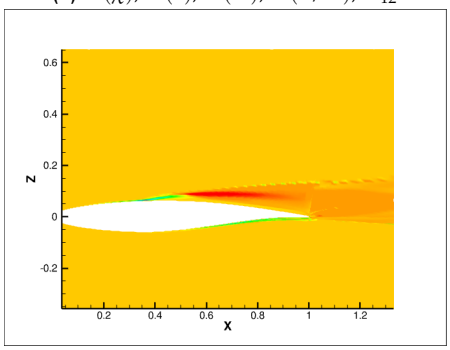
(a) Inversion result



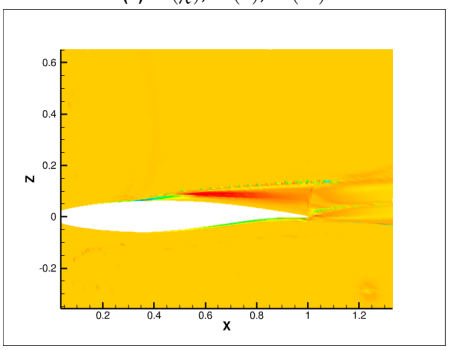
(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}



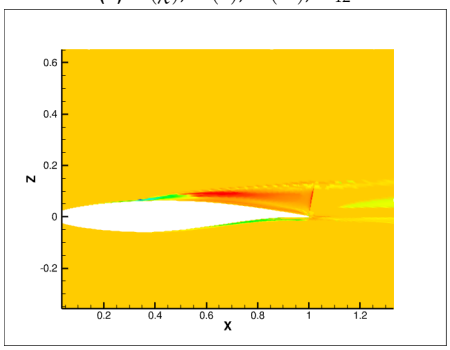
(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$



(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

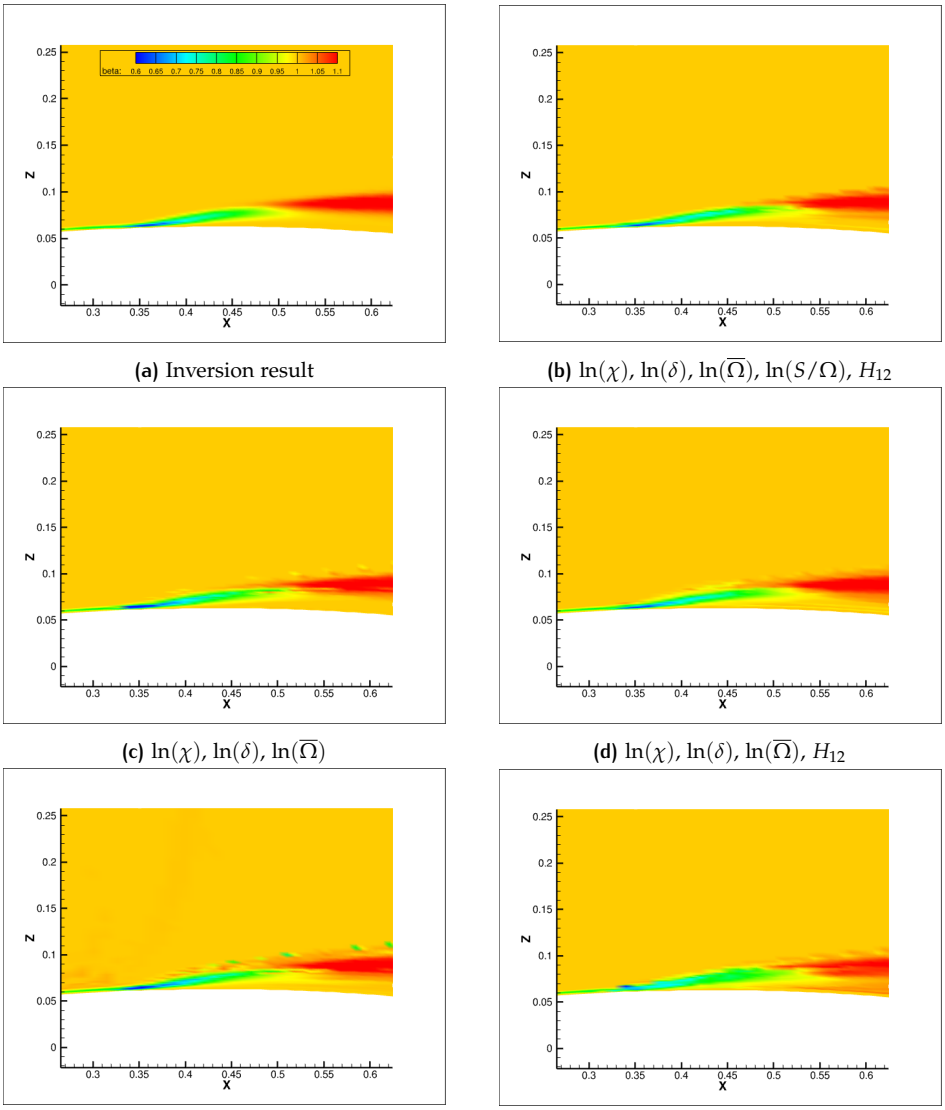


(e) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$



(f) $\ln(\chi)$, $\ln(\delta)$, H_{12}

Figure D.32: Results after connecting the trained neural networks with different features to TAU flow solver.



(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

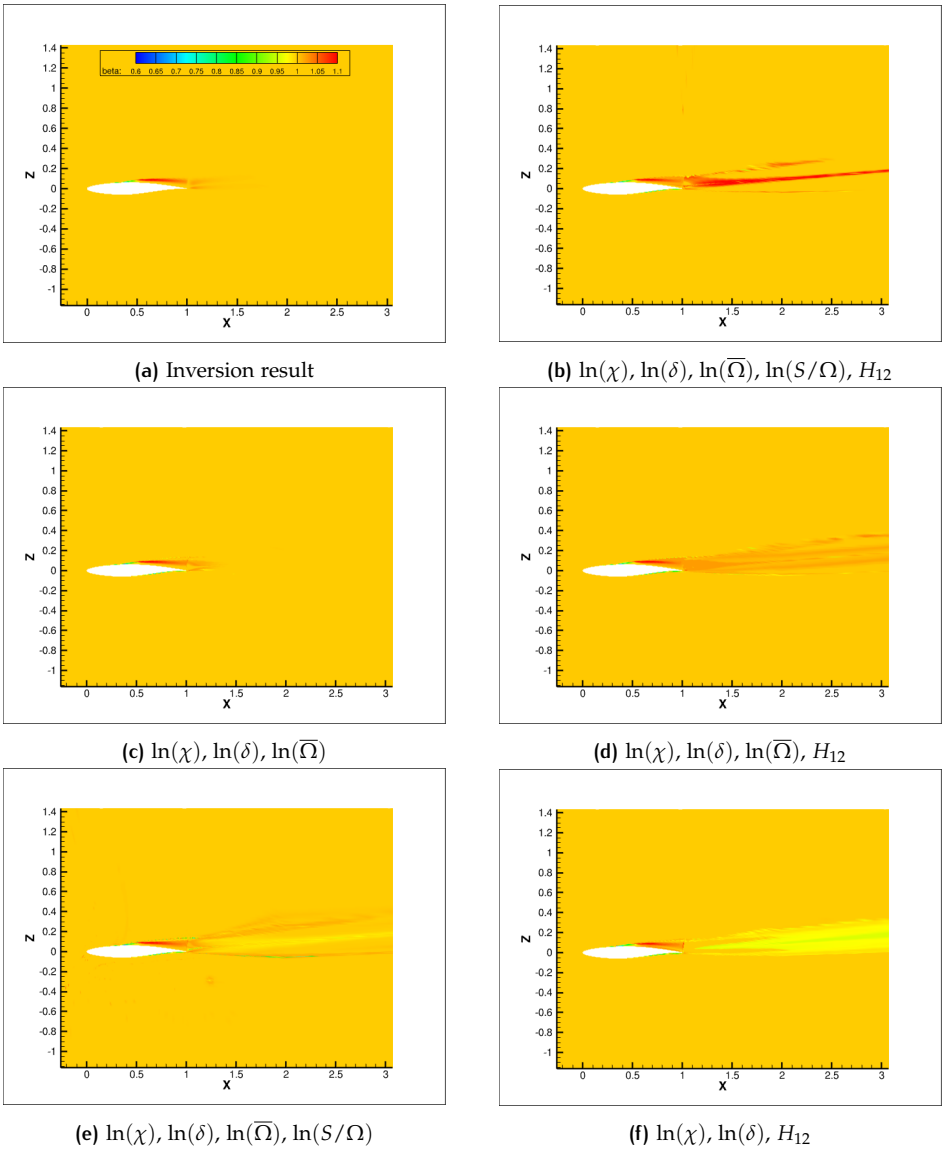
(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

(e) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

(f) $\ln(\chi)$, $\ln(\delta)$, H_{12}

Figure D.33: Results at shock foot after connecting the trained neural networks with different features to TAU flow solver.



(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

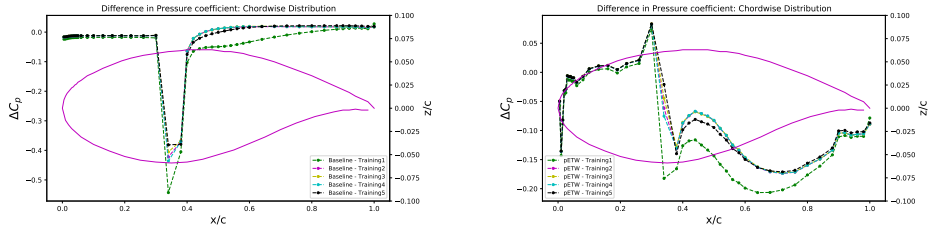
(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

(e) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

(f) $\ln(\chi)$, $\ln(\delta)$, H_{12}

Figure D.34: Results at shock foot after connecting the trained neural networks with different features to TAU flow solver.

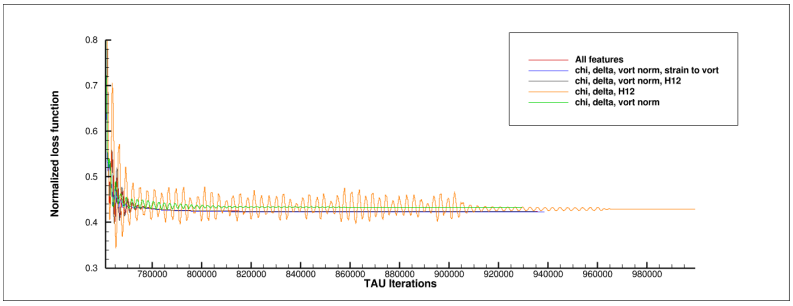


(a) Difference with Baseline solution

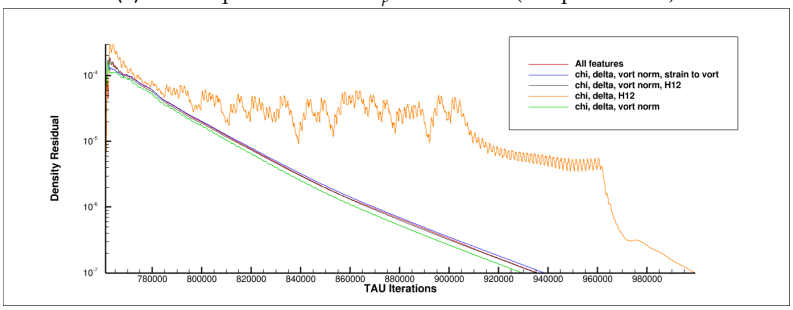
(b) Difference with pETW result

Figure D.35: ΔC_p distribution for the NN-augmented TAU solution. Training 1-5 are indicated in Section 7.2.2.

$M = 0.7235$, $AoA = 5.145$, $Re = 15323800$



(a) Mean squared error of C_p distribution (wrt pETW data)



(b) Density residual

Figure D.36: Residual variation with the flow solution in TAU after augmenting the baseline SA-neg model with the trained neural network at every 500 iterations.

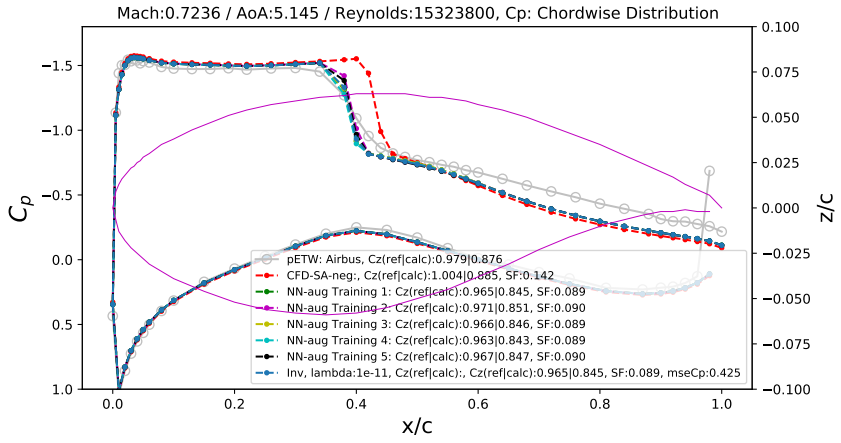
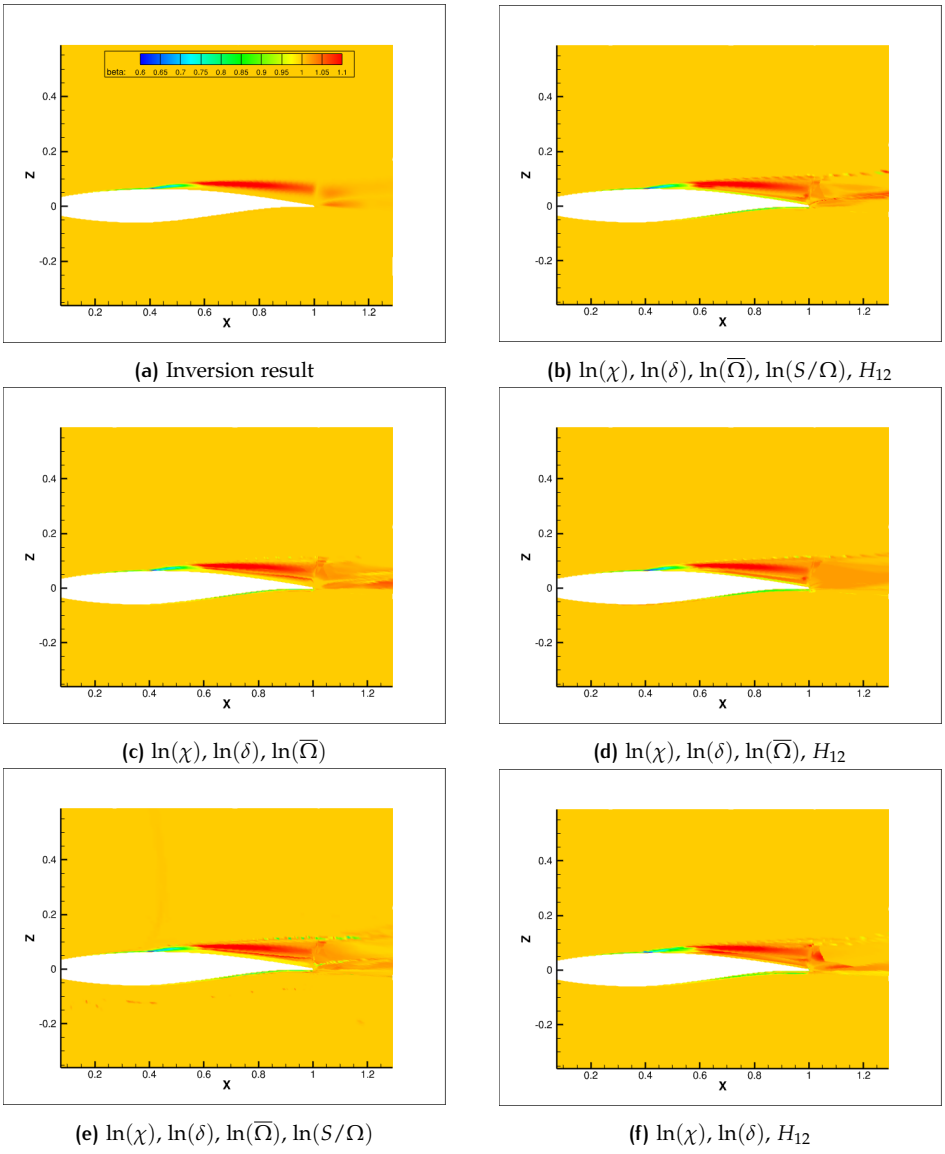


Figure D.37: C_p distribution for the NN-augmented TAU solutions. Training 1-5 are indicated in Section 7.2.2.



(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

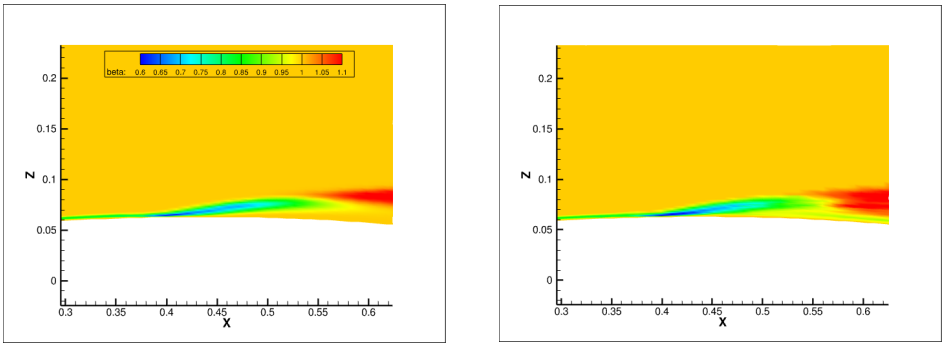
(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

(e) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

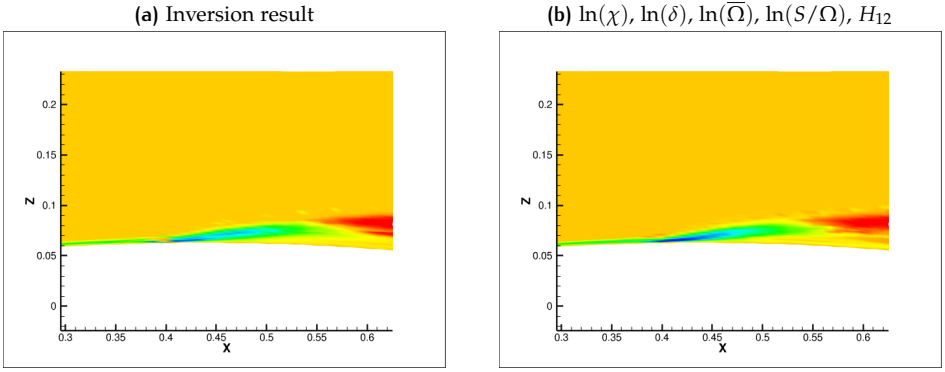
(f) $\ln(\chi)$, $\ln(\delta)$, H_{12}

Figure D.38: Results after connecting the trained neural networks with different features to TAU flow solver.



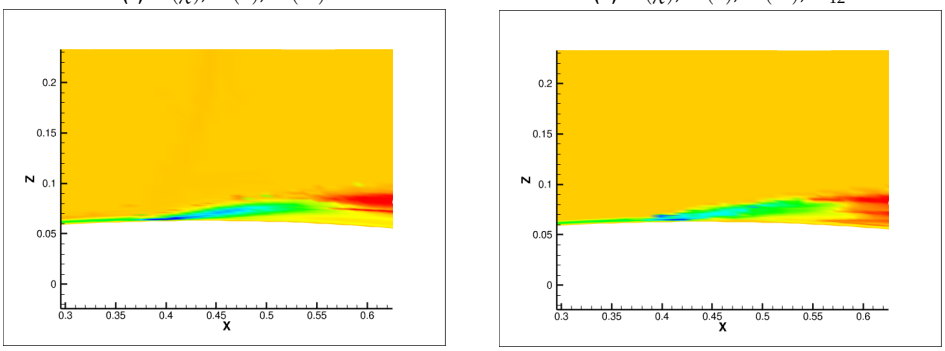
(a) Inversion result

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}



(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

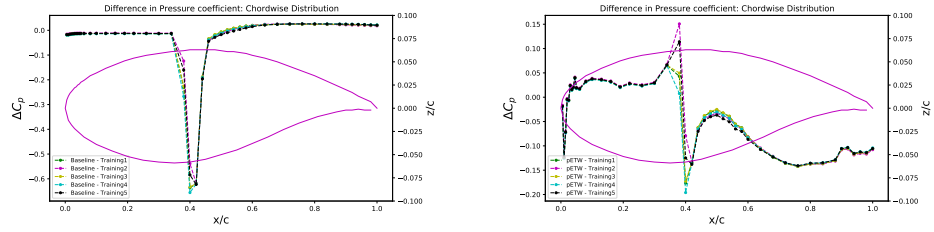
(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}



(e) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

(f) $\ln(\chi)$, $\ln(\delta)$, H_{12}

Figure D.39: Results at shock foot after connecting the trained neural networks with different features to TAU flow solver.



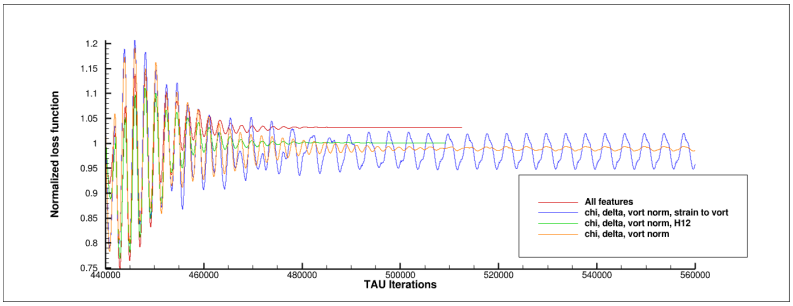
(a) Difference with Baseline solution

(b) Difference with pETW result

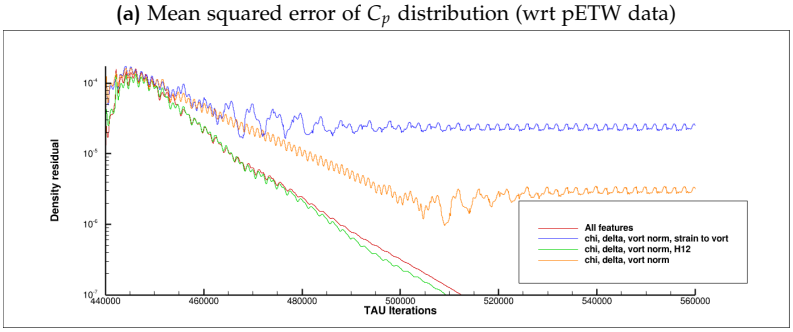
Figure D.40: ΔC_p distribution for the NN-augmented TAU solution. Training 1-5 are indicated in Section 7.2.2.

D.2.2 Testing on unknown flow cases

$M = 0.7206$, $AoA = 5.737$, $Re = 6331480$



(a) Mean squared error of C_p distribution (wrt pETW data)



(b) Density residual

Figure D.41: Residual variation with the flow solution in TAU after augmenting the baseline SA-neg model with the trained neural network after every 2000 iterations.

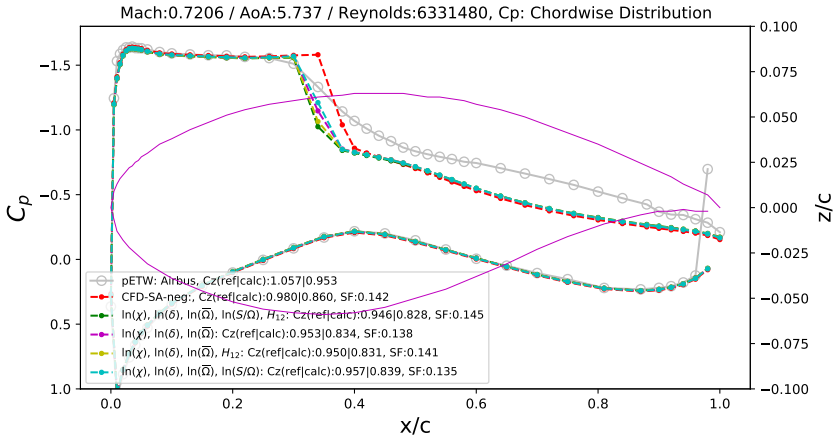
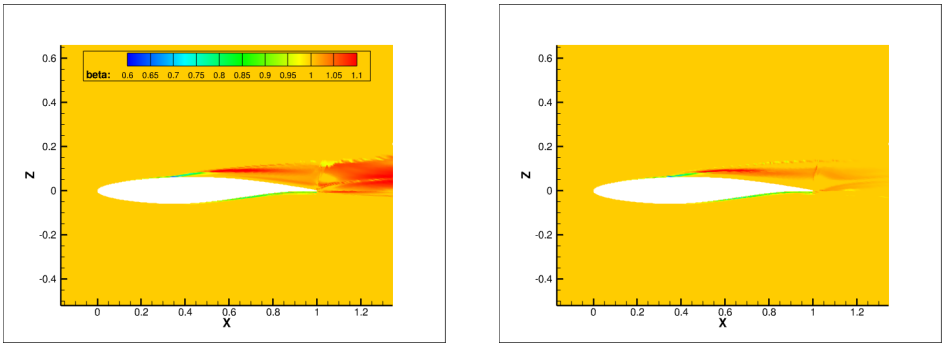
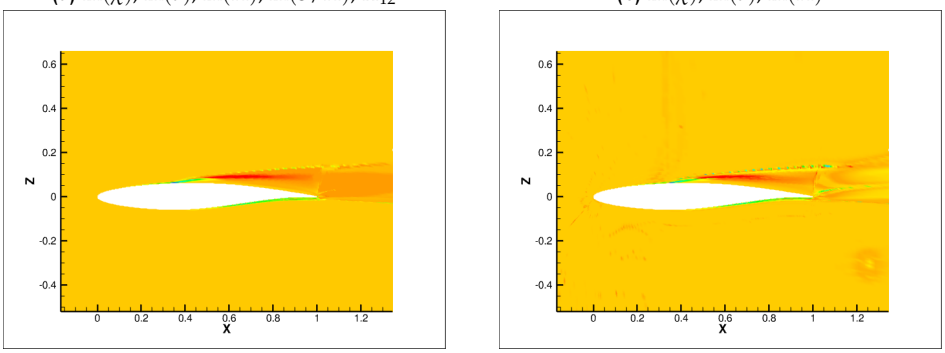


Figure D.42: C_p distribution for the NN-augmented TAU solutions.



(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

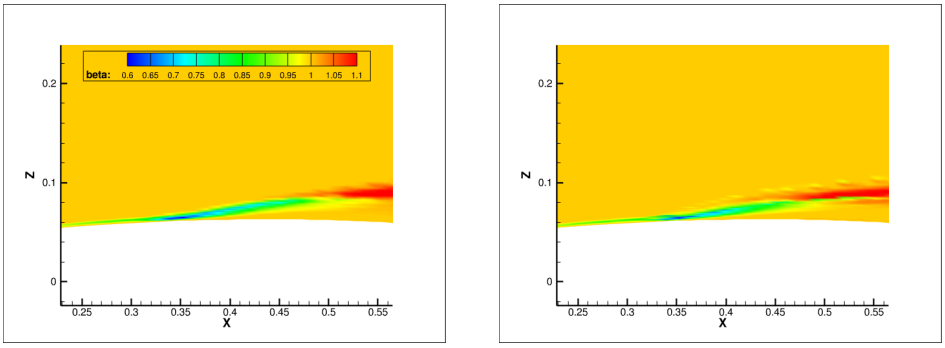
(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$



(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

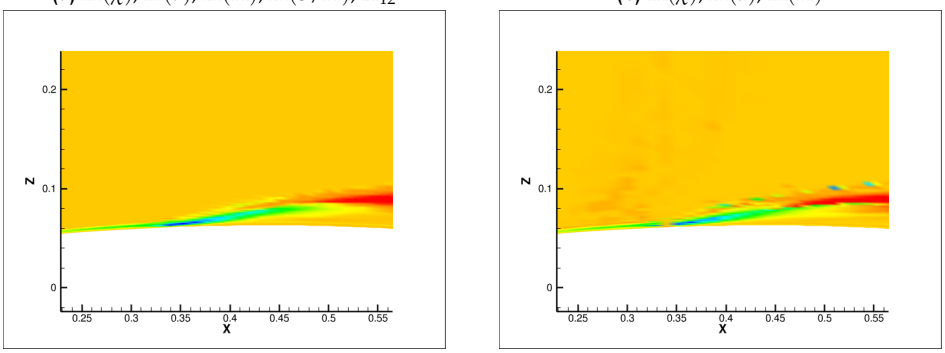
(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

Figure D.43: Results after connecting the trained neural networks with different features to TAU flow solver.



(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

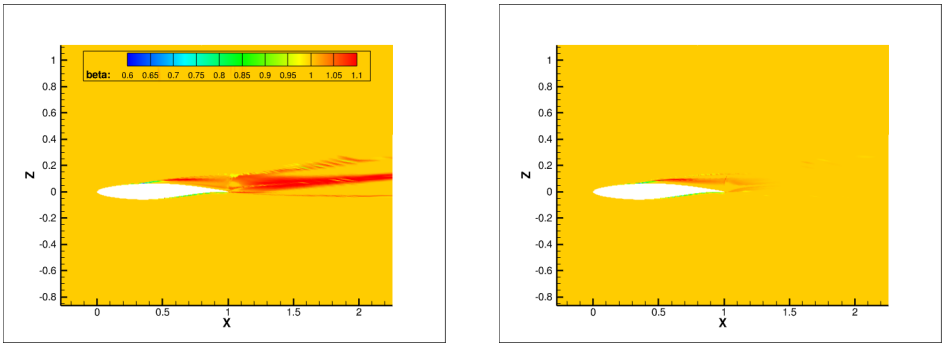
(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$



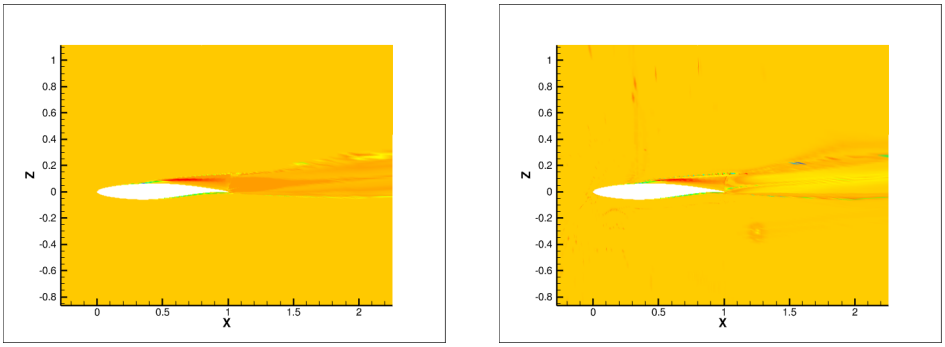
(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

Figure D.44: Results at shock foot after connecting the trained neural networks with different features to TAU flow solver.

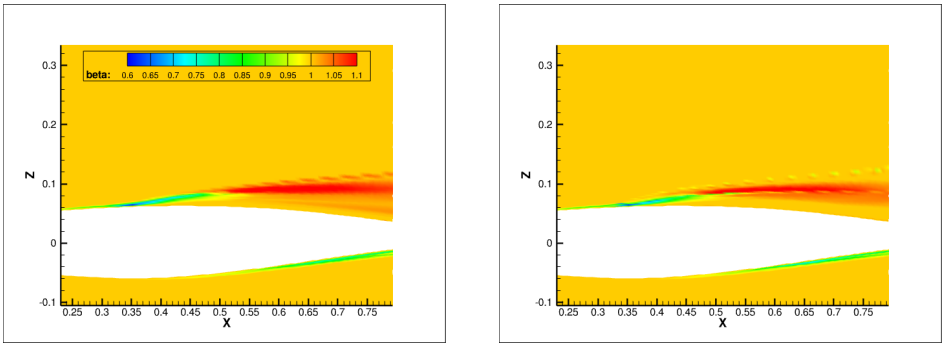


(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12} **(b)** $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

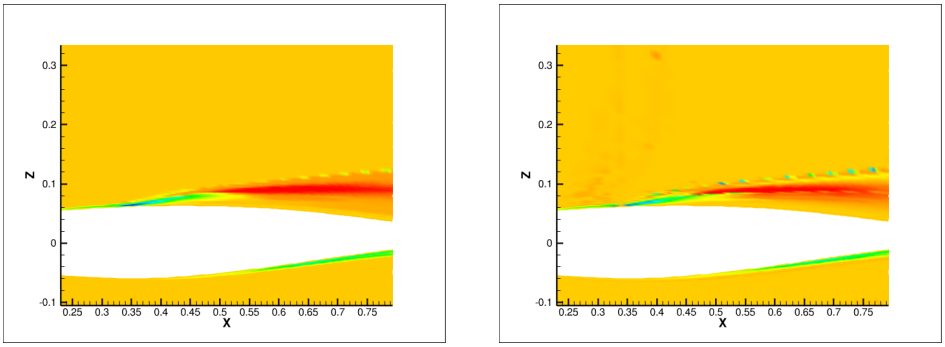


(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12} **(d)** $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

Figure D.45: Results after connecting the trained neural networks with different features to TAU flow solver.

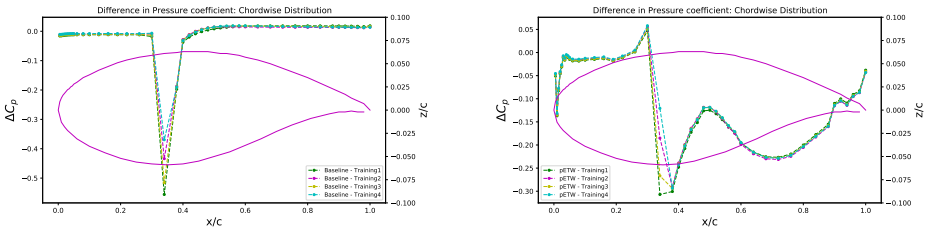


(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12} **(b)** $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$



(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12} **(d)** $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

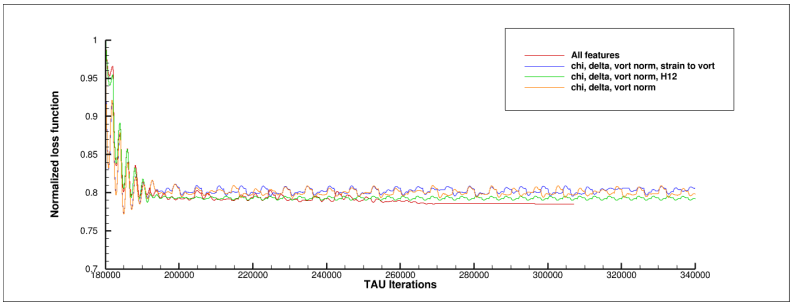
Figure D.46: Far field results after connecting the trained neural networks with different features to TAU flow solver.



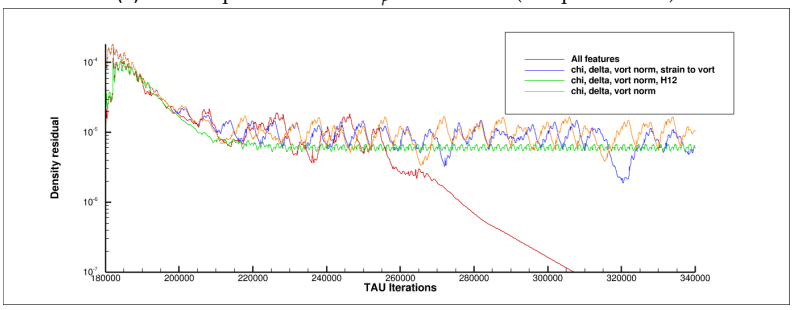
(a) Difference with Baseline solution **(b)** Difference with pETW result

Figure D.47: ΔC_p distribution for the NN-augmented TAU solution. Training 1-4 are indicated in Section 7.2.2.

$M = 0.7421$, $AoA = 4.420$, $Re = 8760680$



(a) Mean squared error of C_p distribution (wrt pETW data)



(b) Density residual

Figure D.48: Residual variation with the flow solution in TAU after augmenting the baseline SA-neg model with the trained neural network after every 2000 iterations.

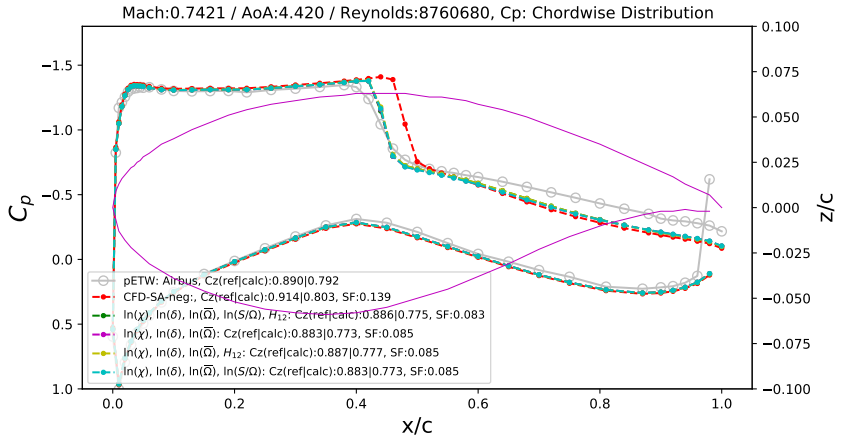
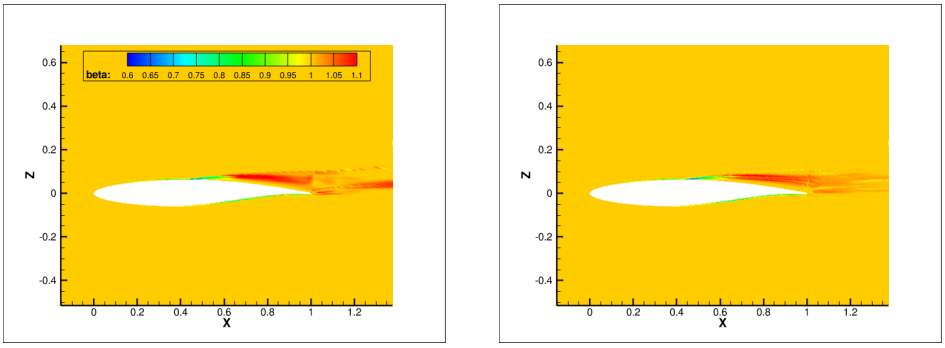


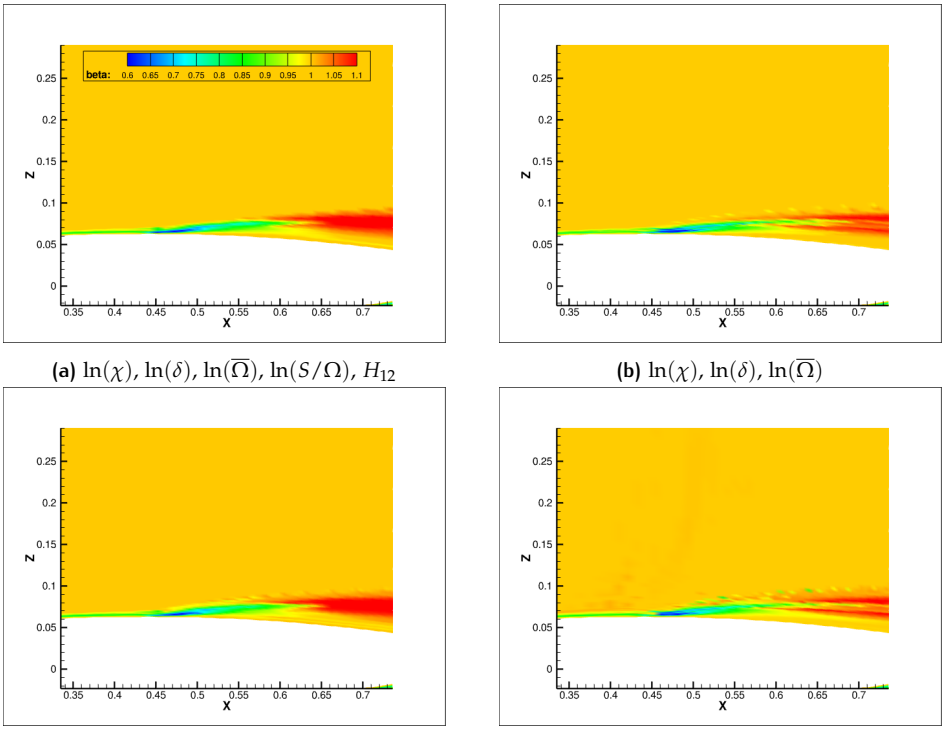
Figure D.49: C_p distribution for the NN-augmented TAU solutions.



(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12} **(b)** $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12} **(d)** $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

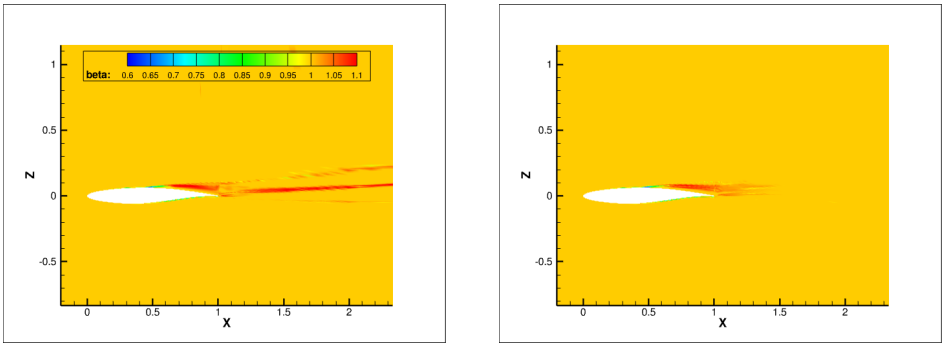
Figure D.50: Results after connecting the trained neural networks with different features to TAU flow solver.



(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12} **(b)** $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12} **(d)** $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

Figure D.51: Results at shock foot after connecting the trained neural networks with different features to TAU flow solver.



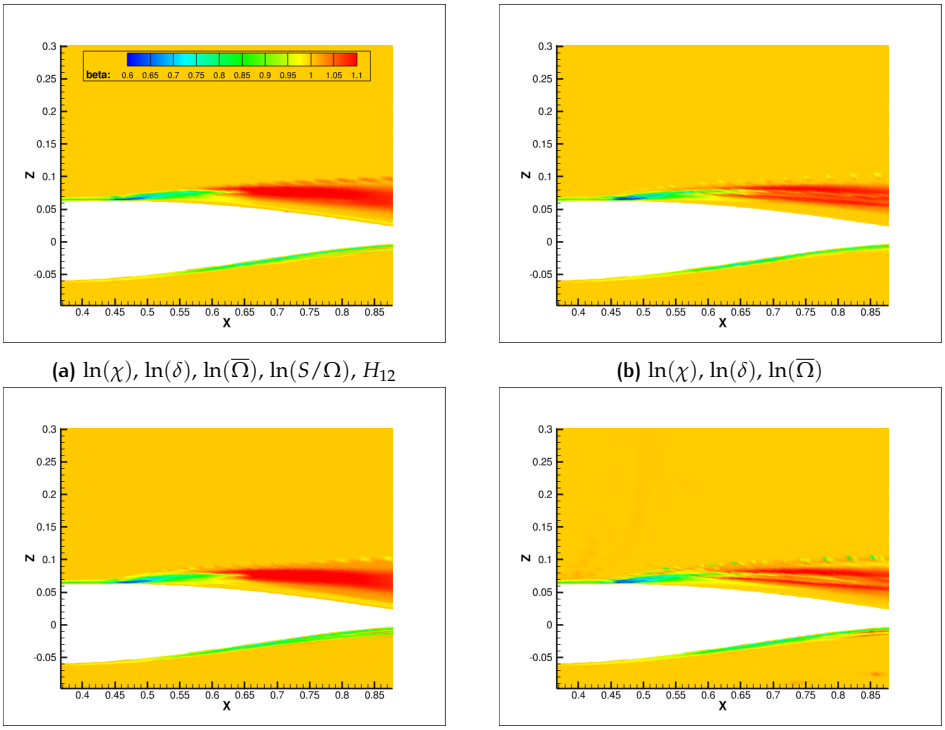
(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

Figure D.52: Results after connecting the trained neural networks with different features to TAU flow solver.



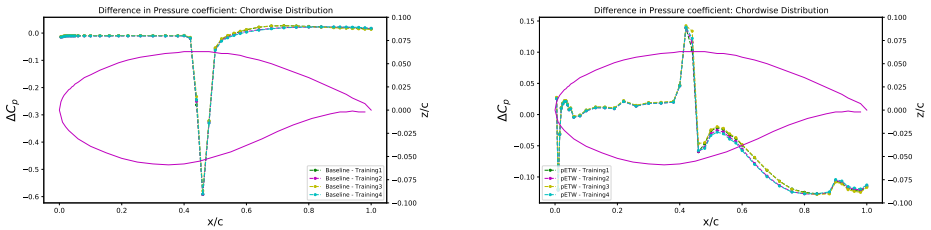
(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

Figure D.53: Far field results after connecting the trained neural networks with different features to TAU flow solver.

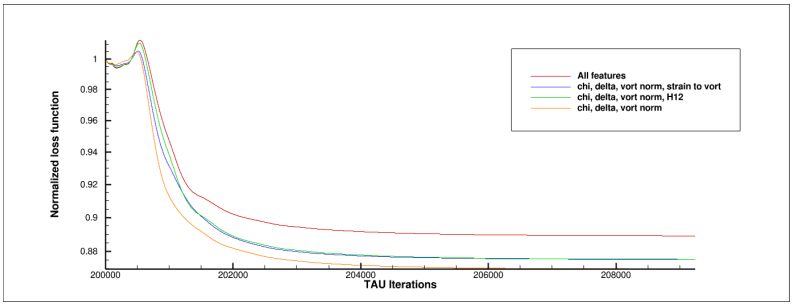


(a) Difference with Baseline solution

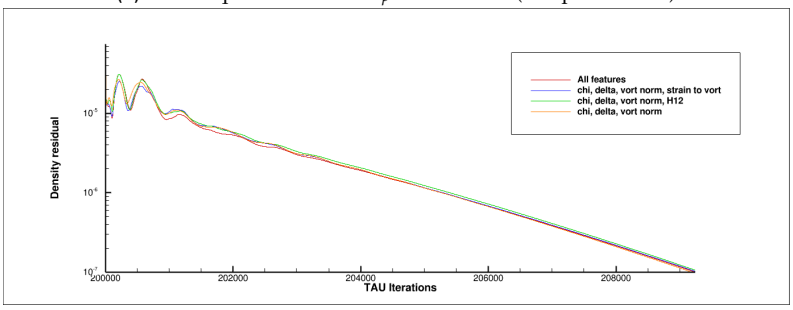
(b) Difference with pETW result

Figure D.54: ΔC_p distribution for the NN-augmented TAU solution. Training 1-4 are indicated in Section 7.2.2.

$M = 0.7173$, $AoA = 2.604$, $Re = 2680960$



(a) Mean squared error of C_p distribution (wrt pETW data)



(b) Density residual

Figure D.55: Residual variation with the flow solution in TAU after augmenting the baseline SA-neg model with the trained neural network after every 2000 iterations.

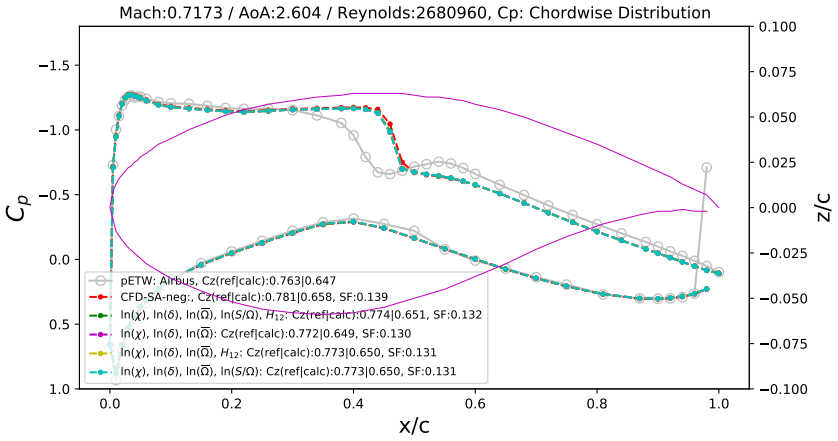
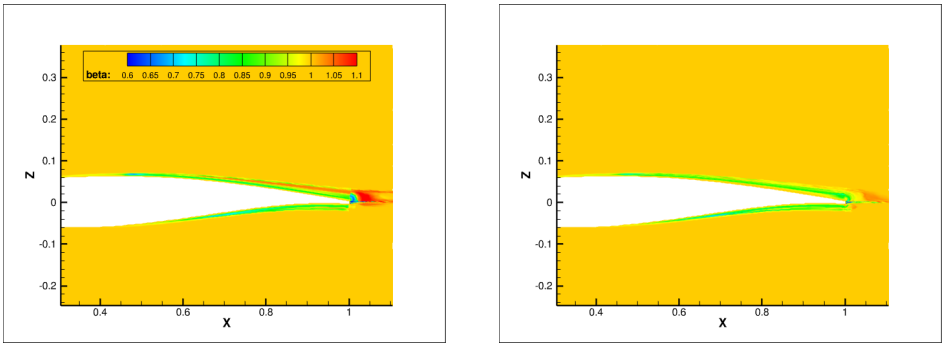
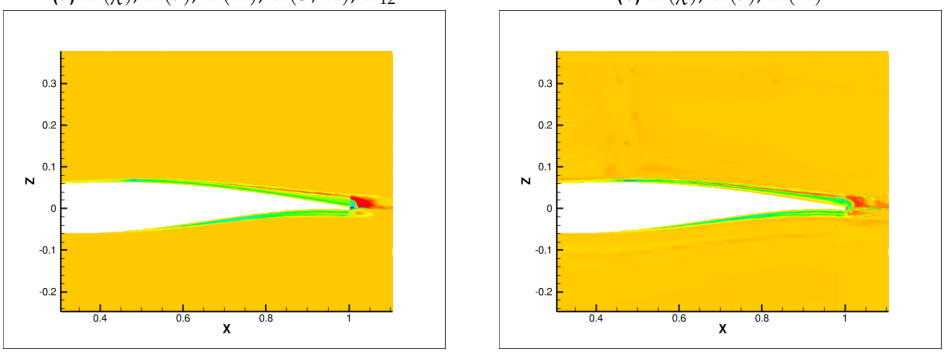


Figure D.56: C_p distribution for the NN-augmented TAU solutions.



(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

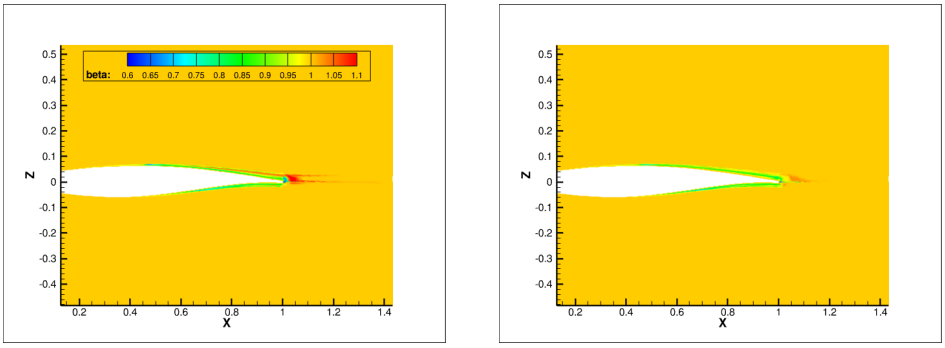
(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$



(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

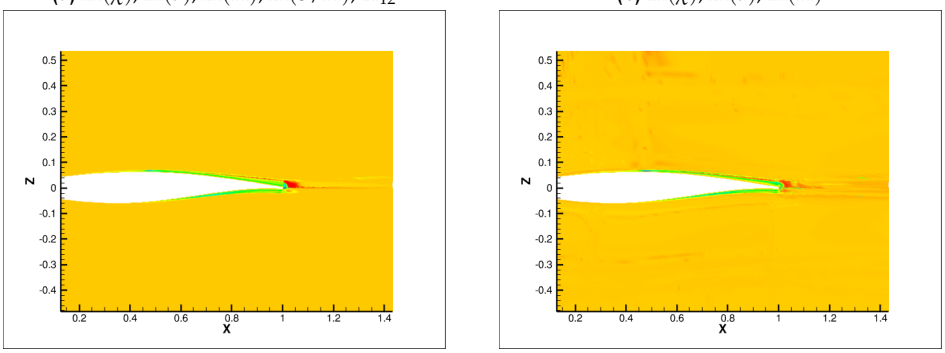
(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

Figure D.57: Results after connecting the trained neural networks with different features to TAU flow solver.



(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

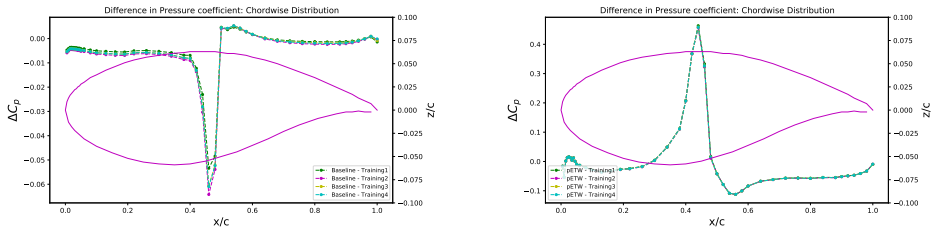
(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$



(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

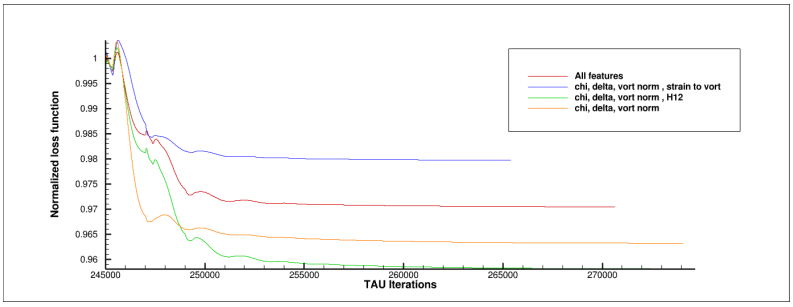
Figure D.58: Results at shock foot after connecting the trained neural networks with different features to TAU flow solver.



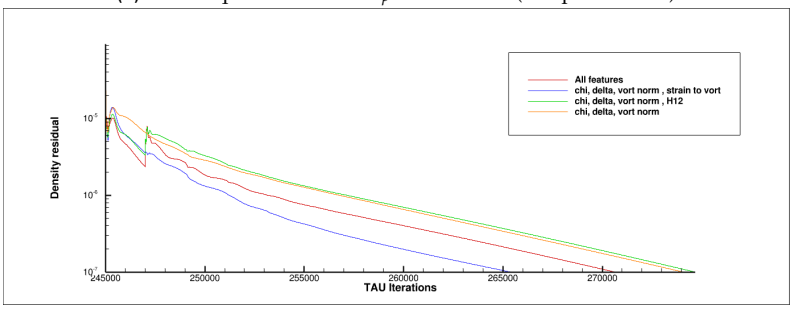
(a) Difference with Baseline solution **(b)** Difference with pETW result

Figure D.59: ΔC_p distribution for the NN-augmented TAU solution. Training 1-5 are indicated in Section 7.2.2.

$M = 0.7397$, $AoA = 1.403$, $Re = 13257500$



(a) Mean squared error of C_p distribution (wrt pETW data)



(b) Density residual

Figure D.60: Residual variation with the flow solution in TAU after augmenting the baseline SA-neg model with the trained neural network after every 2000 iterations.

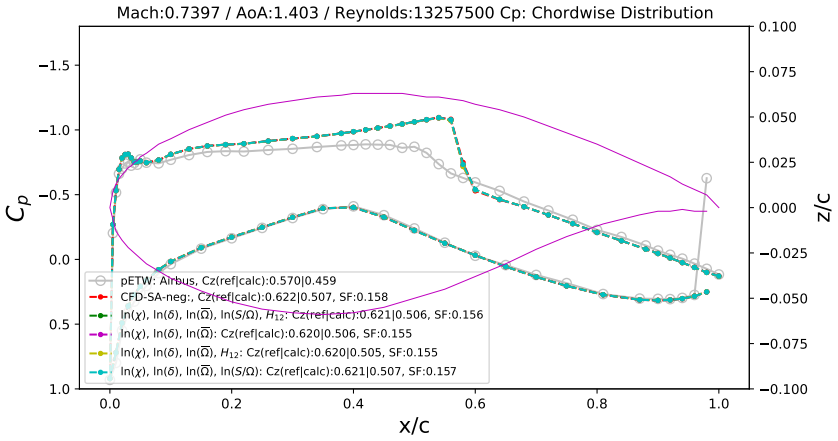
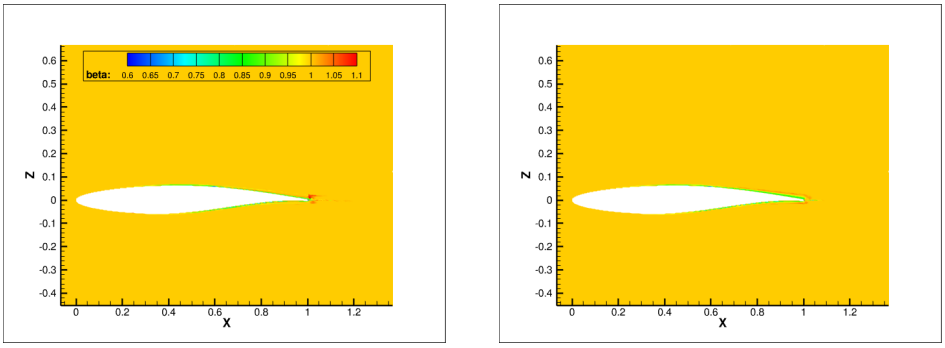
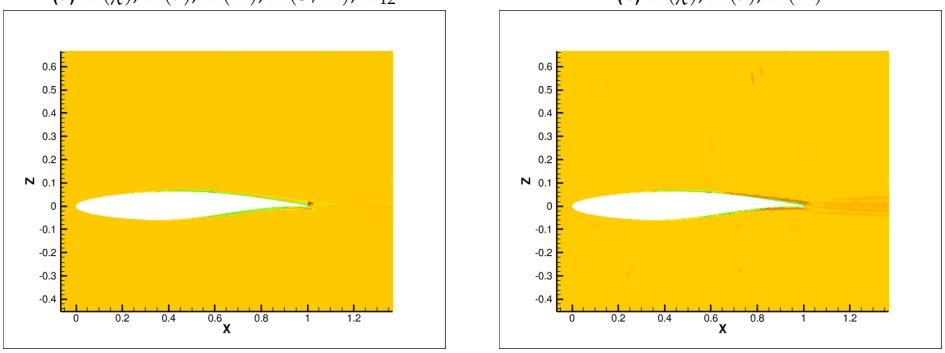


Figure D.61: C_p distribution for the NN-augmented TAU solutions.



(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

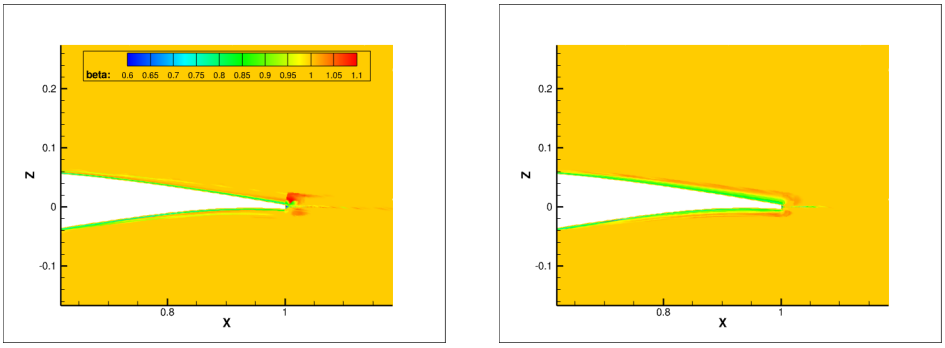
(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$



(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

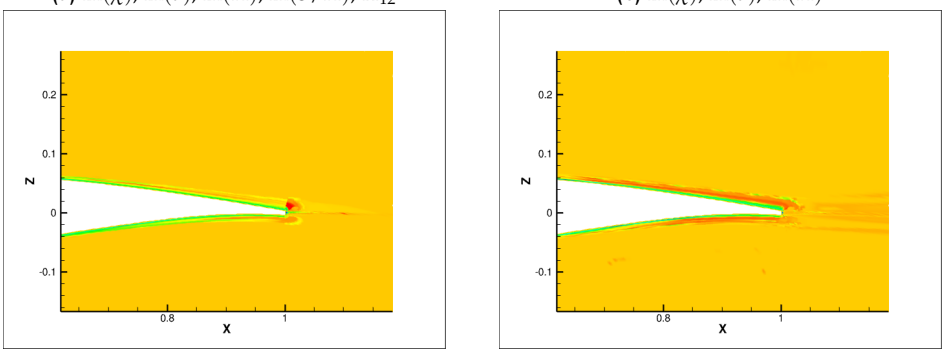
(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

Figure D.62: Results at shock foot after connecting the trained neural networks with different features to TAU flow solver.



(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

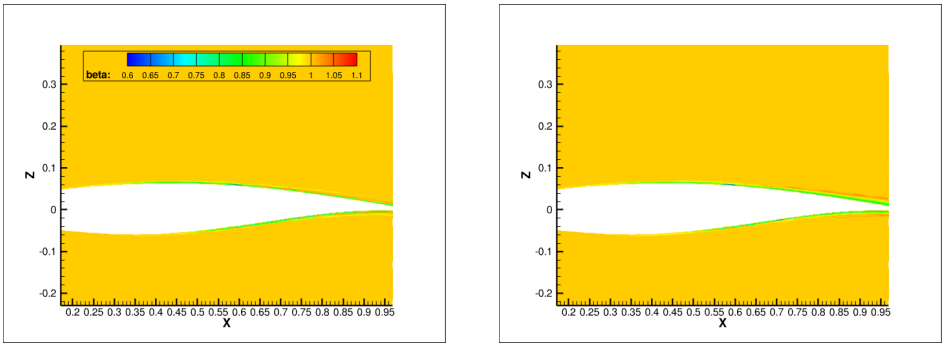
(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$



(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

Figure D.63: Results at shock foot after connecting the trained neural networks with different features to TAU flow solver.



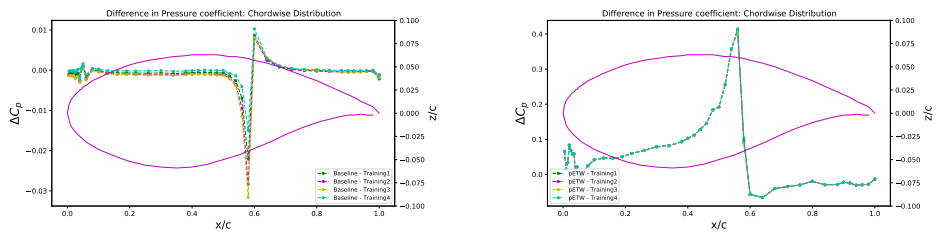
(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}

(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

Figure D.64: Results after connecting the trained neural networks with different features to TAU flow solver.

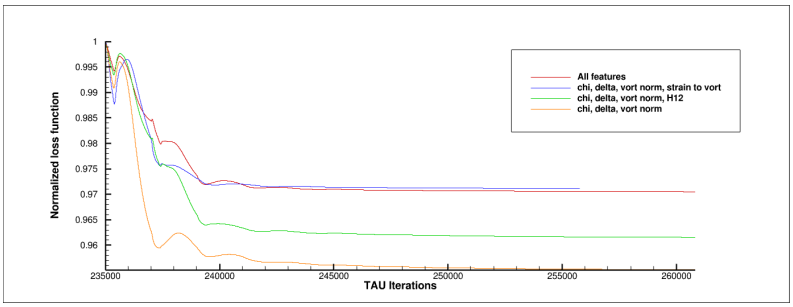


(a) Difference with Baseline solution

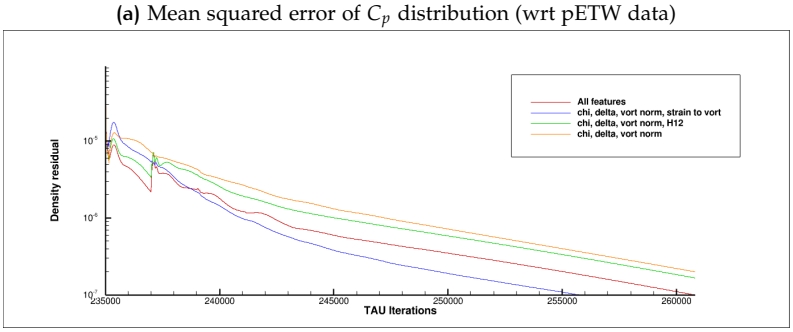
(b) Difference with pETW result

Figure D.65: ΔC_p distribution for the NN-augmented TAU solution. Training 1-4 are indicated in Section 7.2.2.

$M = 0.7110$, $AoA = 5.145$, $Re = 15363000$



(a) Mean squared error of C_p distribution (wrt pETW data)



(b) Density residual

Figure D.66: Residual variation with the flow solution in TAU after augmenting the baseline SA-neg model with the trained neural network after every 2000 iterations.

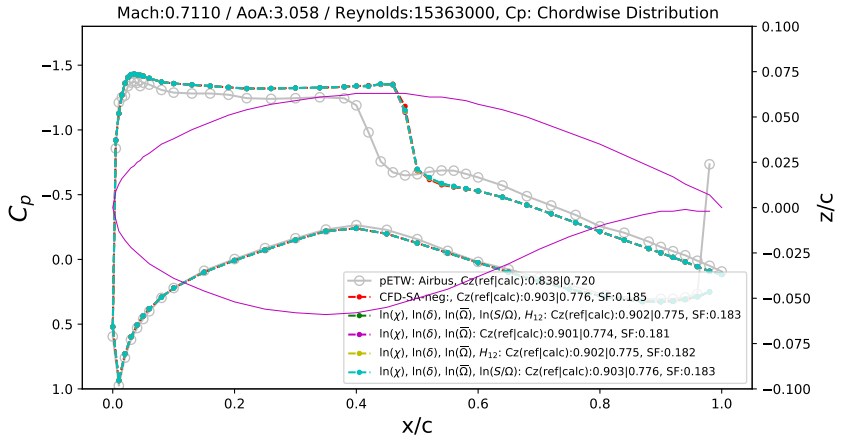
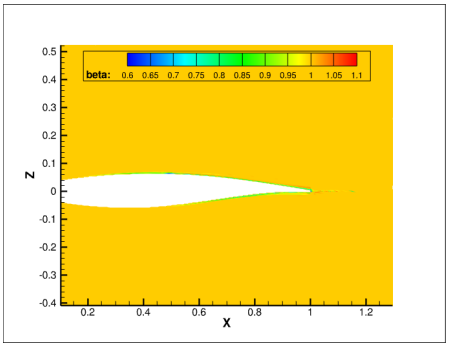
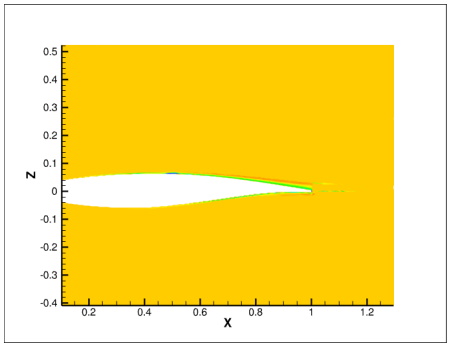


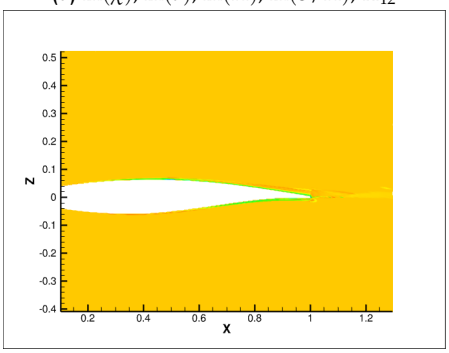
Figure D.67: C_p distribution for the NN-augmented TAU solutions.



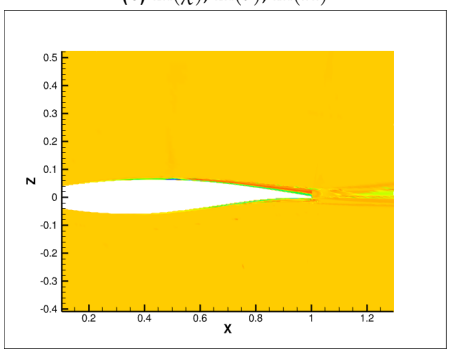
(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}



(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

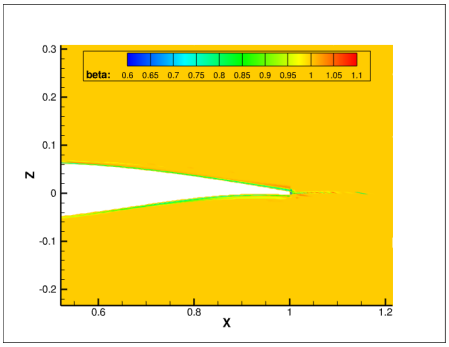


(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

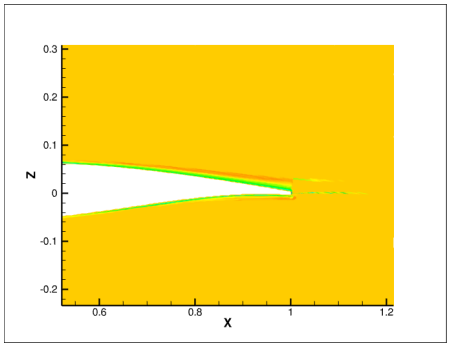


(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

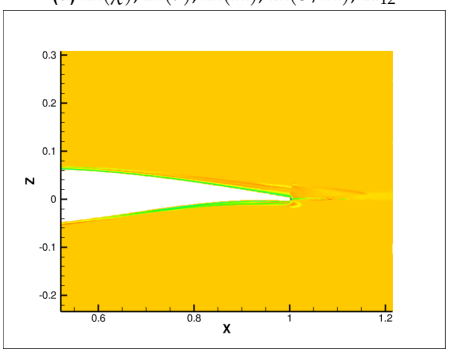
Figure D.68: Results at shock foot after connecting the trained neural networks with different features to TAU flow solver.



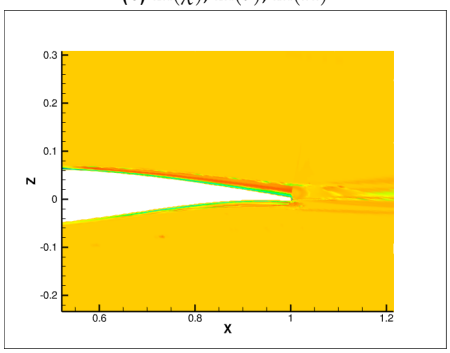
(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}



(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

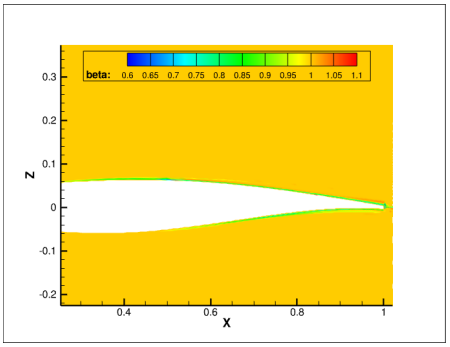


(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

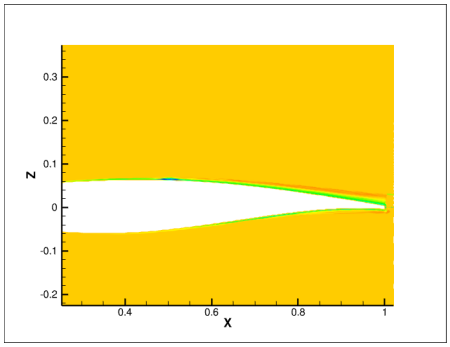


(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

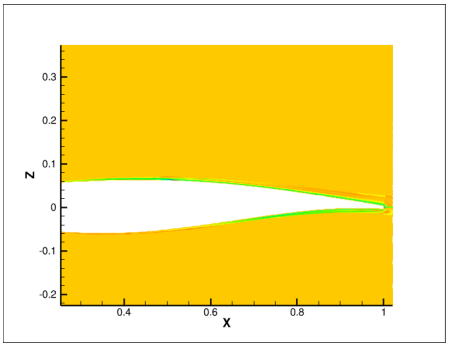
Figure D.69: Results at shock foot after connecting the trained neural networks with different features to TAU flow solver.



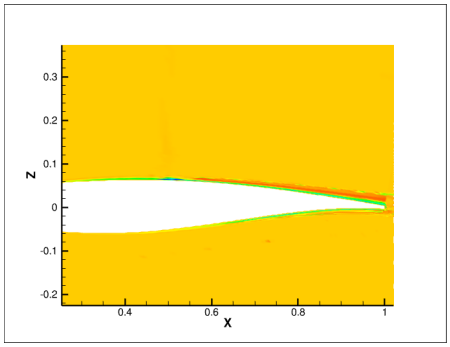
(a) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$, H_{12}



(b) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$

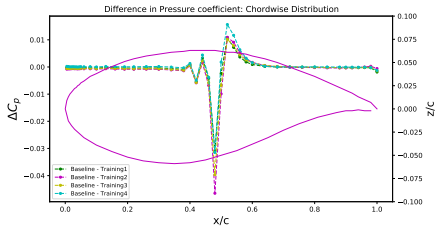


(c) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, H_{12}

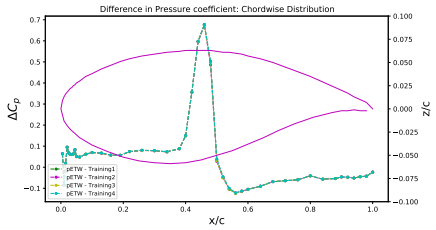


(d) $\ln(\chi)$, $\ln(\delta)$, $\ln(\overline{\Omega})$, $\ln(S/\Omega)$

Figure D.70: Results after connecting the trained neural networks with different features to TAU flow solver.



(a) Difference with Baseline solution



(b) Difference with pETW result

Figure D.71: ΔC_p distribution for the NN-augmented TAU solution. Training 1-4 are indicated in Section 7.2.2.