Improving Data-Driven RANS Turbulence Modelling For Separated Flow Scenarios

In Support of Formula 1 Aerodynamic Development

Monica Lacatus

Delft University of Technology

TU Delft

Improving Data-Driven RANS Turbulence Modelling For Separated Flow Scenarios

In Support of Formula 1 Aerodynamic Development

by

Monica Lacatus

To obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on 25th of June 2024 at 12:00.

Student Number:	5713595
Project Duration:	September 2023 - June 2024
Thesis Committee:	Dr. Richard Dwight, TU Delft, Committee chair and supervisor
	Dr. Saša Kenjereš, TU Delft, Additional member
	Dr. Bijoy Bera, TU Delft, Additional member
	Tyler Buchanan, TU Delft, External member
Faculty:	Thesis carried out in Faculty of Aerospace Engineering, TU Delft
	Thesis defended in Faculty of Chemical Engineering, TU Delft

Cover: DeepAI generated image of turbulent flow around Formula 1 car.

TU Delft

Preface

While I conducted this thesis study at the TU Delft Aerodynamics Department in collaboration with Williams Formula 1 Racing, its defense will take place at the TU Delft Chemical Engineering Department as part of the final stage of obtaining my Master's degree in Chemical Engineering. Although this study aims to improve Data-Driven RANS turbulence modeling for separated airflow scenarios, it still offers many relevant observations and approaches applicable to Chemical Engineering. Turbulence is a universal phenomenon; we can find it just as easily in the wake of a Formula 1 car as in the flow around the tubes of heat exchangers. The challenges in modeling it are also universal and even more significant in the chemical industry, where simple airflow is replaced by multiphase, reacting flow. I hope this work will be useful to future Chemical Engineering students who wish to continue modeling this chaotic physical phenomenon to design our future reactors.

I chose to pursue my Master's thesis in the Aerodynamics Department at TU Delft due to my interest in the transport phenomena part of Chemical Engineering. Topics such as turbulence modeling and computational fluid dynamics stood out to me during my degree, leading me to focus on these topics for my Master's thesis. I believe working in the Aerodynamics Department has provided me with a fresh perspective on these subjects.

First and foremost, I am grateful to my main supervisor, Dr. Richard Dwight, and my daily supervisor, Tyler Buchanan, for entrusting me with this thesis project in the Aerodynamics Department despite my background in Chemical Engineering. I am deeply thankful for your consistent support, always being available to address any questions or concerns I had. It was a pleasure collaborating with you both on a professional and personal level. Special thanks to Tyler for facilitating the APG case used throughout this thesis and to Kay Hoefnagel for sharing his setup of the NASA-Hump case. Additionally, I express my appreciation to Richard and all the researchers involved in developing the SpaRTA framework, which greatly contributed to my thesis. Next, I would like to acknowledge Alistair from Williams F1 Racing for involving me in this project and for our insightful discussions regarding the overarching goals of enhancing RANS for F1 aerodynamic design. Finally, I would like to thank Dr. Saša Kenjereš, my supervisor from the Chemical Engineering Department, with whom I had several interesting and insightful discussions about my progress and the different challenges in turbulence modeling.

Lastly, I would like to thank some of the most important people in my life. Zohar, your unwavering support has been a constant source of strength for me. From the lowest points to the highest, you have been there, building a relationship filled with love. To my friends Endino, Felix, and Elia, thank you for the memorable moments we shared in Delft. I thoroughly enjoyed our coffee sessions, carbonara dinners, tennis matches, concerts, and gossip sessions. You made my time in Delft truly enjoyable. To all my other friends at TU Delft, I cherish the time we spent together, whether it was movie nights, parties, dinners, etc. I also want to express my gratitude to my best friend Isabella, who has been my pillar of strength since secondary school, supporting me every step of the way. Last but not least, I am indebted to my parents for their unwavering support, sacrifices, and endless love, which have been instrumental in my academic journey.

I hope this work serves as an inspiration to others, especially women, encouraging them to pursue their dreams and embrace new challenges in both life and academia. As the saying goes, if you reach for the moon, you'll land among the stars — or better yet, in the TU Delft Aerodynamics Department.

Monica Lacatus
Delft, June 2024

Summary

In recent years, computational fluid dynamics (CFD) has become an indispensable design tool across various industries. It allows engineers to tackle complex fluid dynamics problems that would otherwise require expensive and time-consuming real-life experiments. For Formula 1 teams, who must experiment within very strict time limits in the wind tunnel and on track, the ability to simulate airflow around their race cars under various conditions is crucial for maintaining competitiveness in the fast-paced world of Formula 1 racing. While several simulation approaches have been developed throughout the last century, Reynolds-Averaged Navier-Stokes (RANS) simulations remain the industry standard for simulating turbulent flows as they allow engineers to conduct simulations more efficiently. However, this efficiency comes at the expense of simulation accuracy, as RANS is not able to resolve all the different scales of turbulence and is thus characterized by different levels of uncertainties.

Recent advances in data-driven RANS turbulence modeling have enabled the partial correction of these uncertainties in RANS simulations. However, obtaining a correction that is generalizable under different geometries and flow conditions remains a challenge. Furthermore, turbulence models are calibrated to fit canonical flow regimes, and thus correcting these models in the entire domain leads to a disturbance in these calibrations, worsening model performance. One solution is to divide the domain into separate regions based on identifiable physical phenomena and apply local corrections without disturbing the calibrated regions. For separated flow cases, which are commonly found in Formula 1 race car design, the most important region is the shear layer as that is where RANS shows the largest discrepancies.

In this thesis, a classifier has been developed to distinguish the shear layer from the rest of the domain based on the ratio between turbulent kinetic energy production and destruction, as well as turbulence intensity. Within this classifier region, corrections to the $k-\omega$ SST turbulence model are made by extracting model form errors from high-fidelity data using a technique known as k-corrective-frozen RANS. These corrections include a residual term R added to both the k and ω equations, and a residual term b_{ij}^{Δ} for the anisotropy of the Reynolds stress tensor. The Spars Regression of Turbulent Stress Anisotropy (SpaRTA) framework based on elastic-net regularization is applied to regress symbolic expressions for the corrections, enabling them to be applied to simulations of unseen test cases. To ensure that the models accurately represent local turbulence behavior, new terms have been constructed from local flow variables and included in the model expressions.

The models discovered with the SpaRTA framework for the shear layer demonstrate promising results, notably improving the prediction of separation and reattachment positions. Additionally, the models have been tested on various geometries and simulations at different Reynolds numbers, demonstrating a certain level of generalizability. While there is room for further improvement, the thesis shows that integrating targeted model corrections into RANS simulations, informed by isolated shear layer data, can notably enhance the understanding and prediction of shear layer dynamics in 2D-separated flows.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
BIC	Bayesian Information Criterion
CFD	Computational Fluid Dynamics
DNS	Direct Numerical Simulations
EARSM	Explicit Algebraic Reynolds-Stress Models
FIA	Federation Internationale de l'Automobile
GEP	Gene Expression Programming
GMM	Gaussian Mixture Models
HF	High Fidelity
IQR	Interquartile range
LES	Large Eddy Simulations
LEVM	Linear Eddy Viscosity Models
PC	Propagation Classifier Simulations
PCA	Principal Component Analysis
PCC	Pearson Correlation Coefficient
PCD	Propagation Classifier Dynamic Simulations
RANS	Reynolds-Averaged Navier-Stokes
RITA	Relative Term Importance Analysis
SpaRTA	Sparse Regression of Turbulent Stress Anisotropy
SFS	Sequential Feature Selector
SIMPLE	Semi-Implicit Method for Pressure Linked Equations
SVD	Singular Value Decomposition
TBNN	Tensor Basis Neural Network

Symbols

Greek Alphabet

Symbol	Definition	Unit
α_n	Regression coefficients used in SpaRTA	-
β	Context 1 (β, β^*): Turbulence model coefficients	-
	Context 2: Normalization factor	-
γ	Turbulence model coefficient	-
δ_{ij}	Kronecker delta	-
ϵ	Context 1: Rate of dissipation	$m^2\ s^{-3}$
	Context 2: Tolerance for clustering algorithms	-
Θ	Set of parameters used in GMM analysis	-
Θ_R, $\Theta_{b_{ij}^{\Delta}}$	Coefficient vectors for b_{ij}^{Δ} and R models	-
λ	Elastic net regularization weight	-
μ	Context 1: Dynamic viscosity	$kg\ m^{-1}\ s^{-1}$
	Context 2: Mean of a distribution	-
μ_t	Eddy dynamic viscosity	$kg\ m^{-1}\ s^{-1}$
ν	Kinematic viscosity	$m^2\ s^{-1}$

Greek Alphabet Continued

Symbol	Definition	Unit
ν_t	Eddy kinematic viscosity	m^2 s^{-1}
π	Gaussian distribution weights	-
ρ	Density	kg m^{-3}
σ	Context 1 (σ, σ_k, σ_ω, σ_ϵ): Turbulence model coefficient	-
	Context 2: Classifier function	-
	Context 3: Standard deviation	-
Σ	Context 1: Matrix of singular values	-
	Context 2: Summation	-
τ_{ij}	Reynolds-stress tensor	m^2 s^{-2}
$\tau_{ij,ratio}$	Ratio of total to normal Reynolds stresses	-
ω	Specific rate of dissipation	s^{-1}
Ω	Rotation rate tensor	s^{-1}

Latin Alphabet

Symbol	Definition	Unit
a	Context 1: Mixing parameter used in elastic next regularization	-
	Context 2: Mean intra-cluster distance	-
a_1	Coefficient used in eddy viscosity limiter in $k-\omega$ SST model	-
a_{ij}	Anisotropic component of Reynolds-stress tensor	m^2 s^{-2}
b	Mean nearest-cluster distance	
b_{ij}	Part of Reynolds-stress tensor anisotropic component	-
b_{ij}^{Δ}	Anisotropy correction of Reynolds-stress tensor	-
B	Input vector for SpaRTA library	
c	Context 1: Chord length	-
	Context 2: Centroid in K-Means analysis	-
C_1, C_2, C_μ	Turbulence model coefficients	-
$C_{b_{ij}^{\Delta}}$	Library of functions for b_{ij}^{Δ} models	-
C_f	Skin friction coefficient	-
C_k	Convection of turbulent kinetic energy	m^2 s^{-3}
C_p	Pressure coefficient	-
C_R	Library of functions for R models	-
$CD_{k\omega}$	Cross-diffusion coefficient in $k-\omega$ SST model	-
D_k	Destruction of turbulent kinetic energy	m^2 s^{-3}
$D_{f,k}$	Diffusion of turbulent kinetic energy	m^2 s^{-3}
f_1,f_2, f_μ	Damping functions in $k-\epsilon$ model	-
f_d	Boundary layer marker function	-
F_1, F_2	Blending functions in $k-\omega$ SST model	-
$F1, ..., F15$	Features used in clustering and SpaRTA regression	-
F_{norm}	Normalized form of feature	-
F_{min}, F_{max}	Minimum/maximum value of feature	-
G_n	Part of R modelling ansantz, representative of T_{ij}^n	m^2 s^{-3}
h	Height	m
H	Characteristic domain height	m
i	Context 1: Index notation	-
	Context 2: Number of iterations	-
I_m	mth tensor invariant	-
k	Turbulent kinetic energy	m^2 s^{-2}

Latin Alphabet Continued

Symbol	Definition	Unit
K	Context 1: Number of cluster centroids used in K-Means analysis	-
	Context 2: Number of Gaussian distributions used in GMM analysis	-
l	Context 1: Characteristic domain length	m
	Context 2: Turbulence length scale	m
L	Maximized value of likelihood function of GMM model	-
N	Number of points in computational domain	-
$N(\mu_k, \sigma_k)$	Gaussian distribution	-
p	Pressure	kg m^{-1} s^{-2}
P_k	Production of turbulent kinetic energy	m^2 s^{-3}
PS	Reatio of pressure normal stresses to shear stress	-
Q_1, Q_3	Upper and lower quartiles	-
r_{nk}	Responsibility, part of GMM analysis	-
R	Model-form error correction in $k-\omega$ SST model	m^2 s^{-3}
R^2	Coefficient of determination	
Re	Reynolds number	-
Re_c	Context 1: Reynolds number based on chord length	-
	Context 2: Critical Reynolds number	-
Re_d	Distance based Reynolds number	-
Re_H	Reynolds number based on hill/domain height	-
Re_k	Turbulence based Reynolds number	-
Re_T	Turbulence Reynolds number	-
Re_Ω	Vorticity Based Reynolds number	-
$RITA_{P_k/D_k}$	RITA ratio of production to destruction	-
$RITA_{C_k/D_k}$	RITA ratio of convection to destruction	-
$RITA_{D_{f,k}/D_k}$	RITA ratio of diffusion to destruction	-
$RITA_{C_k/D_{f,k}}$	RITA ratio of convection to diffusion	-
S_{ij}	Mean strain rate tensor	s^{-1}
t	Time	s
t_{ij}	Viscous stress tensor	m^2 s^{-2}
$T_{ij}^{(n)}$	nth Pope tensor basis	-
TI	Turbulence intensity	-
TS	Time scale ratio	-
U_i	Velocity in i direction	m s^{-1}
x_i	Cartesian coordinate vector	m
$x_1, ..., x_n$	Points in dataset	-
X	Dataset	-
y	Wall distance	m
y^+	Dimensionless first cell height of mesh	-

Other Notations

Notation	Definition
$\langle\cdot\rangle$	Averaged quantity
$\cdot'$	Filtered quantity
$\cdot_\infty$	Free stream value of $\cdot$
$\cdot_{ref}$	Reference value of $\cdot$

1

Introduction

Turbulent flows are encountered everywhere in nature and engineering but are notoriously hard to model [1]. Accurate predictions of these flows are vital for many branches of industry and research, especially for Formula 1 (F1) race car design. Turbulent flows are highly irregular and chaotic, characterized by regions of intense vorticity [2]. Turbulence develops at high flow velocities when the instabilities which naturally develop in the flow can no longer be damped out by viscous forces. As an F1 car races around the track at speeds exceeding 350 km/h, it significantly disturbs the surrounding air, generating a turbulent airflow as illustrated in Figure 1.1. This turbulence directly impacts the two primary forces that automotive engineers focus on controlling in high-performance racing car design: drag and downforce.

Drag is the aerodynamic force that opposes the direction of motion of the car. Higher levels of drag are equivalent to increased air resistance, which can significantly reduce the car's speed on the straights. This increase in drag is countered by an increase in thrust provided by the engine, however, this has a direct effect on fuel efficiency and costs. Downforce, on the other hand, is the aerodynamic force that pushes the car into the track, increasing tire grip and improving cornering speeds [3]. It is generated by various aerodynamic elements on the car, such as front and rear wings, diffusers, and the car's floor, which exploit the airflow to create a pressure difference between the top and bottom surfaces of the car [4].

There are two types of drag forces: friction drag, and pressure drag which is formally referred to as form drag. Friction drag arises from viscous shear forces that act tangentially to the car's surface. As the air travels over the surface, a thin layer of fluid known as the boundary layer forms where these viscous shear forces dominate, slowing down the air from its free-stream velocity to zero at the surface.

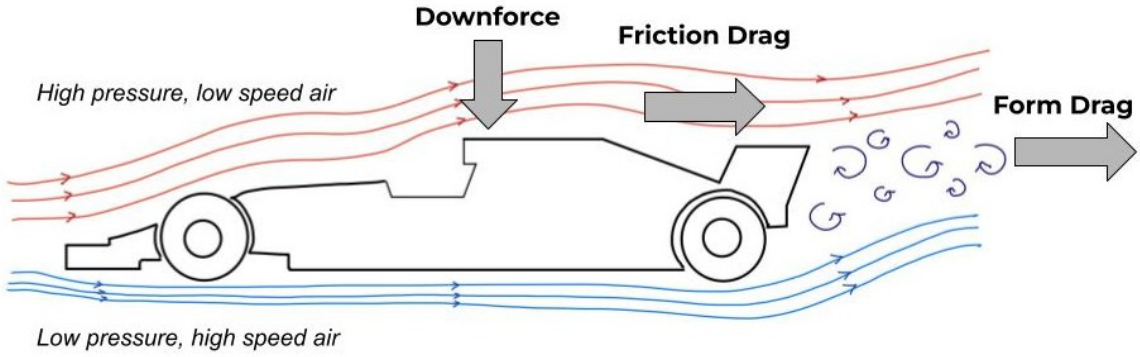
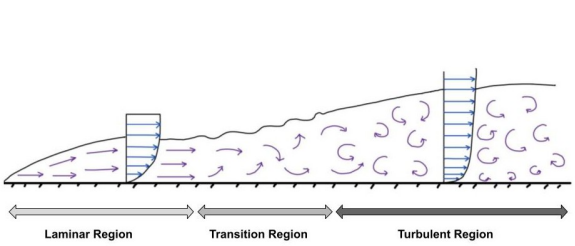


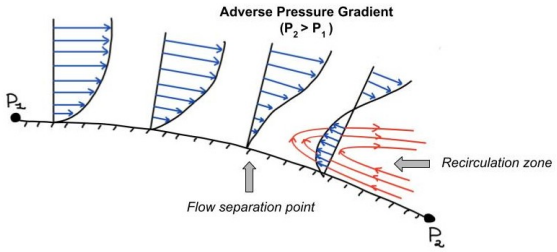
Figure 1.1: In this sketch, the simplified airflow distribution around an F1 car is depicted. Downforce arises as the pressure on the car's upper surface exceeds that on the lower surface. Friction drag occurs when airflow makes direct contact with the car's surface, while pressure drag results from flow separation, leading to the formation of turbulent wakes behind the car.

These velocity changes create high-velocity gradients, leading to the development of viscous shear forces. Figure 1.2a depicts a laminar boundary layer transitioning to turbulence. The turbulent boundary layer is much thicker, and its near-surface velocity gradients are higher, resulting in increased shear stress and overall friction drag compared to laminar flow.

Form drag arises from the pressure imbalance caused by flow separation from the car's surface. Figure 1.2b illustrates the changes in a boundary layer following a downward surface curvature. This curvature induces flow expansion, reducing flow velocity while increasing pressure. As depicted in this figure, the pressure at the end of the domain P_2 exceeds that at the beginning P_1, resulting in an adverse pressure gradient opposing the flow's motion. Flow separation initiates at the point where the wall shear forces diminish to zero, allowing for the development of reverse flow, as evidenced by the inverted velocity profiles and streamlines. As previously discussed, the velocity gradients in a laminar boundary layer are smaller than those in a turbulent boundary layer, enabling turbulent boundary layers to remain attached for longer periods. The changes in pressure distribution during flow separation affect the levels of downforce, potentially compromising the car's stability. Therefore, a thorough understanding of turbulent airflow behavior around an F1 car is crucial for designing a high-performance vehicle.



(a) Changes in mean flow velocity profiles during the transition of a laminar boundary layer to a fully turbulent boundary layer.

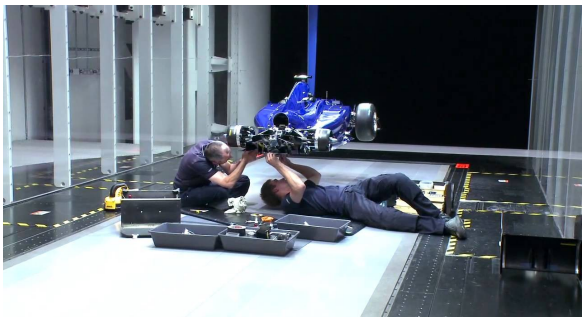


(b) Changes in mean flow velocity profiles of a boundary layer separating from a surface under an adverse pressure gradient.

Figure 1.2: Boundary layer behavior: transition to turbulence and separation under an adverse pressure gradient.

Analyzing the airflow around the car can be achieved through experimental methods in wind tunnels and on the track, as depicted in Figure 1.3. However, the sport's governing body, known as the FIA, has implemented strict time constraints on wind tunnel usage and on-track testing in the last several years [5], prompting teams to heavily rely on computational fluid dynamics (CFD) software to simulate flow behavior around their cars. CFD simulations are based on numerically solving the Navier-Stokes and continuity equations, a set of partial differential equations that describe momentum and mass conservation in a flow in 3D space and time. There are three main approaches available for simulating turbulent flow based on these equations: Direct Numerical Simulations (DNS), Large Eddy Simulations (LES), and Reynolds-Averaged Navier-Stokes (RANS) simulations.

(a) Flow-vis is a fluorescent powder mixed with oil, painted on F1 cars to visualize airflow patterns during on-track testing [6].



(b) Wind tunnel testing on a miniature car model [7].

Figure 1.3: Experimental tools to investigate airflow behavior around an F1 car.

DNS simulations involve resolving all scales of turbulence, which requires the discretization of the Navier-Stokes equations in 3D space and time using a very fine grid and simulating with a very small time step. This makes DNS simulations extremely computationally intensive. For instance, it would take approximately 1.3 years for a supercomputer to complete a full DNS simulation of a 4.6 m long Toyota

Prius driving on the highway at 120 km/h, as detailed in [8]. Due to this high computational cost, DNS simulations are not used in industry. Instead, they serve primarily as a research tool for validating other, less computationally expensive, methods of simulating turbulence (LES and RANS).

LES simulations are slightly less accurate than DNS simulations because they use a pre-defined filter to remove the smallest scales of turbulence from the simulation, focusing only on resolving the large scales in 3D space and time. The smaller scales are instead modeled using statistical methods typically defined in terms of the size of the numerical grid [2]. This makes LES simulations less computationally intensive than DNS. However, LES still requires solving the filtered Navier-Stokes equations in both time and 3D space, which results in a computational cost that is often considered too high for an F1 team.

RANS simulations represent the cheapest and fastest method for simulating turbulence. In RANS simulations, the Navier-Stokes equations are averaged, allowing for the analysis of flows in terms of their mean flow characteristics. However, this averaging procedure introduces several approximations and results in the loss of information regarding the different scales of turbulence [2]. Due to these approximations, RANS simulations are inherently less accurate than LES and DNS simulations. Despite these challenges, RANS simulations remain the preferred method for F1 teams, and much of the current research in turbulence modeling focuses on minimizing uncertainties inherent in these simulations to achieve more accurate flow predictions.

The time-averaged, incompressible RANS equations are formulated as follows:

$$\rho\frac{\partial U_i}{\partial t} + \rho U_j\frac{\partial U_i}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}(2\mu S_{ij} - \overline{\rho u_j' u_i'}) \tag{1.1}$$

where U denotes the mean flow velocity, ρ the density, p the pressure, μ the dynamic viscosity, S_{ij} the mean strain rate, and $\overline{u_j' u_i'}$ the fluctuating velocity tensor which is often referred to as the Reynolds-stress tensor $\tau_{ij} = \overline{\rho u_j' u_i'}$. The Reynolds-stress tensor quantifies the additional stress introduced by turbulent fluctuations in the flow. Since the RANS equations lack equations to solve for these fluctuating velocity components, closing the RANS equations requires additional approximations and model equations. According to Duraisamy et al.'s RANS uncertainty classification [9], this lack of closure and the associated loss of information is classified as a level 1 (L1) uncertainty.

The subsequent level of uncertainty L2 arises from model-form errors in approximating the Reynolds-stress tensor. This uncertainty can be understood by examining the most prevalent class of turbulence models, commonly known as Linear Eddy Viscosity Models (LEVMs), which rely on the Boussinesq hypothesis to approximate the Reynolds-stress tensor. This hypothesis expresses this tensor as being linearly related to the mean strain rate through an eddy viscosity term:

$$\tau_{ij} = 2\mu_t S_{ij} - \frac{2}{3}\rho k\delta_{ij} \tag{1.2}$$

where μ_t is the eddy viscosity and k is the turbulent kinetic energy. The formulation of the Reynolds-stress tensor thus requires additional model equations to solve for μ_t and k. These model equations typically vary depending on the type of turbulence models being used. In the F1 industry, as well as in other branches of automotive and aerospace design, the most commonly used LEVM is the $k-\omega$ SST model. This model equates the eddy viscosity to a ratio between k and ω (the specific rate of dissipation of k) and solves two partial differential equations, one for each of these quantities. This introduces L3 uncertainties arising from model-form errors in these partial differential equations. The final level of uncertainty, L4, pertains to uncertainties in the model coefficients used to close the k and ω equations.

In recent years, a new field of research known as Data-Driven Turbulence Modeling has developed as a response to the efforts made in discovering new and improved RANS models or calibrating existing ones to enhance their accuracy in predicting complex flows. This area of study focuses on applying Machine Learning (ML) tools, which are effective at handling large datasets and revealing complex underlying patterns, to help improve RANS predictions of turbulent flows [10][11].

The earliest data-driven RANS studies focused on addressing L2 uncertainties. Emory et al. approached this by perturbing the Reynolds-stress tensor [12] . This method involves dividing the flow field into regions where the RANS model performs well and regions where it does not. In the latter regions,

a certain level of uncertainty is injected into the anisotropic component of the Reynolds-stress tensor, after which the system is monitored for biases in the flow predictions.

Platteeuw et al. used a similar, though less intrusive, approach to address the L4 uncertainties [13]. Instead of directly perturbing the Reynolds-stress tensor, they treated the coefficients of the LEVM as variables that could vary within specified ranges with certain probabilities. This introduced a probabilistic element into the modeling process, allowing for the consideration of uncertainty in the model's predictions. Cheung et al. continued the efforts to address L4 uncertainties by introducing Bayesian uncertainty quantification methods [14] . They demonstrated this method by re-calibrating the model coefficients of the Spalart-Allmaras model for boundary layers under three different types of pressure gradients: favorable, zero, and adverse. Using the same Bayesian framework, Edeling et al. proved that there do not seem to be universal values for the closure coefficients of turbulence models [15]. In their study, Edeling et al. re-calibrated the model coefficients of the $k - \epsilon$ turbulence model for 13 very similar cases involving a flat-plate boundary layer under different pressure gradients. They found that the optimal values for the model coefficients differed for each case.

The study by Duraisamy et al. introduced a novel method known as Field Inversion Machine Learning (FIML) to address the L3 uncertainties in RANS modeling [9]. FIML applies supervised machine learning methods to infer information about the model-form error in LEVMs from high-fidelity simulation data sources such as LES and DNS. ML techniques are then used to reconstruct this model-form error in terms of flow variables that are available during RANS simulations.

Building upon the FIML framework, subsequent studies focused on advancing the ML methods used to learn the inferred corrections from high-fidelity data. Ling and Templeton introduced a neural network architecture known as the Tensor Basis Neural Network (TBNN) [16]. Their method involves modifying the formulation of the Reynolds-stress anisotropy by applying the extended eddy viscosity hypothesis proposed by Pope [17]. This modification leads to the derivation of Explicit Algebraic Reynolds Stress Models (EARSM), which serve as nonlinear extensions of LEVMs. EARSMs are based on a minimal integrity basis derived by projecting the Reynolds-stress anisotropy onto a set of tensorial polynomials [10]. The TBNN architecture is designed to maintain Galilean and rotational invariance, ensuring that the derived EARSM models can be effectively applied to different flow scenarios. This novel approach represents a significant advancement in Data-Driven RANS modeling, offering improved accuracy and robustness across different flow scenarios.

While ML methods like TBNNs offer flexibility in learning turbulence model corrections from high-fidelity data, they are not based on physically interpretable expressions for these corrections [10]. Relating them to physical observations thus becomes very difficult. Moreover, implementing these methods in CFD solvers, typically written in languages like C++ or Fortran, can be challenging because ML is predominantly performed in Python. To overcome these challenges, several studies have concentrated on developing approaches to infer model expressions for EARSMs that are straightforward to implement in solvers and have physical interpretability, directly from high-fidelity LES or DNS data.

One such approach is non-deterministic Gene Expression Programming (GEP), as outlined in Weatheritt and Sandberg's study [18]. Another method, demonstrated by Schmelzer et al., is deterministic symbolic regression [19]. This approach employs SpaRTA (Sparse Regression of Turbulent Stress Anisotropy), a sparsity-promoting regression technique that directs the search for model equations towards sparse algebraic expression. SpaRTA not only focuses on regressing model-form errors in the Reynolds-stress anisotropy but also aims to derive algebraic expressions for the model-form errors in turbulence model equations, thereby addressing both L2 and L3 uncertainties.

Kay Hoefnagel's Master's thesis at the TU Delft Aerodynamics Department focused on extending, validating, and analyzing SpaRTA's capability to discover universal model corrections for the $k - \omega$ SST model [8]. His research highlighted a significant challenge in Data-Driven RANS modeling: the corrections inferred can be non-local, meaning they may modify RANS turbulence models even in regions where these models have been finely tuned to accurately predict flow behavior, such as the boundary layer [10]. Moreover, these corrections, derived by regressing model equations on the full-field high-fidelity data, often fail to generalize effectively across different flow cases, often leading to convergence issues during simulations.

To address this issue, one approach is to derive and apply corrections selectively in regions where the RANS model performs poorly. This method entails inferring corrections from high-fidelity data specific

to the region of interest and activating these corrections only within that region during simulation. Implementing this approach requires developing a classifier capable of identifying areas with significant uncertainty, where corrections are necessary. By employing such a classifier, the amount of data used for regression can be significantly reduced, as it avoids training on parts of the domain where corrections are unnecessary, such as the boundary layer or the free stream. Ultimately, activating corrections only in regions where they are needed helps preserve the integrity of the well-predicted areas of the RANS model, thereby maintaining the accuracy of predictions in the full computation domain.

In F1 aerodynamic design, RANS often fails to accurately predict flow separation. As discussed earlier, flow separation directly impacts the pressure distribution around the car, influencing both drag and downforce. Therefore, accurately identifying the location of flow separation and subsequent reattachment is crucial in CFD simulations of F1 race cars. The $k-\omega$ SST model, widely used by F1 aerodynamics teams, tends to underestimate turbulent shear stress in the separated shear layer that forms when the boundary layer detaches from the car's surface. This underestimation leads to an exaggerated recirculation region and delayed reattachment location, which are essential for precise computation of drag and downforce. Correcting these RANS simulations of flow separation presents a significant challenge because any corrections must be applied without disturbing the boundary layer, where the RANS model is already well-calibrated, to avoid introducing new inaccuracies.

Therefore, this study aims to develop a classifier capable of distinguishing the shear layer from the rest of the flow domain and apply the SpaRTA methodology to infer symbolic model equations for the Reynolds-stress tensor and model-form errors in the $k-\omega$ SST model. The overarching goal is to discover model equations that can account for the missing shear layer physics that a classical RANS simulation fails to predict. This means that the classifier and model equations should be generalizable for changes in domain geometries and flow velocities. The SpaRTA training will concentrate solely on data obtained from the shear layer, simplifying the complexity of the regression problem by reducing the volume of data to be fitted. Based on this overarching aim, the research questions stated below have been formulated.

Thesis Research Questions

Main Question: Can the understanding and prediction of shear layer dynamics in 2D-separated flows be improved by incorporating targeted model corrections based on isolated shear layer data?

Sub-Questions:

1. Can a classifier be constructed for the shear layer and effectively used to activate model corrections only where necessary?
2. Is the classification consistent across different domain geometries and flow Reynolds numbers?
3. Does applying corrections exclusively in the shear layer region result in improved flow predictions in separated-flow scenarios?
4. Can the SpaRTA framework derive symbolic model expressions that accurately represent the Reynolds-stress tensor and $k-\omega$ SST model-form error in the shear layer?
5. Do the derived model equations yield consistent results for changes in domain geometries or Reynolds numbers of the flow?

2

Theoretical Background

2.1. The Physics of Turbulence

The simplest way to begin understanding turbulence is to observe what happens when water flows through a pipe, as depicted in Figure 2.1. The leftmost sketch illustrates the smooth, laminar flow of water in an ideal pipe with completely smooth surfaces.

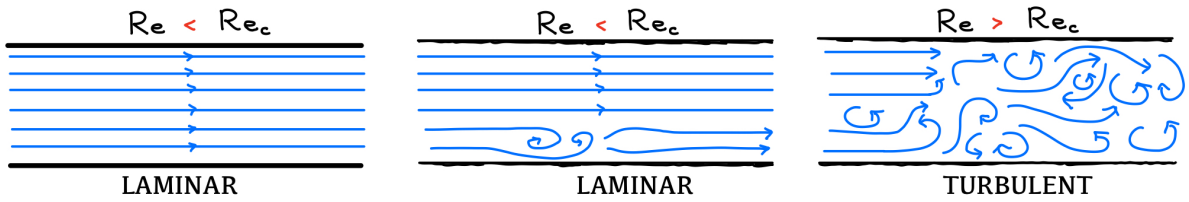


Figure 2.1: Below the critical Reynolds number (Re_c), the flow remains laminar despite natural instabilities that develop in the flow. Above the critical Reynolds number, these instabilities lead to turbulent flow development within the pipe.

In such a system, even with infinitely high flow velocity, there will be no transition to turbulence. This underscores a fundamental aspect of turbulence: the presence of natural disturbances or flow instabilities is essential for its existence. In reality, ideal smooth pipes do not exist; instead, the surfaces of real pipes have a certain degree of roughness and imperfections. As water flows over the surface of these pipes, small instabilities develop in the flow, as depicted in the middle sketch of Figure 2.1. However, these small instabilities do not lead to turbulence as long as the viscous forces in the flow — resulting from friction between fluid layers — are sufficient to dampen their effects.

This introduces the second important aspect of turbulence: turbulence arises only when local viscous forces, which typically suppress natural disturbances and flow instabilities, are overcome by other forces acting on the flow, such as inertial forces [2]. The primary criterion for determining whether a flow is turbulent is the **Reynolds number** Re:

$$Re = \frac{\rho U l}{\mu} \tag{2.1}$$

which depends on the flow density ρ, the dynamic viscosity μ, the flow velocity U and the characteristic length of the system l. Below a certain Reynolds number known as the critical Reynolds number Re_c, the flow remains laminar even if instabilities develop. The Reynolds number is not universal and therefore varies from one flow scenario to another. It is usually determined on an empirical basis or derived from stability theories [2].

The rightmost sketch of Figure 2.1 depicts turbulence as highly irregular and chaotic. The instabilities that develop act to deform the flow which results in flow regions characterized by high vorticity. The term **eddy**, is often used to describe these regions. Eddies require a certain amount of energy to sustain themselves and thus draw energy from the mean flow. The eddies undergo stretching and elongation over time as they encounter velocity gradients which ultimately results in them breaking into smaller eddies. The energy transfer from larger to smaller eddies is usually referred to as the **energy cascade**. The energy of the smallest eddies eventually dissipates into heat due to viscous effects [2].

The chaotic nature of turbulence renders it extremely difficult to model. Despite this challenge, numerous efforts over the last few decades have been dedicated to obtaining turbulence models capable of accurately and consistently representing this natural phenomenon, a topic that will be further explored in the upcoming sections.

2.2. RANS Turbulence Modelling

[1]The origins of RANS turbulence modeling can be traced back to the 19th century when Osborne Reynolds published his famous 1895 work [21] on the Reynolds decomposition approach. This decomposition can be used to express the instantaneous velocity $u_i(x,t)$ of a turbulent flow, as the sum of a mean velocity component $U_i(x)$ and a fluctuating component $u_i'(x,t)$:

$$u_i(x,t) = U_i(x) + u_i'(x,t) \tag{2.2}$$

The difference between these two velocity components is visualized in Figure 2.2, which displays a single axial instantaneous velocity profile across a pipe cross-section and its decomposition into its mean and fluctuating component. For general engineering applications, the mean velocity holds enough information to be able to compute quantities of interest such as drag and downforce. To obtain this mean velocity, the instantaneous velocity profiles need to be averaged out.

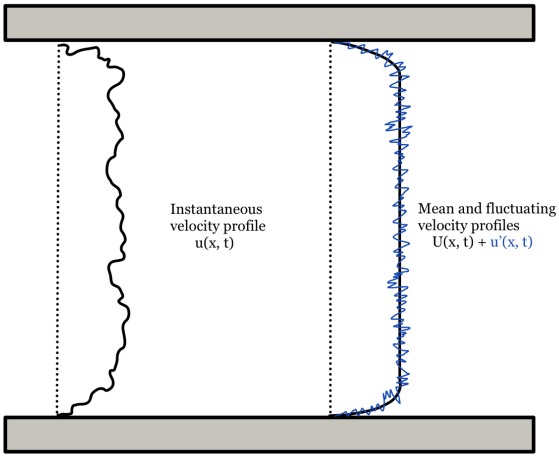


Figure 2.2: Diagram depicting the Reynolds decomposition of an instantaneous velocity profile of a turbulent flow across a pipe cross-section into a mean and fluctuating flow profile. This diagram has been adapted from [2].

Modeling the velocity of any type of flow, whether laminar or turbulent, is achieved by solving a set of non-linear, second-order partial differential equations known as the **Navier-Stokes** equations. These equations describe the conservation of momentum in the flow in 3D space and time. Their formulation for an incompressible fluid, along with the continuity equation for mass conservation, are given below:

$$\frac{\partial u_i}{\partial x_i} = 0 \tag{2.3}$$

[1]The following explanations have been adapted from Wilcox's book on Turbulence modeling for CFD [20].

$$\rho\frac{\partial u_i}{\partial t}+\rho u_j\frac{\partial u_i}{\partial x_j}=-\frac{\partial p}{\partial x_i}+\frac{\partial t_{ij}}{\partial x_j} \tag{2.4}$$

where the u_i is the instantaneous velocity, x_i the position, t is the time, p is the pressure, ρ is the density and t_{ij} is the viscous stress tensor.

To obtain only the mean velocity profiles, the Navier-Stokes equations can thus be subjected to Reynolds decomposition. Reynolds proposed three different approaches to averaging: time, spatial, and ensemble averaging. The turbulent flows generally modeled in engineering applications can be considered as being statistically stationary. This means that the different statistical quantities that describe the flow, such as the mean velocity, do not vary very significantly in time. For these types of flow, time averaging is the preferred type of Reynolds decomposition [20]. The time average $F_T(x)$ of an instantaneous flow variable $f(x,t)$ in such a flow can thus be expressed as:

$$F_T(x)=\lim_{T\to\infty}\frac{1}{T}\int_t^{t+T}f(x,t)dt \tag{2.5}$$

Time averaging the Navier-Stokes equations results in the well-known **Reynolds-Averaged Navier-Stokes** (RANS) equations:

$$\rho\frac{\partial U_i}{\partial t}+\rho U_j\frac{\partial U_i}{\partial x_j}=-\frac{\partial p}{\partial x_i}+\frac{\partial}{\partial x_j}(2\mu S_{ij}-\rho\overline{u_j'u_i'}) \tag{2.6}$$

Due to the averaging process, these equations have lost a significant amount of information regarding the turbulent fluctuations present in the flow, leading to the L1 uncertainties often discussed in turbulence modeling [9]. The term $\overline{u_j'u_i'}$ in these equations is known as the **Reynolds-stress tensor** τ_{ij}:

$$\tau_{ij}=\overline{u_j'u_i'} \tag{2.7}$$

This tensor can be physically interpreted as additional stresses acting on the flow as a result of the turbulent velocity fluctuations. It is symmetric, meaning that $\tau_{ij}=\tau_{ji}$, and thus has six independent components. Time averaging the Navier-Stokes equations has therefore introduced six new unknown quantities. With only 4 available equations, the system of equations lacks closure, and additional equations are required to solve for these unknowns [20]. This unresolved aspect is known as the **Closure problem**, forming the central challenge around which most RANS turbulence modeling efforts revolve.

In 1877, Joseph Boussinesq proposed his famous eddy-viscosity hypothesis [22], which postulates that the Reynolds-stress tensor can be related to the mean strain rate S_{ij}, through an **eddy viscosity** term ν_t:

$$\tau_{ij}=2\nu_t S_{ij} \tag{2.8}$$

This hypothesis essentially assumes that turbulent stresses can be modeled similarly to molecular viscosity [20]. The assumptions made in modeling the Reynolds-stress tensor are the source of the L2 uncertainties in RANS turbulence modeling.

Attempting to derive a functional expression for ν_t became the primary focus of the turbulence community in the years following Boussinesq's study. This pursuit has resulted in the development of several turbulence models, which can be categorized into four main types: algebraic models, one-equation models, two-equation models, and second-order closure models.

2.3. The Development of the $k-\omega$ SST Turbulence Model

The turbulence model used throughout this study is the $k-\omega$ SST model, a two-equation model first introduced by Menter in 1994 [23]. The following sections will describe the development efforts behind this model, its performance, and the approaches used in this study to address its inconsistencies.[2]

[2] A large part of the explanation in the following sections has been adapted from Wilcox's book on Turbulence Modelling for CFD [20] and Menter's original paper presenting the $k-\omega$ SST model [23].

2.3.1. The Turbulent Kinetic Energy Transport Equation

Two-equation models are all based on the following two main concepts: the Boussinesq approximation of the Reynolds-stress tensor and the kinetic energy per unit mass of the turbulent fluctuations, referred to as the **turbulent kinetic energy** k. These models use k to compute ν_t, as will be later shown in this section. The transport of k in the flow is defined as follows:

$$k = \frac{1}{2}\overline{u_i' u_i'} = \frac{1}{2}(\overline{u'^2} + \overline{v'^2} + \overline{w'^2}) \tag{2.9}$$

Based on the above formulation, it is very clear that k can be directly computed from the trace of the Reynolds-stress tensor: $\tau_{ii} = -\rho\overline{u_i' u_i'} = -2\rho k$. Since the trace of the mean strain rate for an incompressible flow is equal to zero ($S_{ii} = 0$), the expression of the Reynolds-stress tensor given in equation (2.8) needs to be modified so that when $S_{ii} = 0$, $\tau_{ii} = -2\rho k$. This leads to the standard formulation of the Reynolds-stress tensor used by two-equation turbulence models:

$$\tau_{ij} = 2\mu_t S_{ij} - \frac{2}{3}\rho k \delta_{ij} \tag{2.10}$$

The transport equation of k is formulated in the following way for an incompressible flow:

$$\rho\frac{\partial k}{\partial t} + \rho\frac{\partial (U_j k)}{\partial x_j} = \tau_{ij}\frac{\partial U_i}{\partial x_j} - \rho\epsilon + \frac{\partial}{\partial x_j}\left[\mu\frac{\partial k}{\partial x_j} - \frac{1}{2}\rho\overline{u_i' u_i' u_j'} - \overline{p' u_j'}\right] \tag{2.11}$$

The two terms on the left-hand side of this equation describe the rate of change of k over time and its **convection** in the flow. The first term on the right-hand side of the equation is known as the **production** of k which describes the rate at which the kinetic energy of the mean flow is transferred into turbulent kinetic energy. The subsequent right-hand side term is known as the **dissipation** of k. This dissipation accounts for the loss of turbulent kinetic energy as it converts into thermal internal energy [20]. The third term is made up of three components. The first one is the **molecular diffusion** term, which describes the diffusion of k due to natural molecular transport in the flow. The second term is a triple velocity correlation known as the **turbulent transport** term, which describes the transport of k due to turbulent fluctuations in the flow. The last term is the **pressure diffusion** term, which describes the transport of k due to the correlation between pressure and velocity fluctuations in the flow [20]. The quantity ϵ, which account for the dissipation of k, is defined as follows:

$$\epsilon = \nu\overline{\frac{\partial u_i'}{\partial x_k}\frac{\partial u_i'}{\partial x_k}} \tag{2.12}$$

Closing the k transport equation is challenging as the dissipation, turbulent transport, and pressure diffusion all depend upon fluctuating velocity components. Since finding an approximation for the pressure diffusion term turns out to be very challenging, the solution found by the turbulence modeling community is to group this term with the turbulent transport term, so that the sum of the two is assumed to behave as a gradient-transport process [20]:

$$\frac{1}{2}\rho\overline{u_i' u_i' u_j'} - \overline{p' u_j'} = \sigma_k \mu_t \frac{\partial k}{\partial x_j} \tag{2.13}$$

The final form of the transport equation for k for an incompressible flow is given below:

$$\rho\frac{\partial k}{\partial t} + \rho\frac{\partial (U_j k)}{\partial x_j} = \tau_{ij}\frac{\partial U_i}{\partial x_j} - \rho\epsilon + \frac{\partial}{\partial x_j}\left[(\mu + \sigma_k\mu_t)\frac{\partial k}{\partial x_j}\right] \tag{2.14}$$

where σ_k is a closure coefficient which is assumed to be constant. The only quantity that remains unresolved is the dissipation term ϵ. The search for a functional expression for ϵ gave rise to the development of various two-equation models, as will be further clarified in the subsequent sections. The modeling assumptions made to determine this final formulation of the above k equation and the model equation for ϵ are the source of the L3 uncertainties in RANS turbulence modeling.

2.3.2. The $k-\omega$ Model

The $k-\omega$ model, proposed in 1942 by Kolmogorov, is the first two-equation model ever developed. He postulated that to compute ϵ and close the k transport equation, one must solve a second transport equation for a quantity known as the **specific dissipation rate**, ω. It is not entirely clear how he derived this quantity, however, in his book on Turbulence Modelling for CFD [20], Wilcox argues that he must have done so through dimensional arguments. The equation that Kolmogorov derived for ω is the following:

$$\rho\frac{\partial\omega}{\partial t}+\rho\frac{\partial(U_j\omega)}{\partial x_j}=\beta\rho\omega^2+\frac{\partial}{\partial x_j}\left[\sigma\mu_t\frac{\partial\omega}{\partial x_j}\right] \tag{2.15}$$

where β and σ are closure coefficients. One of the main flaws of this equation is that it lacks a production term. It is assumed that Kolmogorov omitted this production term because he related ω with the smallest scales of turbulence, assuming that it has no direct interaction with the mean flow. However, in the years following the release of the model, it was discovered that it is the largest scales of turbulence that determine its time scale and its rate of dissipation [20]. The second significant flaw is that the equation does not account for molecular diffusion. This limitation implies that the equation can only describe high Reynolds number flows and cannot be integrated through the viscous sub-layer. Therefore, subsequent developments of the $k-\omega$ model focused on addressing these flaws leading to what is considered as today's standard version of the $k-\omega$ model. This model computes ν_t as the ratio between k and ω.

$k-\omega$ Model

Turbulent Kinetic Energy Equation:

$$\rho\frac{\partial k}{\partial t}+\rho\frac{\partial(U_j k)}{\partial x_j}=\tau_{ij}\frac{\partial U_i}{\partial x_j}-\beta^*\rho\omega k+\frac{\partial}{\partial x_j}\left[(\mu+\sigma_k\mu_t)\frac{\partial k}{\partial x_j}\right] \tag{2.16}$$

Specific Dissipation Rate Equation:

$$\rho\frac{\partial\omega}{\partial t}+\rho\frac{\partial(U_j\omega)}{\partial x_j}=\frac{\gamma\omega}{k}\tau_{ij}\frac{\partial U_i}{\partial x_j}-\beta\rho\omega^2+\frac{\partial}{\partial x_j}\left[(\mu+\sigma_\omega\mu_t)\frac{\partial\omega}{\partial x_j}\right] \tag{2.17}$$

Eddy Viscosity Definition:

$$\nu_t=\frac{k}{\omega} \tag{2.18}$$

Model Coefficients:

$$\gamma=\frac{5}{9},\quad \beta=\frac{3}{40},\quad \sigma_k=0.5,\quad \sigma_\omega=0.5,\quad \beta^*=0.09 \tag{2.19}$$

Auxiliary Equations:

$$\epsilon=\beta^*\omega k\quad\text{and}\quad l=\frac{k^{1/2}}{\omega} \tag{2.20}$$

Two important aspects of this model formulation need to be addressed. First, a production term has been added to the ω equation to account for the deficiency in Kolmogorov's initial formulation. This term is directly related to the production term of the k equation. Second, several new model coefficients have been introduced. These coefficients have been determined based on a combination of dimensional analysis and experimental observations. Since the methods used to obtain the values of these coefficients assume homogeneous turbulence and primarily focus on boundary layer physics, these model coefficients cannot be assumed to be universal and accurately represent every type of turbulent flow. Uncertainties in these coefficients are the cause of the L4 uncertainties in RANS turbulence modeling. Finally, the standard formulation of this model can integrate through the viscous sub-layer and therefore can be used to model both low- and high-Reynolds number flows.

The $k-\omega$ model is known to perform very well in describing boundary layers under both favorable and adverse pressure gradients [24]. Furthermore, due to its simplicity, the $k-\omega$ model is superior to the other two-equation models in terms of numerical stability during a simulation. However, this model shows a strong sensitivity to the free-stream values specified for ω [23]. In terms of modeling boundary layers

undergoing separation, this free-stream sensitivity has a negative effect on the predictions of separation and reattachment locations.

2.3.3. The $k-\epsilon$ Model

The $k-\epsilon$ model is a two-equation model that directly solves a transport equation for ϵ. Therefore, it has the advantage over the $k-\omega$ model of not being sensitive to the free-stream values of ω. The standard form of the model is the one proposed by Jones and Launder in their 1972 paper [25]. However, the model coefficients associated with this standard formulation are taken from the study performed by Launder and Sharma in 1974 [26].

$k-\epsilon$ Model

Turbulent Kinetic Energy Equation:

$$\rho\frac{\partial k}{\partial t}+\rho\frac{\partial(U_j k)}{\partial x_j}=\tau_{ij}\frac{\partial U_i}{\partial x_j}-\rho\epsilon+\frac{\partial}{\partial x_j}\left[(\mu+\mu_t/\sigma_k)\frac{\partial k}{\partial x_j}\right] \tag{2.21}$$

Dissipation Rate Equation:

$$\frac{\partial(\rho\epsilon)}{\partial t}+\frac{\partial(\rho U_j\epsilon)}{\partial x_j}=C_1\frac{\epsilon}{k}\tau_{ij}\frac{\partial U_i}{\partial x_j}-C_2\rho\frac{\epsilon^2}{k}+\frac{\partial}{\partial x_j}\left[(\mu+\mu_t/\sigma_\epsilon)\frac{\partial\epsilon}{\partial x_j}\right] \tag{2.22}$$

Eddy Viscosity Definition:

$$\nu_t=C_\mu k^2/\epsilon \tag{2.23}$$

Model Coefficients:

$$\sigma_k=1.0,\quad \sigma_\epsilon=1.3,\quad C_1=1.44,\quad C_2=1.92,\quad C_\mu=0.09 \tag{2.24}$$

Auxiliary Equations:

$$\omega=\epsilon/(C_\mu k)\text{and}\quad l=C_\mu k^{3/2}/\epsilon \tag{2.25}$$

The above model formulation is often referred to as the high-Reynolds number formulation as it does not provide integration through to the viscous sub-layer. Therefore, a low-Reynolds number formulation has been developed, which essentially dampens the C_1, C_2, and C_μ coefficients close to the wall. The damping functions, as proposed by Launder and Sharma, are formulated as follows:

$$f_1=1,\quad f_2=1-0.3\exp(-Re_T^2),\quad f_\mu=\exp\left(\frac{-3.4}{(1+(Re_T/50)^2}\right) \tag{2.26}$$

where Re_T is the turbulence Reynolds number. Re_T is small within the viscous sub-layer, where viscous effects are predominant, and it increases away from the wall. Consequently, these damping functions all approach one in the free stream, restoring the high-Reynolds number formulation. The damping function f_1 is equal to one, as Launder and Jones observed no significant improvement with a damping function for C_1, which is used to calculate the production term in the dissipation rate equation. The damping function f_μ decreases the turbulent viscosity μ_t, allowing the laminar viscosity μ to dominate in the viscous sub-layer. Lastly, f_2 decreases the dissipation of ϵ near the wall, thereby increasing the dissipation of k in that region. The model coefficients of the $k-\epsilon$ model and the damping functions were determined using a combination of dimensional analysis and experimental investigations.

The $k-\epsilon$ model offers the advantage of not having a free-stream sensitivity like the $k-\omega$ model. However, it remains inaccurate for adverse pressure gradient flows as shown by the study of Wilcox [27] and that of Rodi and Scheuerer [28]. Additionally, it relies on damping functions to integrate through the viscous sub-layer, which introduces additional modeling uncertainties.

2.3.4. The $k-\omega$ SST Model

The $k-\omega$ SST model, where SST stands for Shear Stress Transport, was proposed by Menter in 1994 [23]. This model is a combination of the $k-\epsilon$ and $k-\omega$ models. It takes advantage of the free-stream

insensitivity of the $k-\epsilon$ model and the superior performance of the $k-\omega$ model in describing the behavior of boundary layers under adverse pressure gradients, as well as its ability to integrate through the viscous sub-layer without relying on damping functions. The $k-\omega$ SST model differentiates itself from the other two-equation turbulence models by accounting for the transport of turbulent shear stress, the significance of which will be further clarified below.

The $k-\omega$ SST model has been designed to incorporate the $k-\omega$ model formulation, as described in Section 2.3.2, within the viscous sub-layer and the log-layer. On the other hand, it relies on the standard high-Reynolds number formulation of the $k-\epsilon$ model, as outlined in Section 2.3.3, in the outer wake region of boundary layers and free shear layers [23]. It is important to note that the $k-\epsilon$ model is re-expressed into a $k-\omega$ model formulation before it is used to obtain the final form of the $k-\omega$ SST model. This ensures consistency and compatibility between the formulations. This transformation is achieved by starting with the $k-\omega$ model formulation and defining $\epsilon = \beta * \omega k$, leading to the following re-expression of the $k-\epsilon$ model:

$$\rho\frac{\partial k}{\partial t} + \rho\frac{\partial (U_j k)}{\partial x_j} = \tau_{ij}\frac{\partial U_i}{\partial x_j} - \beta^*\rho\omega k + \frac{\partial}{\partial x_j}\left[(\mu + \sigma_{k2}\mu_t)\frac{\partial k}{\partial x_j}\right] \tag{2.27}$$

$$\rho\frac{\partial \omega}{\partial t} + \rho\frac{\partial (U_j \omega)}{\partial x_j} = \frac{\gamma_2}{\nu_t}\tau_{ij}\frac{\partial U_i}{\partial x_j} - \beta_2\rho\omega^2 + \frac{\partial}{\partial x_j}\left[(\mu + \sigma_{\omega 2}\mu_t)\frac{\partial \omega}{\partial x_j}\right] + 2\rho\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j} \tag{2.28}$$

This reformulation leads to an additional term on the right-hand side of the ω equation, referred to as the **cross-diffusion** term. This term serves as the distinguishing factor between the free-stream sensitivity of the $k-\epsilon$ and $k-\omega$ models. A study led by Menter in [29] found that introducing the cross-diffusion term into the $k-\omega$ model can eliminate its free-stream sensitivity. However, it is important to note that including this term renders the $k-\omega$ model formally identical to the $k-\epsilon$ model, thereby removing its ability to accurately predict the behavior of boundary layers under adverse pressure gradients. This highlights the advantage of the $k-\omega$ SST model, which preserves the $k-\omega$ formulation in the near-wall region while exploiting the free-stream insensitivity of the $k-\epsilon$ model.

The formulation of the $k-\omega$ SST model provided below is based on the version presented in Menter's 2003 paper [30]. Several features of this model need to be addressed. First, a limiter for the production term of the k equations is used to prevent the build-up of turbulence in regions where flow is stagnant [30]. Second, there is an eddy viscosity limiter as seen in equation (2.31). Menter introduced this to better capture the effects of turbulent shear stress transport. The eddy viscosity limiter is based on Bradshaw's observations that the turbulent shear stress is proportional to k in the wake region of boundary layers [31].

The classical formulation of the eddy viscosity in two-equation models ($\nu_t = k/\omega$) does not take this proportionality into account. Therefore, the standard $k-\omega$ model which can accurately predict the eddy viscosity in the log-layer region of the boundary layer, ends up overestimating this quantity in the outer wake of the boundary layer [23]. This overestimation leads to an overprediction of turbulent shear stress and results in premature predictions of the flow reattachment locations.

$k-\omega$ SST Model

Turbulent Kinetic Energy Equation:

$$\rho\frac{\partial k}{\partial t}+\rho\frac{\partial(U_j k)}{\partial x_j}=\hat{P}_k-\beta^*\rho\omega k+\frac{\partial}{\partial x_j}\left[(\mu+\sigma_k\mu_t)\frac{\partial k}{\partial x_j}\right] \tag{2.29}$$

Specific Dissipation Rate Equation:

$$\rho\frac{\partial \omega}{\partial t}+\rho\frac{\partial(U_j \omega)}{\partial x_j}=\frac{\gamma}{\nu_t}\hat{P}_k-\beta\rho\omega^2+\frac{\partial}{\partial x_j}\left[(\mu+\sigma_\omega\mu_t)\frac{\partial \omega}{\partial x_j}\right]+2\rho(1-F_1)\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j} \tag{2.30}$$

Eddy Viscosity Definition:

$$\nu_t=\frac{a_1 k}{\max(a_1\omega, SF_2)} \tag{2.31}$$

Model Coefficients:
Let Φ_1 represent the coefficients in the $k-\omega$ model and Φ_2 those in the transformed $k-\epsilon$ model. The coefficients Φ in the $k-\omega$ SST model are found from : $\Phi=F_1\Phi_1+(1-F_1)\Phi_2$.

The Φ_1 set of coefficients:

$$\sigma_{k1}=0.85,\quad \sigma_{\omega 1}=0.5,\quad \beta_1=0.0750,\quad \beta^*=0.09,\quad \gamma_1=5/9 \tag{2.32}$$

The Φ_2 set of coefficients:

$$\sigma_{k2}=1.0,\quad \sigma_{\omega 2}=0.856,\quad \beta_2=0.0828,\quad \beta^*=0.09,\quad \gamma_2=0.44 \tag{2.33}$$

Auxiliary Equations:

$$\hat{P}_k=\min\left(\tau_{ij}\frac{\partial U_i}{\partial x_j}, 10\beta^* k\omega\right) \tag{2.34}$$

$$F_1=\tanh(arg_1^4)\quad \text{where}\quad arg_1=\min\left[\max\left(\frac{\sqrt{k}}{\beta^*\omega y}, \frac{500\nu}{y^2\omega}\right), \frac{4\rho\sigma_{\omega_2}k}{CD_{k\omega}y^2}\right] \tag{2.35}$$

$$CD_{k\omega}=\max\left(2\rho\sigma_{\omega_2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i}, 10^{-20}\right) \tag{2.36}$$

$$F_2=\tanh(arg_2^2)\quad \text{where}\quad arg_2=\max\left(\frac{2\sqrt{k}}{\beta^*\omega y}, \frac{500\nu}{y^2\omega}\right) \tag{2.37}$$

To satisfy Bradshaw's observations, the limiter introduced in the eddy viscosity formulation of the $k-\omega$ SST model restricts the eddy viscosity, limiting the turbulent shear stress in the wake region. In this new eddy viscosity formulation, as given in equation (2.31), F_2 is a blending function which is zero for free shear flows and one for boundary layer flows.

The F_2 blending function shares a similar structure with the F_1 blending function, which is used in transitioning between the $k-\omega$ and $k-\epsilon$ formulations within the $k-\omega$ SST model. The F_1 function is equal to one in the viscous sub-layer and log-layer, and zero in the wake region of the boundary layer. In equation (2.35), as the wall distance y increases the terms tend toward zero which effectively makes F_1 equal to zero far away from the wall.

Despite significant improvements over other two-equation models [30], the $k-\omega$ SST model is not infallible and does not achieve 100% accuracy for every flow scenario. Specifically, in adverse pressure gradient flows, it tends to underestimate the turbulent shear stress in the separated shear layer formed after boundary layer detachment from the surface, leading to a delayed prediction of the flow reattachment location [32].

2.4. Extracting Model-Form Errors of RANS Equations

L1 uncertainties in RANS turbulence modeling are inherent in the derivation of the RANS equations, so turbulence modeling efforts typically concentrate on addressing the other three levels of uncertainties. As previously discussed, L4 uncertainties arise due to errors in the estimation of the turbulence model closure coefficients. Since these coefficients are calibrated for specific flow regions, tuning them to fit experimental results must be done only in certain regions of the flow domain. With each turbulence model having multiple coefficients, tuning their values becomes challenging and highly case-specific. Consequently, most data-driven turbulence modeling efforts have focused on addressing only L2 and L3 uncertainties, which is also the focus of this study. The first step in correcting for these uncertainties is to extract the model-form error in the Reynolds-stress tensor and the $k-\omega$ SST model from high-fidelity data (LES or DNS simulation data). To achieve this, the methodology proposed by Schmelzer et al. is applied [19].

As previously discussed, the Reynolds-stress tensor τ_{ij} based on the Boussinesq approximation is formulated in the following way:

$$\tau_{ij} = 2k\left(b_{ij} + \frac{1}{3}\delta_{ij}\right) \tag{2.38}$$

where $b_{ij} = -\frac{\nu_t}{k}S_{ij}$. This tensor can be decomposed into an anisotropic component $a_{ij} = 2kb_{ij}$ and an isotropic component $\frac{2}{3}k\delta_{ij}$. Only a_{ij} is effective in capturing the momentum transport in different flow directions, as the isotropic component is absorbed in a modified mean pressure and is thus responsible for the pressure-like behavior of turbulence [19].

To identify the model-form errors, residuals are computed between the predictions of the original $k-\omega$ SST turbulence model (referred to as the Baseline model) and the high-fidelity data (HF). This high-fidelity data contains values for the mean velocity field U_{HF}, the Reynolds-stress tensor $\tau_{ij,HF}$, and the turbulent kinetic energy k_{HF}. Since only the anisotropic part of the Reynolds-stress tensor is expected to differ between $\tau_{ij,HF}$ and the τ_{ij} formulation under the Boussinesq approximation, one can compute a b_{ij}^{Δ} residual. To compute this residual, first $a_{ij,HF}$ and $a_{ij,Boussinesq}$ are computed given the high fidelity data:

$$a_{ij,HF} = \tau_{ij,HF} - \frac{2}{3}\delta_{ij}k_{HF}, \quad a_{ij,Boussinesq} = 2kb_{ij,Boussinesq} = -2\nu_t S_{ij} \tag{2.39}$$

The residual between the two can be found from: $a_{ij}^{\Delta} = a_{ij,HF} - a_{ij,Boussinesq}$, so that $b_{ij}^{\Delta} = \frac{a_{ij}^{\Delta}}{2k_{HF}}$. Therefore in the augmented model, b_{ij} now equals:

$$b_{ij} = -\frac{\nu_t}{k}S_{ij} + b_{ij}^{\Delta} \tag{2.40}$$

To calculate this augmented form of b_{ij}, the eddy viscosity ν_t needs to be computed. The $k-\omega$ SST model computes the eddy viscosity according to equation (2.31), which requires both k and ω to be known. While k is available from the high-fidelity data, ω is not. Therefore, to obtain ω, an approach known as k-corrective frozen-RANS is used. This approach iteratively solves the ω turbulence model equation while all the remaining variables are frozen. Since the production term of the ω equation is based on the production term of the k equation, which is now altered due to the augmented b_{ij} equation, and there still exists a model-form error in the k equation that is not addressed solely by correcting the model-form error of the Reynolds-stress tensor, the residual of the k equation is also computed. This residual is equivalent to an additive correction term defined as R in both the ω and k equations. This leads to the Augmented $k-\omega$ SST model formulation stated below.

To ensure that the R and b_{ij}^{Δ} correction fields obtained through the k-corrective frozen RANS approach effectively address the L2 and L3 RANS uncertainties, they are implemented in a simulation where the Augmented $k-\omega$ SST model is executed using these fields. This validation procedure is referred to as a Propagation simulation. If these fields are valid, the simulation should be able to achieve an almost perfect reconstruction of the high-fidelity data fields.

Augumented $k-\omega$ SST Model

Turbulence Kinetic Energy Equation:

$$\rho\frac{\partial k}{\partial t}+\rho\frac{\partial(U_j k)}{\partial x_j}=\hat{P}_k+\sigma R-\beta^*\rho\omega k+\frac{\partial}{\partial x_j}\left[(\mu+\sigma_k\mu_t)\frac{\partial k}{\partial x_j}\right] \tag{2.41}$$

Specific Dissipation Rate Equation:

$$\rho\frac{\partial \omega}{\partial t}+\rho\frac{\partial(U_j \omega)}{\partial x_j}=\frac{\gamma}{\nu_t}(\hat{P}_k+\sigma R)-\beta\rho\omega^2+\frac{\partial}{\partial x_j}\left[(\mu+\sigma_\omega\mu_t)\frac{\partial \omega}{\partial x_j}\right]+2\rho(1-F_1)\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j} \tag{2.42}$$

The production term in the above equations and the Reynolds-stress tensor are augmented by b_{ij}^{Δ}:

$$\hat{P}_k=\min\left(-2k\left(b_{ij,Boussinesq}+\sigma b_{ij}^{\Delta}\right)\frac{\partial U_i}{\partial x_j},10\beta^*\omega k\right) \tag{2.43}$$

$$\tau_{ij}=2k\left(\left(b_{ij,Boussinesq}+\sigma b_{ij}^{\Delta}\right)+\frac{1}{3}\delta_{ij}\right) \tag{2.44}$$

The other terms and model coefficients in the above equations are the same as the ones listed for the $k-\omega$ SST Baseline model in Section 2.3.4.

The σ variable in front of the b_{ij}^{Δ} and R corrections is used to enable or disable corrections in different parts of the computational domain. The RITA/TI classifier developed in this study computes values of 1 for σ in the shear layer that forms when flow separates from the surface and values of 0 everywhere else in the domain. Therefore, this activates the b_{ij}^{Δ} and R corrections only in the shear layer cluster. The RITA/TI classifier is further discussed in Section 4.3.

2.5. Modeling Ansatz for b_{ij}^{Δ} and R

The k-corrective frozen RANS approach discussed above provides two correction fields: one for b_{ij}^{Δ} and another for R. However, these correction fields are typically specific to the RANS simulations of the case from which they were extracted. To make them applicable to other test cases, this study uses a methodology that involves deriving symbolic model expressions from these correction fields using deterministic symbolic regression. This process is further elaborated in Section 2.6. To infer these symbolic model expressions, a modeling ansatz is required.

The modeling ansatz used in this study is a slightly modified version of the approach presented in the study by Schmelzer et al. [19]. This ansatz is rooted in Pope's effective-viscosity hypothesis, which postulates that the anisotropy of the Reynolds-stress tensor is not solely determined by the mean strain rate tensor, as suggested by Boussinesq, but also depends on the rotation rate tensor Ω_{ij} [17]. According to this hypothesis, the most general form of the anisotropic part of the Reynolds-stress tensor can be expressed as:

$$b_{ij}=\sum_{n=1}^{N}T_{ij}^{(n)}\alpha_n(I_1,...,I_m) \tag{2.45}$$

where $T_{ij}^{(n)}$ are nonlinear base tensors and I_m are the corresponding invariants. For 2D cases, only the first three tensors form a linear independent basis and only the first two invariants are nonzero [19]:

$$T_{ij}^{(1)}=S_{ij},\quad T_{ij}^{(2)}=S_{ik}\Omega_{kj}-\Omega_{ik}S_{kj},\quad T_{ij}^{(3)}=S_{ik}S_{kj}-\frac{1}{3}\delta_{ij}S_{mn}S_{nm},\quad I_1=S_{mn}S_{nm},\quad I_2=\Omega_{mn}\Omega_{nm} \tag{2.46}$$

This modeling ansatz is extended to b_{ij}^{Δ} with a slight modification: the α_n coefficients are not only functions of the invariants but also functions of several input features computed from ratios between different

flow variables. These features, detailed in Section 4.4, are used to account for the local turbulence characteristics. This modification aims to ensure that the models align more closely with the underlying physics. All input features, including scalar features, invariants and basis tensors, are computed from the flow variables of the converged and validated propagated solution. The modified modeling ansatz is thus formulated as:

$$b_{ij}^{\Delta} = \sum_{n=1}^{N} T_{ij}^{(n)} \alpha_n(I_1, ..., I_m; F1, ...F13) \tag{2.47}$$

where $F = F1, ..., F14$ represent the additional scalar features. The aim of the symbolic regression discussed in section 2.6, is to find functional expressions for the α_n coefficients.

The modeling ansatz for R is based on the fact that this correction term acts to locally increase or decrease the production of k. Therefore, it is modeled in a similar way to the $\hat{P_k}$ term in the k model equation of the Augmented $k-\omega$ SST model (see equation (2.43)):

$$R = 2kb_{ij}^{R} \frac{\partial U_i}{\partial x_j} \tag{2.48}$$

which depends on b_{ij}^{R}, modeled according to the ansatz proposed above for b_{ij}^{Δ} in equation (2.47). Thus, it takes the following form:

$$R = \sum_{n=1}^{N} G_{ij}^{(n)} \alpha_n(I_1, ..., I_m; F1, ...F13) \tag{2.49}$$

where $G_{ij}^{(n)} = 2k\frac{\partial U_i}{\partial x_j} T_{ij}^{(n)}$. It is important to note, that ϵ is also used as an extra basis function for the R field, such that $R = \sum_{n=1}^{N} \alpha_n(I_1, ..., I_m; F_1, ...F_k)\epsilon$. This was introduced following promising results obtained in the study of Steiner et al. [33].

2.6. Model Discovery Through SpaRTA

The modeling ansatz of the b_{ij}^{Δ} and R correction fields requires a functional expression for the α_n coefficients. The approach used in this study to find these expressions is a deterministic symbolic regression method known as Sparse Regression of Turbulent Stress Anisotropy, SpaRTA for short. The following listed methodology behind SpaRTA has been adapted from the study of Schmelzer et al. [19] and Kaj Hoefnagel's Master thesis [8].

As can be seen from equation (2.47), the α_n coefficients depend on the scalar invariants I and scalar features F. Therefore, the first step in SpaRTA is to build a library of candidate functions based on different mathematical combinations of these features and invariants. This is achieved through different mathematical transformations such as addition, multiplication, etc. The input vector of the library takes the form of:

$$\mathbf{B} = \left[1, I_1, I_2, I_1^2, I_2^2, ..., F1, F2, F1^2, F2^2, ...\right]^T \tag{2.50}$$

The final step in constructing the library for regressing models for b_{ij}^{Δ} is to multiply each entry in $\mathbf{B}$ with the base tensors $T_{ij}^{(n)}$:

$$\mathbf{C_{b_{ij}^{\Delta}}} = \left[T_{ij}^{(1)}, T_{ij}^{(2)}, ..., I_2^2 T_{ij}^{(1)}, ..., F2^2 T_{ij}^{(1)}, ...\right]^T \tag{2.51}$$

The library for regressing models for R is constructed in a similar way:

$$\mathbf{C_R} = \left[G_{ij}^{(1)}, G_{ij}^{(2)}, ..., I_2^2 G_{ij}^{(1)}, ..., F_2^2 G_{ij}^{(1)}, ...\right]^T \tag{2.52}$$

To prevent the size of the library from making the regression very computationally expensive, a combination of three approaches is used. Firstly, each entry in $\mathbf{B}$ is limited to a maximum of two invariants or input features per term, ensuring that the overall global degree of the library is 2. Secondly, input features are selected by computing their mutual information (MI) scores with the target of the regression. Features and invariants with high MI scores, indicating a strong dependency with the target, are retained in the input feature set, while those with scores close to zero, indicating complete independence, are discarded. Finally, a technique known as cliqueing is used to further decrease the size of the library. This method calculates the correlation coefficient between various candidate functions and the target data. Candidates with a correlation coefficient of at least 0.99 are grouped into cliques, and from each clique, the simplest algebraic expression is chosen as the representative candidate function. Together, these methods guarantee that the library maintains a manageable size and remains computationally efficient [8].

The target data for the regression is obtained from the b_{ij}^{Δ} and R correction fields. Since this study focuses on deriving model equations for these fields specifically in regions where RANS performs poorly— the shear layer in boundary-layer flows undergoing separation due to adverse pressure gradients— only data extracted from the shear layer cluster, identified following the method presented in Section 4.3, is used for the regression. Consequently, the models obtained using this approach are calibrated to fit the physics of the shear layer and are active only within the shear layer cluster during a simulation. This is achieved by multiplying the b_{ij}^{Δ} and R terms, as presented in the Augmented $k-\omega$ SST model in Section 2.4, by a variable σ. This variable takes values of one or zero, depending on whether a point in the domain is identified by the classifier as being in the shear layer or outside of it, respectively. Multiplying the model corrections by σ ensures that they are turned off outside of the shear layer cluster.

The linear model used to regress the b_{ij}^{Δ} and R target data can be constructed from the linear combinations of candidate functions in the reduced library: $R = C_R\Theta_R$ and $b_{ij}^{\Delta} = C_{b_{ij}}\Theta_{b_{ij}}$, where Θ represents the α_n coefficient vector. A simple least-squares regression would regress very dense coefficient vectors, leading to overly complex model formulations that would overfit the target data. Moreover, due to a certain level of co-linearity between the candidate functions, the coefficient vector is likely to show significant differences in magnitudes between its members. This renders the discovered models unsuitable for CFD solver applications as they increase numerical stiffness and adversely affect solution convergence [19].

In SpaRTA, the regression approach of choice is elastic-net regression, which effectively addresses the challenges associated with constructing linear models for b_{ij}^{Δ} and R. Elastic-net regression combines the strengths of Lasso (based on the l_1-norm) and Ridge (based on the l_2-norm) regularization techniques. The formulation of elastic-net regression involves two key parameters: the mixing parameter a and the regularization weight λ. These parameters play crucial roles in promoting sparsity and reducing the magnitude of coefficient values in the coefficient vector. The regression problem is formulated in the following way:

$$\Theta = \underset{\hat{\Theta}}{\text{arg min}} \|C_{\Delta}\hat{\Theta} - \Delta\|_2^2 + \lambda a\|\hat{\Theta}\|_1 + 0.5\lambda(1-a)\|\hat{\Theta}\|_2^2 \tag{2.53}$$

Lasso regression encourages sparsity by allowing only a few nonzero coefficients and shrinking the rest to zero. On the other hand, Ridge regression enforces small coefficients without setting them to zero, thus identifying correlated candidate functions instead of selecting just one. By combining Lasso and Ridge regularization in elastic-net regression, SpaRTA strikes a balance between sparsity and model complexity. This approach ensures that the discovered models remain parsimonious while capturing the essential relationships in the data. Consequently, the resulting models are well-suited for integration into CFD solvers, as they do not introduce numerical stiffness issues and are less prone to overfitting, thereby enhancing solution convergence and stability.

The mixing parameter a controls the balance between Lasso and Ridge regularization. A value of $a = 1$ corresponds to pure Lasso regression, while $a = 0$ corresponds to pure Ridge regression. Intermediate values of a allow for a combination of both regularization techniques, offering flexibility in handling collinear features and promoting sparsity. The regularization weight λ controls the overall strength of regularization applied to the model. A higher value of λ leads to more aggressive regularization, resulting in simpler models with smaller coefficient values. Conversely, a lower value of λ relaxes the regularization, allowing the model to capture more intricate relationships in the data. In this study, various λ and

a values have been tested to ensure a comprehensive usage of different regularization types.

The regression process generates a series of models based on the candidate functions found in the coefficient vector. To assess the significance of each candidate independently of its magnitude during model selection, all candidates undergo standardization before elastic-net regression. However, an additional step is required to ensure the models have appropriate units. This involves applying Ridge regression using the original, unstandardized candidate functions. Ridge regression helps maintain small refit coefficients, thereby improving stability in the CFD solver. This step, referred to as model inference, is essential for preserving the physical interpretability of the models and ensuring they align with the units of the variables in the CFD simulations [19].

The final models obtained for R and b_{ij}^{Δ} are tested a-posteriori in the CFD solver to assess their true performance. While some models may exhibit high R^2 regression scores, their behavior in a flow simulation and their generalizability cannot be guaranteed solely based on regression metrics. Therefore, subjecting the models to actual flow simulations provides crucial insights into their performance under realistic conditions and their ability to generalize beyond the training data.

2.7. Comparison Tools Between Baseline and Augmented RANS

To compare the performance of the Baseline RANS with that of the high-fidelity data and the Augmented RANS based on the R and b_{ij}^{Δ} symbolic models identified using SpaRTA, various flow variables, and derived quantities are examined:

1. **Skin friction coefficient** C_f: The skin friction coefficient is defined as the ratio of the shear stress at the wall to the dynamic pressure of the free stream flow. It provides information about flow separation and reattachment. A drop to zero in C_f indicates the onset of flow separation, while its return to zero indicates flow reattachment. It is computed according to the following formula:

$$C_f = \frac{\tau_w}{\frac{1}{2}\rho_\infty v_\infty^2} \tag{2.54}$$

   where τ_w is the wall shear stress, ρ_∞ is the density of the free stream and v_∞ is the free stream velocity.

2. **Pressure coefficient** C_p: This coefficient describes the pressure distribution over a surface, providing insights into areas of flow acceleration or deceleration. It is calculated using the following formulation:

$$C_p = \frac{p - p_\infty}{\frac{1}{2}\rho_\infty v_\infty^2} \tag{2.55}$$

   where p is the pressure at the surface and p_∞ is the free stream pressure.

3. **Profiles of the shear stress component of Reynolds-stress tensor,** $\overline{u_i' u_j'}$: This component quantifies the turbulent shear forces acting perpendicular to the mean flow direction. In practical terms, $\overline{u_i' U_j'}$ reflects the momentum exchange between fluid layers moving at different velocities, contributing to the generation and maintenance of turbulence in the flow. It is typically underpredicted by Baseline RANS, resulting in a delayed reattachment location prediction. It is computed using the following expression:

$$\overline{u_i' u_j'} = -\nu_t \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) \tag{2.56}$$

   For the Augmented RANS, the b_{ij}^{Δ} anisotropy corrections are also taken into account:

$$\overline{u_i' u_j'} = -\nu_t \left(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i} \right) + 2k b_{ij}^{\Delta} \tag{2.57}$$

4. **Profiles of turbulent kinetic energy** k: These profiles are obtained at various locations along the computational domain. They provide crucial insights into areas where k is overpredicted or underpredicted compared to the high-fidelity data across different types of simulations.

5. **Profiles of axial velocity** U_i: These profiles are obtained at various locations along the computational domain. They are insightful for identifying regions of reverse flow (i.e., a recirculation region) and for pinpointing locations of flow separation and reattachment.

3

Computational Methods

3.1. Turbulence Modelling in OpenFOAM

The CFD solver used in this study is the free, open-source software OpenFOAM [34]. The simulations performed with this software are all based on solving the incompressible RANS equations presented in Section 2.2. OpenFOAM uses a finite volume approach to discretize and solve these equations.

3.1.1. Simulation Types

Baseline Simulations: Baseline simulation refers to a RANS simulation performed with the $k-\omega$ SST turbulence model presented in Section 2.3.4 and the Boussinesq approximation of the Reynolds-stress tensor. All baseline simulations are executed with the `simpleFOAM` solver within the OpenFOAM software framework. This solver is specifically designed to simulate steady-state, incompressible turbulent flow simulations based on the SIMPLE (Semi-Implicit Method for Pressure Linked Equations) algorithm.

Frozen Simulations: Frozen simulations are used to extract the model-form error in the k-ω SST model and the Reynolds-stress tensor from high-fidelity data. This is achieved through the k-corrective-frozen-RANS approach detailed in Section 2.4. The outputs of these simulations are the R and b_{ij}^{Δ} correction fields.

Propagation Simulations: Propagation simulations serve to validate the R and b_{ij}^{Δ} correction fields identified by the Frozen simulations. They solve the augmented version of the Reynolds-stress tensor and the k-ω SST turbulence model, as presented in Section 2.4. Consequently, the outputs of these simulations are expected to closely match the high-fidelity data if the correction fields have been accurately extracted during the Frozen simulations.

Model Propagation Simulations: Similar to Propagation simulations, Model Propagation simulations use the augmented version of the Reynolds-stress tensor and the k-ω SST turbulence model. However, they use the symbolic model equations for R and b_{ij}^{Δ} discovered by the SpaRTA regression, as described in Section 2.6, instead of the correction fields identified by the Frozen simulations.

In both Propagation and Model Propagation simulations, corrections can be applied exclusively within the shear layer cluster, identified using the RITA/TI classifier further discussed in Section 4.3. The code infrastructure developed as a part of this study for performing these types of simulations is further elaborated upon in Appendix Section C.

3.1.2. Monitoring Residuals and Convergence Probes

CFD simulations are based on an iterative approach to solving the discretized RANS and k-ω SST model equations. The most common method to assess the convergence of this iterative process is through residuals. These are a measure of the local imbalance of a conserved variable within each mesh cell [35], therefore each equation being solved will have its own residual. The individual residuals for each

cell are added together and normalized so that each equation will have a single value for the residual for the entire mesh domain at every iteration. OpenFOAM offers the possibility to monitor and plot these residuals through time. Generally, a simulation in which the residuals of all monitored flow variables fall below 1 X 10^{-5} is considered converged, however, this criterion varies depending on the complexity of the simulation. In this study, the general aim is to have the residuals fall below 1 X 10^{-5}.

As the residuals provide a global overview of the convergence, sometimes it is beneficial to add point probes through the computational domain to monitor the conserved variables in certain locations over time directly. For example, if oscillations appear in the velocity residual plot, one cannot directly know which region of the domain is responsible for these oscillations. In this study, 3 point probes are used per conserved variable and they are placed in different parts of the computational domain, depending on the type of case being simulated. Table 3.1 provides an overview of the conserved variables whose residuals or values are monitored in the different types of simulations.

Table 3.1: Overview of the monitored variables via residuals and probes in the different types of simulations. Variables between brackets (...) are only monitored via probes.

Simulation	Residual/ Probes
Baseline	U, p, k, ω
Frozen	$\omega, (R, b_{ij}^{\Delta})$
Propagation	U, p, k, ω
Model Propagation	$U, p, k, \omega, (R, b_{ij}^{\Delta})$

3.2. Setup of 2D-Separated Flow Cases

The cases selected for this study show similar physics - a boundary layer separating from a surface under an adverse pressure gradient - but differ in their geometrical setup and the Reynolds number of the flow. Various geometries and Reynolds numbers are chosen to ensure that the classifier of the shear layer and the models discovered with SpaRTA are generalizable and can be applied to a variety of different flow scenarios. These are all 2D cases previously configured either by researchers at the TU Delft Aerodynamics department or other literature studies. Consequently, no mesh sensitivity studies have been performed since the meshes have been previously validated. 2D cases are preferred over 3D cases due to their significantly lower computational demands, which facilitate easier experimentation. Additionally, all selected cases are incompressible, eliminating the added complexity associated with density variations that can influence the pressure distribution and, consequently, separation behavior. It is important to note that for all the cases listed below, the reference pressure is provided in units of [m^2 s^{-2}], as the OpenFOAM `simpleFOAM` solver employed for the various simulations calculates a normalized pressure by dividing the actual pressure by the flow density.

3.2.1. NASA-Hump

The NASA-Hump case is part of NASA's 2D-separated flow validation cases, designed to assess the capabilities of turbulence models in simulating flow separation [36]. Modeled after the Glauert-Goldschmied type body, the computational setup closely follows the real-life experimental setup outlined in Greenblatt et al. [37]. The OpenFOAM case setup has been acquired from the study of Kaj Hoefnagel [8]. The reference Mach number for this case is 0.1, therefore it is low enough to justify the assumption of incompressibility. The case features a turbulent boundary layer developing over a flat plate, accelerating over the hump geometry due to a favorable pressure gradient, then separating from the hump's edge under an adverse pressure gradient, before reattaching and recovering further downstream of the hump.

3.2.1.1 Geometry and Mesh Specifications

Figure 3.1 displays the geometry and mesh of the NASA-Hump case. The downstream domain length of the hump is tailored to accommodate the incoming fully turbulent boundary layer. The hump's chord length c is 0.42 m. Notably, the upstream domain length is shorter than that used in the experimental setup in [37], owing to the availability of high-fidelity data for Frozen simulations only with this shorter configuration. Consequently, the upstream length does not allow for complete turbulent boundary layer recovery. The upper boundary of the domain features a slight contour to address the blockage resulting from the end plates used in experimental measurements. The mesh consists of 51,626 cells, with the

y^+ values listed in Table 3.2, indicating that the first mesh cells are situated in the viscous sub-layer and the boundary layer is fully resolved.

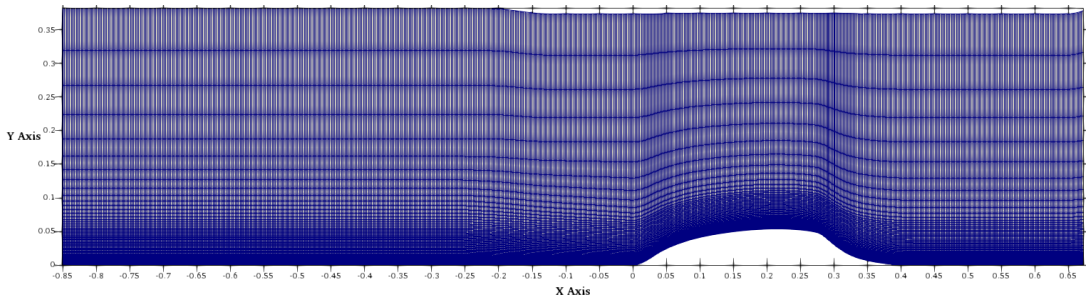


Figure 3.1: NASA-Hump case geometry and mesh layout.

Table 3.2: y^+ values of the NASA-Hump case mesh at the bottom wall boundary of the domain.

Wall Location	y^+ min	y^+ max	y^+ average
Bottom	0.06	2.11	1.45

3.2.1.2 Flow Parameters

An overview of the main flow parameters specified for the NASA-Hump case is given in Table 3.3. This case has the highest Reynolds number equal to 936,000, computed based on the chord length of the hump c and the free-stream reference velocity U_{ref}.

Table 3.3: Overview of the flow parameters specified for the NASA-Hump case.

Transport Property	Parameter	Value
Reynolds number based on chord Length	Re_c	936,000
Kinetic viscosity	ν	1.55 x 10^{-5} m^2 s^{-1}
Free-stream reference velocity	U_{ref}	34.6 m s^{-1}
Reference kinetic energy	k_{ref}	0.00107 m^2 s^{-2}
Reference specific dissipation rate	ω_{ref}	0.118 s^{-1}
Reference pressure	p_{ref}	0 m^2 s^{-2}

3.2.1.3 Initial and Boundary Conditions

The boundary conditions specified for the NASA-Hump case in OpenFOAM are outlined in Table 3.4. These remain consistent across all simulation types used in this study: Baseline, Frozen, Propagation, and Model Propagation. The exact values used for the inlet and outlet `fixedValue` boundary conditions vary depending on the type of simulation under consideration. The Frozen simulation uses LES data fields obtained from the study of Uzun et al. [38].

For the bottom boundary of the domain, which represents the wall, a zero-gradient boundary condition is specified for k via the `kqRWallFunction`. The boundary condition for ω has been derived by Menter for the $k-\omega$ SST model [23]. It sets ω at the wall equal to $\omega = \frac{6\nu_w}{\beta_1 y^2}$, where ν_w is the kinematic viscosity of fluid near the wall, y is the wall-normal distance and β_1 is one of the $k-\omega$ SST model coefficients (see equations in (2.32)). This is implemented in OpenFOAM using the `omegaWallFunction`. The `nutUSpaldingWallFunction` is used to specify the bottom boundary condition for ν_t. This function calculates a continuous ν_t profile up to the wall based on Spalding's law. It is typically used when the y^+ values of the mesh vary across the domain, as is the case here due to the hump geometry distorting the contour of the bottom wall boundary. For the velocity field, a no-slip boundary condition is enforced at the bottom wall. In the experimental setup, no specific surface wall boundary is present on top of the hump. Therefore, the top boundary condition set in the OpenFOAM case setup is symmetry, applied to all flow variables.

Table 3.4: Boundary and initial conditions for the NASA-Hump case for different simulation types: Baseline, Frozen, Propagation (Pr.) and Model Propagation (Model Pr.).

Boundary Conditions					
Location	U [m/s]	p [m^2/s^2]	k [m^2/s^2]	ω [s^{-1}]	ν_t [m^2/s]
Inlet	`fixedValue`	`zeroGradient`	`fixedValue`	`fixedValue`	`calculated`
Outlet	`zeroGradient`	`fixedValue`	`zeroGradient`	`zeroGradient`	`calculated`
Top	`symmetry`				
Bottom	`noSlip`	`zeroGradient`	`kqRWallFunction`	`omegaWallFunction`	`nutUSpaldingWallFunction`
Initial Conditions					
Baseline	[U_{ref} 0 0]	0	k_{ref}	ω_{ref}	0.009
Frozen	$U_{LES,field}$	0	$k_{LES,field}$	ω_{ref}	0.009
Pr.	Baseline simulation outputs.				
Model Pr.	Baseline simulation outputs.				

The initial conditions employed for the various simulation types are also outlined in Table 3.4. In the Baseline simulation, these initial conditions are determined from the flow parameters listed in Table 3.3. For the Frozen simulation, the variables U and k are derived from the LES fields, while ω and ν_t remain consistent with those of the Baseline simulation. Lastly, in both the Propagation and Model Propagation simulations, the initial conditions are extracted from the results of the converged Baseline simulation.

3.2.2. Periodic-Hill

The Periodic-Hill case setup and the high-fidelity LES data have been obtained from the study of Breuer et al., who extensively studied this geometry across various Reynolds numbers [39]. This configuration consists of a series of repeating hills separated by a flat surface region. The spacing between the hills is calculated to enable the flow to reattach to the flat surface post-separation before encountering the next hill.

3.2.2.1 Geometry and Mesh Specifications

The geometry and mesh of the Periodic-Hill case are depicted in Figure 3.2. The dimensions of the domain are L_x=9.0H and L_y=3.3H where H is the hill height non-dimensionalized to 1.

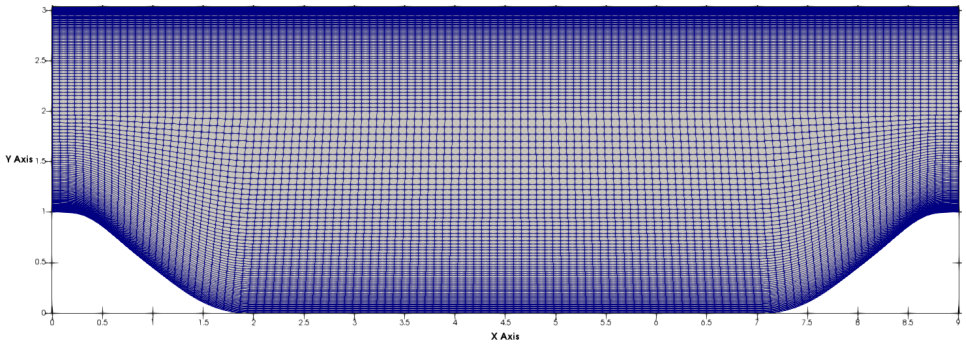


Figure 3.2: Periodic-Hill case geometry and mesh layout.

The mesh consists of 15,600 cells, with the y^+ values listed in Table 3.5. The bottom wall has a slightly higher maximum y^+ value due to the contour of the hill. Nevertheless, the y^+ values for both the top and bottom walls indicate that the mesh is designed to fully resolve the boundary layer. Towards the top boundary of the domain, the mesh is slightly less refined, as the primary physics of interest, namely flow separation and reattachment, occur at the bottom boundary.

Table 3.5: y^+ values of the Periodic-Hill case mesh at the top and bottom wall boundaries of the domain.

Wall Location	**y^+ min**	**y^+ max**	**y^+ average**
Bottom	0.18	2.10	1.03
Top	1.15	1.39	1.26

3.2.2.2 Flow Parameters

An overview of the main flow parameters specified for the Periodic-Hill case is given in Table 3.6. This case has a Reynolds number equal to 10,595, computed based on the hill height H and reference bulk velocity U_{ref}.

Table 3.6: Overview of the flow parameters specified for the Periodic-Hill case.

Transport Property	Parameter	Value
Reynolds number based on hill height	Re_H	10,595
Kinetic viscosity	ν	9.44 x 10^{-5} m^2 s^{-1}
Free-stream reference velocity	U_{ref}	1 m s^{-1}
Reference kinetic energy	k_{ref}	0.00375 m^2 s^{-2}
Reference specific dissipation rate	ω_{ref}	0.110 s^{-1}
Reference pressure	p_{ref}	0 m^2 s^{-2}

3.2.2.3 Initial and Boundary Conditions

The OpenFOAM boundary and initial conditions for this case are outlined in Table 3.7. To simulate the series of hills, periodic boundary conditions are applied at the inflow and outflow of the domain. The boundary conditions for k and ν_t differ from those specified for the NASA-Hump case. Here, k is specified to be zero at the walls. The `nutLowReWallFunction` also sets ν_t to zero and provides an access function to calculate y^+ [40].

Table 3.7: Boundary and initial conditions for the Periodic-Hill case for different simulation types: Baseline, Frozen, Propagation (Pr.) and Model Propagation (Model Pr.).

Boundary Conditions					
Location	U [m/s]	p [m^2/s^2]	k [m^2/s^2]	ω [s^{-1}]	ν_t [m^2/s]
Inlet	`cyclic`	`cyclic`	`cyclic`	`cyclic`	`cyclic`
Outlet	`cyclic`	`cyclic`	`cyclic`	`cyclic`	`cyclic`
Top Wall	`noSlip`	`zeroGradient`	1 x 10^{-15}	`omegaWallFunction`	`nutLowReWallFunction`
Bottom Wall	`noSlip`	`zeroGradient`	1 x 10^{-15}	`omegaWallFunction`	`nutLowReWallFunction`
Initial Conditions					
Baseline	[U_{ref} 0 0]	0	k_{ref}	ω_{ref}	0
Frozen	$U_{LES,field}$	0	$k_{LES,field}$	ω_{ref}	0
Pr.	Baseline simulation outputs.				
Model Pr.	Baseline simulation outputs.				

3.2.3. Curved Backward Facing Step

The Curved Backward Facing Step case, henceforth referred to as CBFS, closely resembles the NASA-Hump and Periodic-Hill cases. It involves a turbulent boundary layer separating from a curved step under an adverse pressure gradient. The key distinction lies in the contour of the step, which has a much gentler curvature compared to the other two cases, thereby promoting the flow to remain attached for longer. The case setup and the high-fidelity LES data have been obtained from the study of Bentaleb et al. [41].

3.2.3.1 Geometry and Mesh Specifications

The geometry and mesh of the CBFS case are depicted in Figure 3.3. The step height H has been non-dimensionalized to 1. At x/H = -7.34, the computational domain inlet, the boundary layer thickness is prescribed to be $0.8H$. The upstream domain length has been designed to allow the boundary layer to recover post-separation fully.

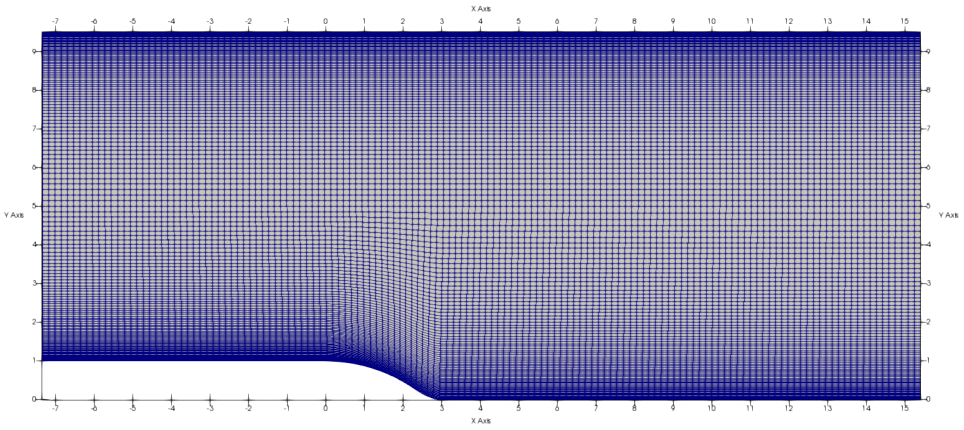


Figure 3.3: CBFS case geometry and mesh layout.

The mesh consists of 21,000 cells, with y^+ values listed in Table 3.8. While this mesh has significantly higher maximum y^+ values compared to other cases, this discrepancy primarily stems from the mesh's lesser refinement towards the outlet of the domain. In the inlet region, around the step contour, and through most of the downstream step area, y^+ values remain below 5, thereby fully resolving the boundary layer.

Table 3.8: y^+ values of the CBFS case mesh at the top and bottom wall boundary of the domain.

Wall Location	y^+ min	y^+ max	y^+ average
Bottom	0.87	5.12	3.63
Top	4.04	6.47	4.46

3.2.3.2 Flow Parameters

An overview of the main flow parameters specified for the CBFS case is given in Table3.9. This case has a Reynolds number equal to 13,700, computed based on the step height H and the free-stream inlet reference velocity U_{ref}.

Table 3.9: Overview of the flow parameters specified for the CBFS case.

Transport Property	Parameter	Value
Reynolds Number based on step height	Re_H	13,700
Kinetic viscosity	ν	7.23 x 10^{-5} m^2 s^{-1}
Free-stream reference velocity	U_{ref}	1 m s^{-1}
Reference kinetic energy	k_{ref}	0.00668 m^2 s^{-2}
Reference specific dissipation rate	ω_{ref}	0.110 s^{-1}
Reference pressure	p_{ref}	0 m^2 s^{-2}

3.2.3.3 Initial and Boundary Conditions

The OpenFOAM boundary and initial conditions for this case are detailed in Table 3.10. These are similar to the ones prescribed for the other 2D-separated flow cases.

3.2.4. APG Case

The APG case stands out among the other 2D-separated flow cases examined in this study due to its unique geometry. It features an incoming fully turbulent boundary layer exposed to an adverse pressure gradient, resulting from domain expansion caused by a modification in the contour of the top boundary of the domain. Post-separation, reattachment of the boundary layer is facilitated by a favorable pressure gradient before it returns to its zero-pressure gradient state. Unlike for the other cases, the adverse pressure gradient is not induced by changes in the bottom wall contour but rather by changes in the top wall contour. The high-fidelity DNS data has been acquired from Coleman et al.'s study [42] and the case has been set up by Tyler Buchanan at the TU Delft Aerodynamics department.

Table 3.10: Boundary and initial conditions for the CBFS case for different simulation types: Baseline, Frozen, Propagation (Pr.) and Model Propagation (Model Pr.).

Boundary Conditions					
Location	U [m/s]	p [m^2/s^2]	k [m^2/s^2]	ω [s^{-1}]	ν_t [m^2/s]
Inlet	`fixedValue`	`zeroGradient`	`fixedValue`	`fixedValue`	`calculated`
Outlet	`zeroGradient`	`fixedValue`	`zeroGradient`	`zeroGradient`	`calculated`
Top Wall	`noSlip`	`zeroGradient`	1 x 10^{-15}	`omegaWallFunction`	`nutLowReWallFunction`
Bottom Wall	`noSlip`	`zeroGradient`	1 x 10^{-15}	`omegaWallFunction`	`nutLowReWallFunction`
Initial Conditions					
Baseline	[U_{ref} 0 0]	0	k_{ref}	ω_{ref}	0
Frozen	$U_{LES,field}$	0	$k_{LES,field}$	ω_{ref}	0
Pr.	Baseline simulation outputs.				
Model Pr.	Baseline simulation outputs.				

3.2.4.1 Geometry and Mesh Specifications

The geometry and mesh of the APG case are depicted in Figure 3.4. The characteristic reference height H has been non-dimensionalized to 1. The upstream and downstream lengths have been designed to allow the boundary layer to fully develop and recover pre- and post-separation. The mesh consists of 92,777 cells, with y^+ values listed in Table 3.11. This mesh is the most refined among all the cases investigated in this study, with y^+ values well below one, effectively resolving the boundary layer.

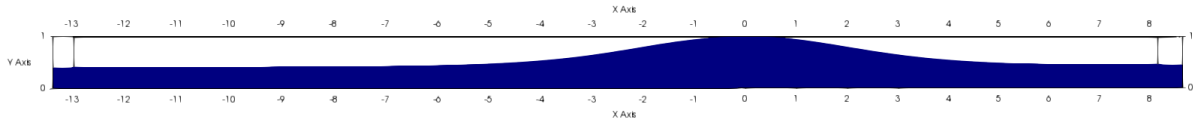


Figure 3.4: APG case geometry and mesh layout.

Table 3.11: y^+ values of the APG case mesh at the bottom wall boundary of the domain.

Wall Location	y^+ **min**	y^+ **max**	y^+ **average**
Bottom	6.614 x 10^{-6}	4.01 x 10^{-5}	1.71 x 10^{-5}

3.2.4.2 Flow Parameters

An overview of the main flow parameters specified for the APG case is given in Table3.12. This case has a Reynolds number equal to 80,000, computed based on the domain maximum height H and the free-stream reference velocity U_{ref}. The reference Mach number for this case is 0.1, similar to the NASA-Hump case.

Table 3.12: Overview of the flow parameters specified for the APG case.

Transport Property	**Parameter**	**Value**
Reynolds number based on domain height	Re_H	80,00
Kinetic viscosity	ν	4.29 x 10^{-4} m^2 s^{-1}
Free-stream reference velocity	U_{ref}	34.3 m s^{-1}
Reference kinetic energy	k_{ref}	4.347 m^2 s^{-2}
Reference specific dissipation rate	ω_{ref}	250 s^{-1}
Reference pressure	p_{ref}	1.01 x 10^5 m^2 s^{-2}

3.2.4.3 Initial and Boundary Conditions

The OpenFOAM boundary and initial conditions for this case are detailed in Table 3.13. The `nutkWallFunction` sets ν_t to zero at the bottom wall boundary of the domain.

Table 3.13: Boundary and initial conditions for the APG case for different simulation types: Baseline, Frozen, Propagation, and Model Propagation.

Boundary Conditions					
Location	U [m/s]	p [m^2/s^2]	k [m^2/s^2]	ω [s^{-1}]	ν_t [m^2/s]
Inlet	`fixedProfile`	`zeroGradient`	`fixedProfile`	`fixedProfile`	`fixedProfile`
Outlet	`zeroGradient`	p_{ref}	`zeroGradient`	`zeroGradient`	`zeroGradient`
Top Boundary	`slip`	`slip`	`slip`	`slip`	`slip`
Bottom Wall	`noSlip`	`zeroGradient`	`kqRWallFunction`	`omegaWallFunction`	`nutkWallFunction`
Initial Conditions					
Baseline	[U_{ref} 0 0]	p_{ref}	k_{ref}	ω_{ref}	0.6
Frozen	$U_{DNS,field}$	p_{ref}	$k_{DNS,field}$	$\omega_{DNS,field}$	$\nu_{t,DNSfield}$
Pr.	Baseline simulation outputs.				
Model Pr.	Baseline simulation outputs.				

4

Clustering Approaches

The approach used in this study to apply local corrections to the k-ω SST model and the Reynolds-stress tensor involves partitioning the flow domain into distinct regions, known as clusters, characterized by identifiable physical features such as shear layers, recirculation regions, and boundary layers. The corrections are selectively applied to the regions where RANS shows the poorest performance. Therefore, it is important to consider the desired clustering output. For separated flows, the primary discrepancy between RANS and high-fidelity data arises from the underprediction of turbulent shear stress in the separated shear layer. Consequently, a successful clustering algorithm should reliably identify this layer across all cases outlined in Section 3.2. Moreover, if the algorithm can identify other regions, it would greatly aid future studies. For instance, effective separation of the boundary layer from the rest of the domain enables avoiding the activation of corrections in this region, potentially impacting the calibration of the k-ω SST turbulence model.

Dividing the flow domain poses a significant challenge due to the continuous nature of the flow and the chaotic nature of turbulence, compared to the discrete and organized nature inherent in most clustering approaches. Establishing the boundaries between regions is particularly difficult. Nonetheless, Figure 4.1 provides a rough sketch of the desired clustering output for the NASA-Hump case.

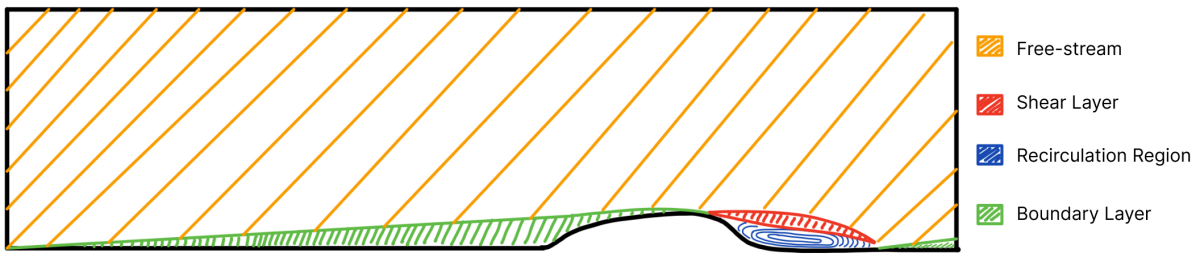


Figure 4.1: Rough sketch of the desired clustering output for the NASA-Hump case.

4.1. K-Means Clustering

4.1.1. Algorithm Overview

K-Means is a popular unsupervised learning clustering algorithm that aims to identify underlying patterns in datasets by grouping together similar data points. The algorithm for identifying these clusters is iterative, attempting to partition the dataset into K (user pre-defined), distinct and non-overlapping clusters of equal variance, where each data point belongs to one cluster only. Data points are assigned to clusters in a way that minimizes the **inertia**, which is the sum of the squared distances between each data point x and the cluster centroid c :

$$\sum_{x \in X} \min_{c \in C} \|x - c\|^2 \tag{4.1}$$

The centroid c is the arithmetic mean of all the data points that belong to that cluster [43]. The algorithm on which K-Means is based on is known as Lloyd's algorithm, further detailed in Algorithm 1. The original algorithm is initialized with K centroids, chosen uniformly at random from the dataset. Next, it assigns each point in the dataset to the nearest centroid and recalculates the cluster centroids by computing the mean value of all the data points assigned to each of the previous cluster centroids. The difference between the new and old cluster centroids is computed, and the algorithm repeats until this difference falls below a specified threshold, ϵ, indicating that the clusters are no longer changing [44], or until the maximum number of iterations has been reached.

While the K-Means algorithm is guaranteed to converge, it may converge to a local minimum, which can affect the clustering performance. This usually occurs if the randomly initialized cluster centroids are too close to each other. To avoid this, one can recompute the algorithm several times with different centroid initializations, which can be very time inefficient [44]. This issue can be addressed by using an initialization method known as `k-means++` [45]. This method ensures that the initialized centroids are distant from each other, which leads to better clustering performance. The exact steps of this initialization procedure are further detailed in Algorithm 1.

Algorithm 1 Lloyd's K-Means Algorithm with `k-means++` Initialization [45] [44]

Input:
Dataset $X = [x_1, \ldots, x_n]$ where $X \in \mathbb{R}^d$, k number of clusters where $K \in \mathbb{N}$
Initialization Based on `k-means++` :
1. Random initialization of center c_1 from X.
2. Choose the next center c_i, where $c_i = x'$ from X, selecting based on the probability $\frac{D(x')^2}{\sum_{x \in X} D(x)^2}$ where $D(x)$ is the shortest distance from a data point x to the closest center already chosen.
3. Iterate step 2 until a total of K centers have been chosen.
Repeat:
1. For each $i \in \{1, \ldots, K\}$, define the cluster C_i to be the set of data points in X that are closer to center c_i than they are to c_j for all $j \neq i$. This is based on solving the following optimization problem $\varphi = \sum_{x \in X} \min_{c \in C} \|x - c\|^2$.
2. For each $i \in \{1, \ldots, K\}$, set c_i to be the center of mass of all points in C_i: $c_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$.
Until:
Condition 1: Clusters no longer change compared to previous iteration: $c^{(t)} - c^{(t-1)} < \epsilon$
Condition 2: The maximum number of iterations has been reached.
Output:
Cluster centers $c_1, .., c_K$, cluster label for each point in dataset.

For this study, the Python scikit-learn machine learning library is used, due to its reliable implementation of the K-Means algorithm class, along with the option to use `k-means++` initialization. The inputs to this library's K-Means class include the feature dataset, the number of clusters, and the initialization method. Additionally, two key parameters are specified: the maximum number of iterations per single run and the tolerance, which is the ϵ threshold mentioned above. The maximum number of iterations is set at 400 and the tolerance is set to 1×10^{-5}. These values are chosen empirically to ensure both convergence and computational efficiency.

Typically, when unsupervised learning algorithms are trained, the input feature dataset is divided into separate training and testing sets. This division is crucial to prevent data leakage from occurring, which happens when information from the training set unintentionally leaks into the testing set, potentially leading to an overestimation of the model's performance. However, when dealing with flow simulation cases, the traditional approach of splitting the input dataset becomes challenging. When clustering is applied to flow simulations, one of the primary evaluation metrics is based on visually inspecting the clusters generated by the algorithm. This visual inspection is essential to ensure that the clusters align well with the expected clustering output, as depicted in Figure 4.1. If the simulation dataset were to be split into separate training and testing sets, the ability to visually inspect the clusters would be removed. Therefore, in this study, the entire dataset from one flow simulation case is used as the

training data for the algorithm. Then, the entirety of the data from another case as the testing dataset. Using this approach ensures that the algorithm can be properly evaluated without compromising the visual inspection process and without the risk of data leakage.

4.1.2. Algorithm Performance

The K-Means clustering algorithm offers two main advantages: efficiency and simplicity. Its time complexity is linear, represented as $O(N \times K \times i \times d)$, where N denotes the number of data points, K represents the number of clusters, i signifies the number of iterations, and d indicates the number of dimensions for the dataset. This linear time complexity enables it to handle larger datasets with ease.

Nevertheless, the algorithm also has several disadvantages such as its reliance on a user-defined number of centroids. Opting for too few centroids can result in overly broad clusters, whereas selecting too many can lead to overly specific clusters. To navigate this issue, various metrics, aside from visual inspection of the obtained clusters, are available to assess clustering performance:

- **Elbow Method:** This approach involves plotting the inertia against the number of clusters and identifying the ”elbow” point. This is where the rate of change in inertia sharply levels off, indicating the optimal number of centroids. Determining the exact location of the elbow point can often be subjective and unclear, therefore a more accurate metric such as the silhouette score is advised.
- **Silhouette score:** This is based on computing a Silhouette coefficient using two distances for each data point: the mean intra-cluster distance a and the mean nearest-cluster distance b. For a particular clustering output, the Silhouette coefficient is obtained from the formula $\frac{b-a}{\max(a,b)}$, which takes values in the range of [-1, 1]. A score of 1 denotes ideal clustering, while -1 indicates the poorest clustering scenario. Values approaching 0 imply overlapping clusters and negative values suggest that samples are likely assigned incorrectly to clusters, indicating greater similarity with other clusters [46].

Another drawback of K-Means is its reliance on the measure of inertia, which assumes clusters are spherical. Consequently, when clusters are elongated or irregularly shaped, inertia may not accurately capture this variation in shape. This limitation makes K-Means highly sensitive to outliers and datasets with non-spherical or irregular shapes. To address outliers, the method outlined in Section 4.1.4 is used. Additionally, Gaussian Mixture Models (GMM) will be investigated in this study as detailed in Section 4.2, to explore whether they can surpass the performance of the K-Means algorithm. GMM models assume Gaussian distributions, offering greater flexibility in capturing complex cluster shapes. Furthermore, inertia is not a normalized metric, so as the dimensionality of the input dataset increases, leading to increased sparsity of the data in Euclidean space, Euclidean distances become less effective in measuring the distances between data points [44]. To counteract this issue, several dimensionality reduction techniques can be used as detailed below.

4.1.3. Dimensionality Reduction Techniques

As previously discussed, the performance of the K-Means clustering algorithm can be negatively impacted if high-dimensional datasets are used. In the field of machine learning, this issue is often referred to as the **Curse of Dimensionality**. Four different methods have been used in this study to address this issue:

- **Filter Selection Techniques:** This type of feature selection relies on statistical criteria. The specific filter selection technique used in this study is the Pearson Correlation method, which measures the linear correlation between two features. It is computed from the ratio between the covariance of two features and the product of their standard deviations. It is essentially a normalized measurement of the covariance [47]. The value of this measure is known as the Pearson correlation coefficient (PCC) and it ranges between -1 for a perfect negative correlation and 1 for a perfect positive correlation. When two features have a high correlation, it suggests that one feature can be computed from the other, therefore keeping both features in the dataset becomes redundant. Following the guidelines associated with this coefficient, a threshold magnitude of 0.65 has been chosen for this filter technique [48]. Therefore, if two features show a PCC higher than 0.65, only one is kept in the dataset for K-Means.
- ***Wrapper Selection Technique:*** This method is based on selecting subsets of features, passing

them through the K-Means algorithm, and evaluating the clustering performance using a predefined metric. The wrapper technique selected for this study is the Forward Sequential Feature Selection (Forward-SFS). Implemented in the scikit-learn library, this method iteratively identifies the best new feature to include in the input dataset. The algorithm starts with zero features and selects the first feature that minimizes the inertia when the K-Means algorithm is trained on this single feature. Subsequently, additional features are added, and the process repeats until adding new features no longer increases the inertia [49].

- **Inspection Selection Techniques:** A subset of the selected features in this study fails to meet certain criteria: they either lack invariance or are dependent on the Reynolds number. This is further addressed in Section 4.4. Retaining these features in the dataset would compromise the generalizability of the K-Means model to other types of flow cases. Moreover, some of these features exhibit significant noise in specific areas of the domain due to their particular formulation. If the outlier removal process fails to detect this noise, it may bias the K-Means algorithm to recognize it as a distinct cluster. Therefore, the objective of using this dimensionality reduction technique is to eliminate all features that are either noisy or do not satisfy the requirements of invariance and independence from the Reynolds number.
- **Principal Component Analysis:** This technique is a linear method for reducing dimensionality, which works by transforming the original d dimensional input feature set into n dimensional principal components. These principal components capture the directions of maximum variance present in the data. Typically, these principal components are obtained by performing singular value decomposition (SVD) on the covariance matrix of the input feature set. In PCA, terms that do not have a significant contribution to explaining variance can be disregarded. This process efficiently captures the most significant patterns in high-dimensional data, aiming to retain as much variance as possible within the reduced-dimensional space [50]. This method is available in the scikit-learn library and requires as input the number of principal components to keep in the reduced output representation.

4.1.4. Outlier Removal

The K-Means algorithm is sensitive to outliers, which can significantly impact its clustering performance. One way to detect these outliers is using box plots, as they are relatively robust to skewed distributions, a common characteristic of flow feature datasets. To identify outliers using this method, any data point lying outside the upper and lower fences of the inter-quartile (IQR) range of the box plot is labeled as an outlier:

$$\textbf{Upper Fence:} \quad Q_3 + 1.5 \times \text{IQR} \tag{4.2}$$

$$\textbf{Lower Fence:} \quad Q_1 - 1.5 \times \text{IQR} \tag{4.3}$$

where Q_3 is the upper quartile, Q_1 is the lower quartile and the IQR is defined as $Q_3 - Q_1$. Once the outliers are identified, they need to be removed. However, as discontinuities in the mesh are not allowed, one cannot simply eliminate the outliers from the dataset. Instead, outliers are clipped by substituting them with the value of either the upper or lower fence, depending on which threshold defines the outlier. Although this method may seem crude, it can maintain the overall distribution of features in geometric space.

4.2. Gaussian Mixture Models

4.2.1. Algorithm Overview

The Gaussian mixture model (GMM) is a probabilistic clustering method where observed data points are assumed to arise from a mixture of K components, each modeled as a Gaussian distribution. In this model, each data point is not assigned exclusively to a single Gaussian distribution but rather to all of them with varying probabilities, which are referred to as weights [51]. This is known as a soft clustering assignment and is much more flexible compared to the hard clustering assignment used in K-Means, which assigns each data point to only one cluster. Each Gaussian distribution has its own set of parameters θ (which include the mean μ and standard deviation σ), which can be mixed together by

adjusting the weights π. Therefore, a GMM with K univariate Gaussian distributions $N(x|\mu_k, \sigma_k)$, having the set of parameters $\theta = \{\mu_k, \sigma_k, \pi_k\}_{k=1}^{K}$, can be defined as:

$$N(\mu_k, \sigma_k) = \sum_{k=1}^{K} \pi_k N(x|\mu_k, \sigma_k) \quad \text{where } 0 \leq \pi_k \leq 1, \sum_{k=1}^{K} \pi_k = 1 \tag{4.4}$$

To estimate the probability that a certain data point, x_n, has been generated by the k-th component, which is formally known as the responsibility, r_{nk}, the following posterior distribution can be computed:

$$r_{nk} = \frac{\pi_k N(x|\mu_k, \sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_n|\mu_j, \sigma_j)} \tag{4.5}$$

The total responsibility of the k-th mixture component for the entire dataset is defined as $N_k = \sum_{n=1}^{N} r_{nk}$. These responsibilities are in fact the soft cluster labels. To estimate the parameters of each Gaussian distribution, the following procedure is used:

(i) The likelihood function is defined, representing the probability of observing a set of data points given the parameters of the model:

$$p(X|\theta) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{(x_n - \mu_k)^2}{2\sigma_k^2}\right) \tag{4.6}$$

(ii) The log of the likelihood function is computed to simplify the mathematics:

$$L = \log p(X|\theta) = \sum_{n=1}^{N} \log\left[\sum_{k=1}^{K} \pi_k \frac{1}{\sqrt{2\pi}\sigma_k^2} \exp\left(-\frac{(x_n - \mu_k)^2}{2\sigma_k^2}\right)\right] \tag{4.7}$$

(iii) The partial derivative of the log-likelihood function is computed with respect to μ_k:

$$\frac{\partial L}{\partial \mu_k} = \sum_{n=1}^{N} r_{nk} \frac{(x_n - \mu_k)}{\sigma_k^2} \tag{4.8}$$

(iv) The above derivative is set to zero to find the value of μ_k and therefore also σ_k and π_k:

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk} x_n, \quad \sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} r_{nk}(x_n - \mu_k)^2, \quad \pi_k = \frac{N_k}{N} \tag{4.9}$$

As can be seen from the above equations, to estimate the parameters, the responsibilities must be known. However, in an unsupervised clustering approach, these responsibilities are also unknown. Therefore, to estimate both the parameters and the responsibilities, an algorithm known as Expectation-Maximization is used. The algorithm consists of two steps: the E-step, in which a function for the expectation of the log-likelihood is computed based on the current parameters, and an M-step, where the parameters found in the first step are maximized. Every iteration increases the log-likelihood function (or decreases the negative log-likelihood). The algorithm stops when the log-likelihood has reached a certain threshold ϵ, or the maximum number of iterations has been reached. The threshold has been set to $\mathbf{1 \times 10^{-5}}$ for this study, and the maximum number of iterations fixed at $\mathbf{400}$, to ensure both convergence and computational efficiency. An overview of this algorithm is available in Algorithm 3, assuming univariate Gaussian distributions.

Algorithm 3 GMM Clustering Algorithm

Input:
Dataset $X = [x_1, \ldots, x_n]$ where $X \in \mathbb{R}^d$, k number of components where $K \in \mathbb{N}$
Initialization:
Estimate initial parameter values $\theta = \{\mu_k, \sigma_k, \pi_k\}_{k=1}^{K}$ for each Gaussian distribution component in the mixture model.
Repeat:
1. *E-Step* : Using current values of μ_k, π_k, σ_k, evaluate the responsibilities r_{nk}.
2. *M-Step* : Compute new set of parameters μ_k, π_k, σ_k.
Until:
Condition 1: The negative log-likelihood falls below the threshold value ϵ :
$-N \sum_{n=1}^{N} \left[\log \left(\frac{1}{\sqrt{2\pi\sigma^2}} \right) - \frac{(x_n - \mu)^2}{2\sigma^2} \right] < \epsilon$
Condition 2: The maximum number of iterations has been reached.
Output:
The final parameter values $\theta = \{\mu_k, \sigma_k, \pi_k\}_{k=1}^{K}$ and the mixture component labels r_{nk} for each point in dataset.

4.2.2. Algorithm Performance

The scikit-learn implementation of GMM offers different covariance parameter options for handling feature datasets with varying degrees of correlation among features [52]. As the features used in this study are primarily constructed from the same local flow variables used in different ratios, as further explained in Section 4.4, the features are expected to be correlated to a certain extent. The two available options which take this into account are known as **'full'** and **'tied'**. The latter assumes each component has its own covariance matrix, which can have different variances along different dimensions, as well as non-zero covariances between dimensions. This method provides the most amount of modeling flexibility but also has the highest computational cost. The **'tied'** option assumes that all components share the same covariance matrix so that features are equally correlated across components. This makes this method less computationally intensive but also less flexible. Both options will be investigated to determine which one offers the best performance.

While GMMs can handle large feature datasets, their time complexity is linear and has an equivalent formulation to the one of K-Means. Therefore having a high-dimensional dataset can make computations very expensive. As this study aims to investigate the advantages of the GMM models compared to that of K-Means, the same feature subset identified using the dimensionality reduction techniques used for the K-Means algorithm will also be used for GMM.

Similar to K-Means, this algorithm also has the disadvantage of relying on a user-defined number of K components that define how many clusters the domain will be divided into. The most common metric used to help with this selection, aside from visual inspection, is the Bayesian Information Criterion (BIC). The BIC is computed as:

$$\text{BIC} = -2\log(L) + \theta\log(N) \tag{4.10}$$

where L is the maximized value of the likelihood function of the model, θ is the number of free parameters to be estimated and N is the number of data points. The BIC score can also be used to evaluate the performance of the GMM model when using the two different options for the covariance matrices as discussed above. The scikit-learn implementation of the BIC metric will be used in this study.

4.3. Relative Term Importance Analysis Clustering

The Relative Term Importance Analysis (RITA) technique was developed during this study as an attempt to cluster the shear layer manually, rather than relying on unsupervised machine learning approaches. It was developed based on observations made while analyzing the trends in the relative importance between terms of the k equation of the k-ω SST model. For clarity, the terms of the k-equation have been labeled and displayed below.

$$\frac{\partial k}{\partial t} + \underbrace{U \cdot \nabla k}_{\text{Convection - } C_k} = \underbrace{\min\left(\tau_{ij}\frac{\partial U_i}{\partial x_j}, 10\beta^* k\omega\right)}_{\text{Production - } P_k} - \underbrace{\beta^*\rho k\omega}_{\text{Destruction - } D_k} + \underbrace{\frac{\partial}{\partial x_j}\left[(\nu + \sigma_k\frac{\omega}{\omega_t})\frac{\partial k}{\partial x_j}\right]}_{\text{Diffusion - } D_{f,k}} \tag{4.11}$$

These relative term importances can be computed for each computational mesh cell and are defined as:

$$\text{RITA}_{P_{k,imp}} = \frac{|P_k|}{(|P_k| + |D_k| + |C_k| + |D_{f,k}|)} \tag{4.12}$$

$$\text{RITA}_{D_{k,imp}} = \frac{|D_k|}{(|P_k| + |D_k| + |C_k| + |D_{f,k}|)} \tag{4.13}$$

$$\text{RITA}_{C_{k,imp}} = \frac{|C_k|}{(|P_k| + |D_k| + |C_k| + |D_{f,k}|)} \tag{4.14}$$

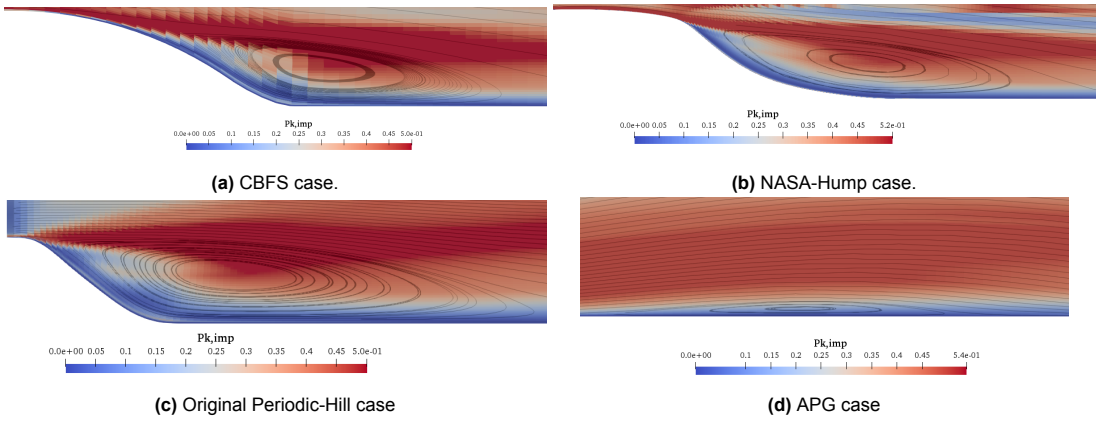
$$\text{RITA}_{D_{f,k,imp}} = \frac{|D_{f,k}|}{(|P_k| + |D_k| + |C_k| + |D_{f,k}|)} \tag{4.15}$$

The time derivative term is not accounted for as all simulations in this study are steady-state. The k equation directly addresses the transport of turbulent kinetic energy, providing insight into the distribution and behavior of turbulence within the flow field. Therefore, the relative importance between the terms of this equation should in theory indicate the local turbulent flow phenomena: the production should dominate in the shear layer, the destruction should dominate in the re-circulation region, etc. Furthermore, because the same equations are used to model the same kind of physics in all separated flow scenarios, the balances between the terms should behave the same among all the separated flow cases in this study. This makes this method generalizable and independent of geometry changes and flow Reynolds number.

The inspiration behind RITA emerged from attempting and failing to achieve promising results while applying a method described in the paper of Brunton et al., focusing on learning dominant physical processes through data-driven balance models [53]. This paper explored the dominant balance physics of a laminar boundary layer transitioning to turbulence using data obtained from a DNS simulation. The researchers used a GMM model trained on the terms of the RANS momentum equations to cluster the computational domain. RITA was developed as an attempt to apply the same ideas used in this paper, without relying on any unsupervised clustering approaches and focusing on the k equation which is deemed to be more insightful regarding the physical behavior of turbulence.

In analyzing the relative importance of terms in the k equation for the separated flow cases, clear trends emerged as depicted in Figure 4.2. This figure illustrates the variations in the $\text{RITA}_{P_{k,imp}}$ importance term across the recirculation region of each 2D-separated flow case. Notably, the production term shows its highest importance above the recirculation region, coinciding with the location of the shear layer. This can be explained by the high values of the velocity gradients in this region which increase the value of the production term in the k equation.

As evident from these results, these trends persist across all cases even when the geometry drastically changes, such as in the APG case, or when the flow conditions are different, as seen in the very high Reynolds number flow in the NASA-Hump case. Based on these trends, several attempts have been made to define value bounds to create a classifier that can cluster the shear layer from the rest of the domain.



(a) CBFS case. **(b)** NASA-Hump case.

(c) Original Periodic-Hill case **(d)** APG case

Figure 4.2: Trends in the RITA$_{P_{k,imp}}$ for all separated flow-cases.

These attempts resulted in a slight modification of the relative term importance, giving the following three RITA ratios:

$$\text{RITA}_{P_k/D_k} = \frac{|D_k|}{(|P_k| + |D_k|)} \tag{4.16}$$

$$\text{RITA}_{C_k/D_k} = \frac{|D_k|}{(|C_k| + |D_k|)} \tag{4.17}$$

$$\text{RITA}_{D_{f,k}/D_k} = \frac{|D_k|}{(|D_{f,k}| + |D_k|)} \tag{4.18}$$

These modifications were made to align the clustering analysis more closely with the physical insight of the underlying turbulence. For instance, in the shear layer, focusing on convection and diffusion is redundant, as the primary characteristic of this region is the dominance of the production of k over its destruction. The final form of this classifier has been named **RITA/TI** classifier and its formulation is the following:

$$\text{RITA}_{P_k/D_k} = \frac{|D_k|}{(|D_k| + |P_k|)} < 0.5 \quad \text{and} \quad TI = \frac{k}{k + 0.5\|U\|^2} \geq 0.12 \tag{4.19}$$

In this context, TI represents the turbulence intensity, which is a ratio between the mean flow kinetic energy and turbulent kinetic energy. Mesh cells meeting these conditions are classified as part of the shear layer. The use of the turbulence intensity, alongside the RITA ratio, was deemed necessary to ensure complete separation of the shear layer from all other areas of the flow domain, particularly the outer boundary layers where the physics closely resembles that of the outer shear layer region.

The classifier assigns a value of 0 or 1 to a variable named σ, which multiplies the b_{ij}^{Δ} and R corrections for each cell in the domain. If $\sigma = 1$, the cell is classified as being within the shear layer cluster, and full corrections are applied. Conversely, if $\sigma = 0$, the cell is classified as being outside the shear layer cluster, and no corrections are applied. Further details on this classifier can be found in Section 5.3.

4.4. Overview of Features

Features serve two purposes in this study: they make up the input dataset for the clustering algorithms, and they are part of the model discovery within the SpaRTA framework. The majority of features used in this study have been obtained from literature. Additionally, the RITA ratios discussed in Section 4.3 are also included in the feature dataset.

4.4.1. Important Feature Characteristics

Ideally, the features should reflect the physical behavior of turbulent flows and highlight important areas such as shear layers and turbulent boundary layers. Furthermore, they should be computable during a RANS simulation so that corrections can be applied locally in each cluster during the simulation. Finally, the features should be normalized, Reynolds number independent and invariant, both in terms of Galilean and rotational invariance:

- ***Normalization:*** This ensures that the features are non-dimensionalized and scaled to a specific range, which is particularly important when features have different magnitudes. Without normalization, the features with the largest magnitudes will dominate the clustering or SpaRTA regression and diminish the contributions of the other features. The most common normalization approach used in data-driven turbulence modeling literature is based on a scheme proposed by [54] which uses local quantities to scale the features. Specifically, a feature F, is normalized by a corresponding normalization factor β according to:

$$F_{norm} = \frac{F}{|F| + |\beta|} \tag{4.20}$$

  This will ensure that F_{norm} will fall within a [-1,1] range. It is also important to note that this ensures that F_{norm} will also be non-dimensional as β is always chosen to have the same units as F. For features that cannot be normalized using this approach, their normalization is achieved using Min-Max scaling:

$$F_{norm} = \frac{F - F_{min}}{F_{max} - F_{min}} \tag{4.21}$$

  where F_{min} is the minimum value of the feature and F_{max} is the maximum value of the feature.
- ***Invariance:*** The principle of Galilean invariance states that the laws of physics remain the same in all inertial reference frames that are moving at constant velocity to each other. Features that are Galilean invariant remain the same regardless of the reference frame in which they are analyzed. On the other hand, rotational invariance refers to features that remain unchanged under the rotation of the coordinate system. This ensures that the algorithm can identify and characterize flow structures regardless of the orientation of the coordinate axes. Therefore, when selecting features, it is important to consider what types of variables are invariant:
  1. **Invariant:** velocity gradients and quantities related to these gradients (strain rate, rotation rate, vorticity), Reynolds-stress tensor, and transported scalar quantities (pressure, turbulent kinetic energy, specific rate of dissipation).
  2. **Not invariant:** velocity and its partial time derivative, gradients of pressure, gradients of kinetic energy, gradients of specific rate of dissipation
- ***Reynolds Number Independence:*** Generally, a feature is Reynolds number independent if it is not constructed based on molecular viscosity. As the flow Reynolds number increases, the molecular viscosity becomes very small compared to other variables, often causing the features to tend towards infinity, rendering the feature unusable.

4.4.2. List of Features

It is common for the same feature to be formulated slightly differently among various bodies of work or to be normalized by different β parameters, as explained above. In this study, all the different formulations have been investigated before selecting the list of features below. Several of the features mentioned in this list do not meet the requirements of Galilean invariance and Reynolds number independence, as discussed above. However, it is very difficult to find or create features that meet all these requirements and demonstrate useful trends at the same time. This is one of the current major struggles in the turbulence modeling community. Therefore, these features will still be used in the analysis of this study, with attempts made to eliminate them from the SpaRTA regression and clustering analysis. Plots of all the features for the NASA-Hump case are included in Appendix Section A:

- **F1 - Distance Based Reynolds Number**: This feature is useful in distinguishing between shear and boundary layer flows and to inform the clustering algorithm of the wall distance [55]. It has the following formulation as presented in [16].

$$Re_d = \frac{\sqrt{k}d}{\nu} \tag{4.22}$$

  This feature requires further Min-Max scaling to constrain it to a [0,1] range. Furthermore, the denominator of this feature is based on the molecular viscosity which makes this feature Reynolds number dependent.

- **F2 - Turbulence Intensity**: This feature carries information regarding the length scale of turbulence. It is based on the ratio between the mean flow kinetic energy and the turbulent kinetic energy. The mean flow kinetic energy will dominate in areas such as the free stream, while the turbulent kinetic energy will dominate in areas such as shear layers where the turbulence intensity is higher. It is expressed by the following formulation [16]:

$$TI = \frac{k}{k + 0.5\|U\|^2} \tag{4.23}$$

  This feature is already bounded within a [0,1] range, so it does not require any further normalization. Since it depends on the velocity vector, it is not Galilean invariant.

- **F3 - Time Scale Ratio:** This feature, obtained from [54], is based on the ratio between the turbulent time scale and the mean-strain time scale. It provides similar information to feature F2 and can therefore be used to differentiate regions of varying turbulence intensity. The formulation is already bounded within a [0,1] range, so it does not require any further normalization:

$$TS = \frac{k\|S\|}{\epsilon + k\|S\|} \tag{4.24}$$

- **F4 - The Boundary Layer Marker:** This feature was first introduced in [56] as a protection function for Detached Eddy Simulations (DES). It was designed to be applicable to any eddy-viscosity model and to maintain robustness in irrotational regions [56]. This function assigns values of 0 within the boundary layer and 1 outside, making it a valuable feature in data-driven turbulence modeling [57], [58], as it enables differentiation between boundary and shear layers. It is expressed as:

$$f_d = 1 - \tanh\left([8r_d]^3\right) \quad \text{where} \quad r_d = \frac{\nu + \nu_t}{\kappa^2 d^2 \|\nabla U\|} \tag{4.25}$$

- **F5 - Vorticity Based Reynolds Number:** This feature provides information regarding a boundary layer undergoing separation, resulting in a separated shear layer [59]. A separated shear layer exhibits much higher vorticity values compared to a boundary layer due to the higher magnitude of the velocity gradients. It is formulated as:

$$Re_\Omega = \frac{d^2\|\Omega\|}{\nu} \tag{4.26}$$

  This feature requires Min-Max scaling to constrain it to a [0,1] range. Similar to feature F1, the denominator is based on the molecular viscosity, which makes this feature Reynolds number dependent.

- **F6 - Turbulence Based Reynolds Number:** This feature is based on the ratio of eddy viscosity to molecular viscosity and represents the ratio between turbulent and mean flow quantities [60]. It has the following formulation:

$$Re_k = \frac{\nu_t}{\nu} \tag{4.27}$$

  This feature requires Min-Max scaling to constrain it to a [0,1] range. Similar to features F1 and F5, the denominator is based on the molecular viscosity, which makes this feature Reynolds number dependent.

- **F7 - Q-criterion:** This feature is useful for identifying regions where the rotation rate is much higher than the strain rate, such as vortices and recirculation regions [55]:

$$Q_{\text{criterion}} = \frac{\|\Omega\|^2 - \|S\|^2}{\|\Omega\|^2 + \|S\|^2} \tag{4.28}$$

- **F8 - Ratio of Pressure Normal Stresses to Shear Stresses:** Pressure normal stresses result from variations in pressure in the flow, while shear stresses result from velocity gradients. Both stresses act as forces influencing the overall behavior of the flow. Therefore, the ratio between the two contains information regarding the dominant force. The following formulation is already normalized within a [0,1] range [55]:

$$PS = \frac{\|\nabla P\|}{\|\nabla P\| + \|U\nabla U\|} \tag{4.29}$$

- **F9 - Ratio of Total to Normal Reynolds Stresses:** This ratio contains information regarding the anisotropy of turbulence within a flow and provides insights into the relative importance of turbulent momentum transport in different directions. The following formulation is already normalized within a [0,1] range:

$$\tau_{ij,ratio} = \frac{\|\tau_{ij}\|}{10k + \|\tau_{ij}\|} \tag{4.30}$$

  To extract valuable information from this feature, k is multiplied by a factor of 10 to bring it to a comparable magnitude with the Reynolds-stress tensor.

- **The Relative Term Importance (RITA) Features:** The following set of normalized features were constructed in this study based on the ratios of different terms of the k equation of the $k-\omega$ SST model: production (P_k), destruction (D_k), convection (C_k), and diffusion ($D_{f,k}$). These terms are invariant and independent of the Reynolds number, making them very useful for constructing features. The formulations for these RITA features are as follows:

$$\mathbf{F10} = \frac{|D_k|}{(|D_k| + |P_k|)} \tag{4.31}$$

$$\mathbf{F11} = \frac{|D_k|}{(|D_k| + |C_k|)} \tag{4.32}$$

$$\mathbf{F12} = \frac{|D_k|}{(|D_k| + |D_{f,k}|)} \tag{4.33}$$

  The following RITA feature is only used in the SpaRTA regression:

$$\mathbf{F13} = \frac{|C_k|}{(|C_k| + |D_{f,k}|)} \tag{4.34}$$

5

Clustering Results

5.1. K-Means Clustering Results

The clustering analysis that has been carried out using the K-Means algorithm for this study is quite extensive and contains dozens of results, which cannot all be included in this report. Therefore, the results in this chapter have been selected to demonstrate the general performance that can be expected from applying this algorithm to the 2D separated flow cases and to provide an overview of the kind of difficulties that can be expected when clustering using unsupervised learning methods. These results have been obtained by training the K-Means algorithm on the NASA-Hump Baseline simulation case and subsequently testing the trained algorithm on the Periodic-Hill, CBFS, and APG Baseline simulation cases. Several modeling notes have been added throughout the chapter to caution the reader about the underlying difficulties related to clustering using unsupervised learning methods and to explain the reasons behind certain modeling assumptions.

5.1.1. Establishing Baseline Performance

Before making any modifications or adjustments to the feature dataset or the K-Means algorithm, it is crucial to establish a baseline performance. This baseline is obtained using a dataset constructed from all the features listed in section 4.4. Features that are not normalized in their raw form have been normalized using Min-Max scaling. The number of centroids is fixed to 4 to align with the clustering expectations displayed in Figure 4.1. The resulting baseline clustering assignment is depicted in Figure 5.1, with a Silhouette coefficient score of 0.56.

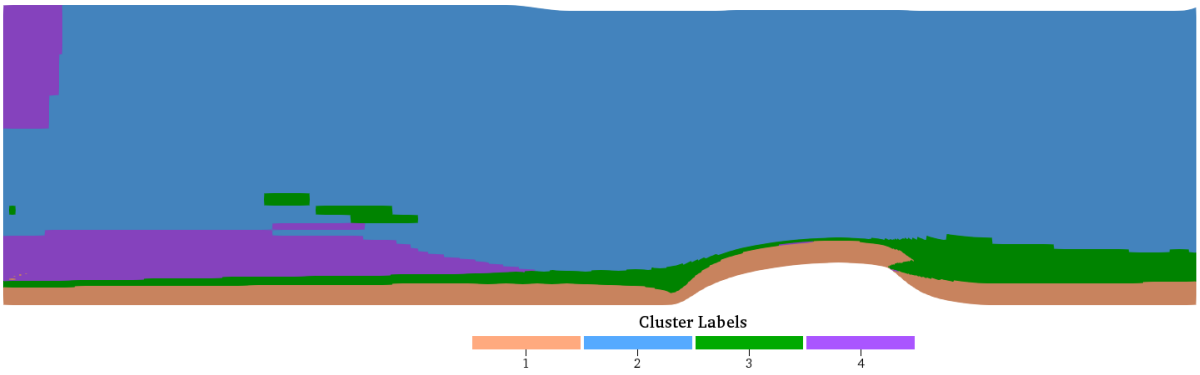


Figure 5.1: Baseline clustering assignment for the NASA-Hump obtained using the K-Means algorithm, with a Silhouette score of 0.56.

Cluster 1 groups together the boundary layer with part of the recirculation region, while cluster 2 captures

the free-stream. Cluster 3 encloses part of the transition region between the developing boundary layer and the free-stream with the shear layer. Cluster 4 is artificial and is a result of free-stream noise. Overall, the clustering output compared to the clustering expectations in Figure 4.1 is relatively poor, as also indicated by the Silhouette score, which suggests some level of overlap between clusters. It should be noted that in subsequent results, the cluster labels will change as the K-Means algorithm is always re-initialized when trained on a different set of features or with a different number of centroids. Nevertheless, the color map in each plot is kept consistent for clarity.

5.1.2. Outlier Removal

Figure 5.2 displays the box-plot distribution of the normalized feature set for the NASA-Hump case, used in establishing the above mentioned baseline performance.

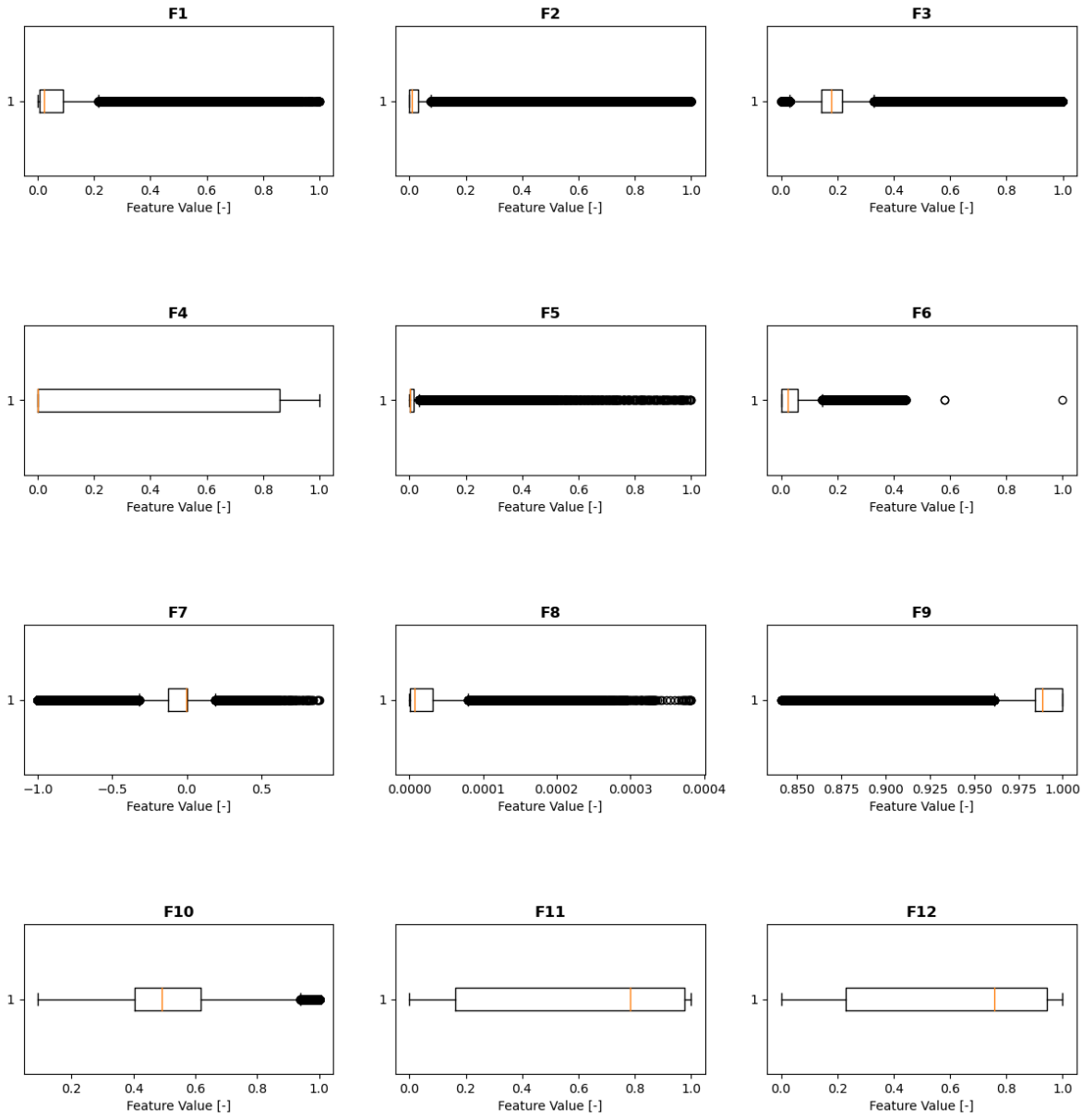


Figure 5.2: Normalized feature distribution for the NASA-Hump case.

There are two important aspects to note from this figure. Firstly, features F4 (f_d), F11 (RITA$_{C_k/D_k}$), and F12 (RITA$_{D_{f,k}/D_k}$) have very large interquartile ranges (IQRs), suggesting that they follow bi-or-multi-modal distributions. This is confirmed by the histogram plots in Figure 5.4. Since box plots rely on the

IQR to detect outliers, they may fail to identify any outliers that lie between the modes. For feature F4 (f_d), this is not a problem, as it is bimodal due to its classification of the majority of the domain into one of two regions and does not have a significant amount of noise. However, this is a problem for features F11 (RITA$_{C_k/D_k}$) and F12 (RITA$_{D_{f,k}/D_k}$), as the box plots are unable to identify the significant amount of free-stream noise present in the distribution of these features as visualized in their respective plots in the Appendix section A.1.

Modelling Note (1) on Outlier Removal

The IQR outlier removal method struggles to identify outliers in features with multi-modal distributions. Therefore, one might consider using a different outlier removal approach for such features. While this seems like a reasonable solution, it presents a challenge: the distribution of a feature can vary across different cases. For example, comparing the box plots of feature F4 (f_d) for the Periodic-Hill case (Figure 5.3) and the NASA-Hump case (Figure 5.2), the distribution of feature F4 (f_d) is no longer multi-modal in the Periodic-Hill case, and the majority of data points are classified as outliers. This variability makes it difficult to determine apriori whether a feature in a particular case will show a multi-modal distribution and what the best outlier removal method for it would be.

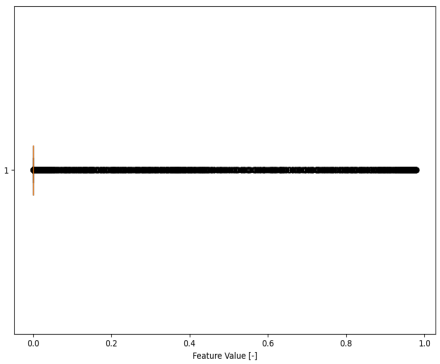


Figure 5.3: Feature F4 (f_d) distribution for the Periodic-Hill case.

Building on the above observation, if features show varying distributions across different cases and outlier removal is only applied to features that benefit from it, then this process must be conducted on a case-by-case basis. Consequently, the overall clustering method becomes less generalizable, as future cases will require in-depth analysis of the feature distributions before any clustering can be performed.

The second important aspect to notice from Figure 5.2 is that the majority of features have a large number of outliers. As outlier removal is based on the upper and lower IQR fences, capping the outlier data to the values of the fences can significantly influence the distribution of these features. Therefore, the question now becomes whether such a change is positive or negative in terms of increasing the performance of the K-Means algorithm.

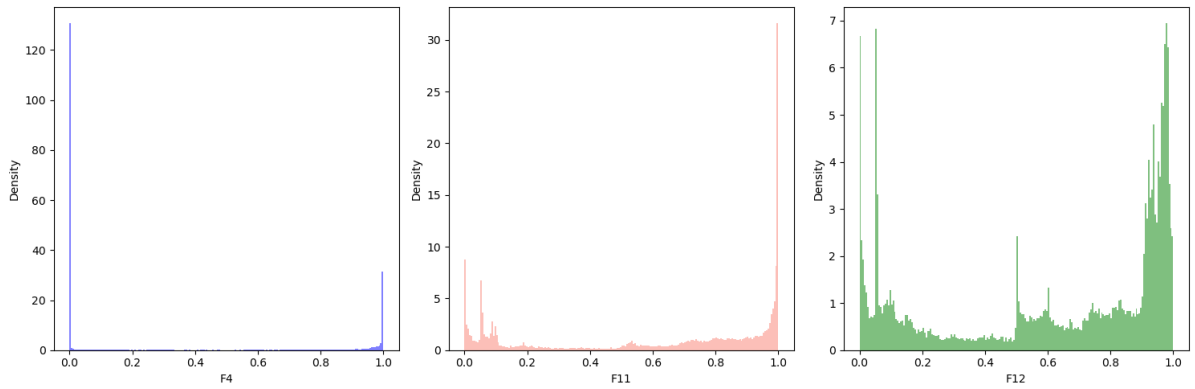


Figure 5.4: Histogram plots of features F4 (f_d), F11 (RITA$_{C_k/D_k}$), and F12 (RITA$_{D_{f,k}/D_k}$) showing that these features have bimodal or multi-modal distributions.

For example, considering the distribution of F1 (Re_d) in Figure 5.2, the outliers are most likely data points residing in the downstream hump area, as the values of this feature in that region are significantly different from the rest of the domain. Capping the values of these data points during the outlier removal process can lead to the region losing its physical significance. To verify whether this holds true, Figure 5.5 has been created, displaying how the feature appears after outlier removal for the NASA-Hump

case. As observed, the values inside the shear layer and recirculation have been capped at the upper value of 0.21. Consequently, the K-Means algorithm may face challenges in distinguishing between these two regions. A similar analysis has been conducted for the other features, except for F4 (f_d), F11 (RITA$_{C_k/D_k}$), and F12 (RITA$_{D_{f,k}/D_k}$), for which the box plot method did not identify any outliers.

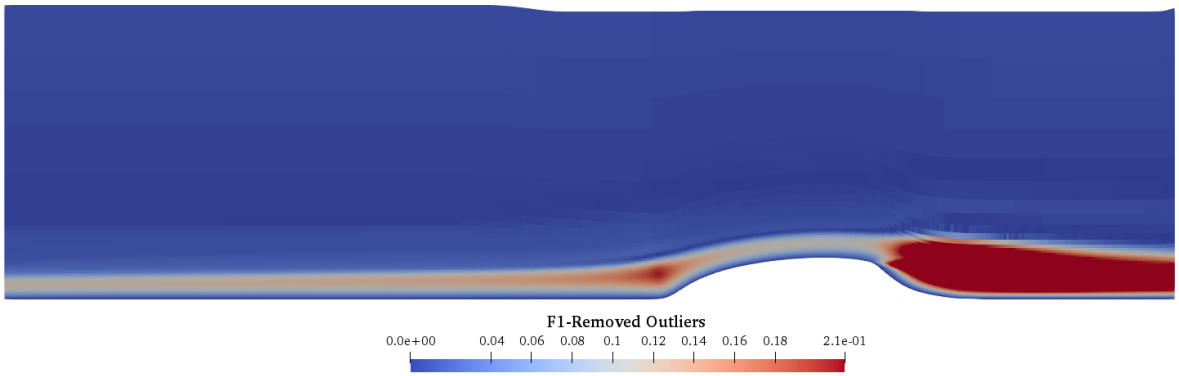


Figure 5.5: Distribution of feature F1 (Re_d) for the NASA-Hump case, post IQR outlier removal.

This analysis showed that outlier removal has a similar negative effect on feature F2 (TI) and F6 (Re_k) as it did on feature F1 (Re_d), rendering the physics in the downstream region of the hump indistinguishable. On the other hand, for feature F3 (TS), outlier removal had a positive influence by reducing noise in the free-stream and revealing patterns that make the shear layer much more discernible from the rest of the domain. For feature F5 (Re_Ω), outlier removal had a negative impact, blending part of the recirculation region with the shear layer and the outer boundary layer. For feature F7 ($Q_{criterion}$), some of the free-stream noise was reduced. For features F8 (PS) and F9 ($\tau_{ij,ratio}$), areas have been grouped that should not have been, similar to features F1 (Re_d) and F2 (TI). Outlier removal for F10 (RITA$_{P_k/D_k}$) did not seem to significantly affect the feature distribution, which aligns with the observation that this feature's box-plot distribution does not show many outliers. Overall, outlier removal has shown both positive and negative impacts on feature distributions, as well as instances where it has no impact at all. Before drawing any conclusion from these results, it is important to consider the observations stated in the modeling note below.

Modelling Note (2) on Outlier Removal

The effects of outlier removal on clustering performance are not always immediately apparent. To demonstrate this for the NASA-Hump case, two runs of the K-Means algorithm were performed. In the first run, outlier removal was applied to all features, while in the second run, outlier removal was only applied to features (F3 (TS), F7 ($Q_{criterion}$), and F10 (RITA$_{P_k/D_k}$)) that it either positively affected or had no effect on their distributions. The results of these two runs are depicted in Figures 5.6 and 5.7, characterized by Silhouette scores of 0.62 and 0.57, respectively.

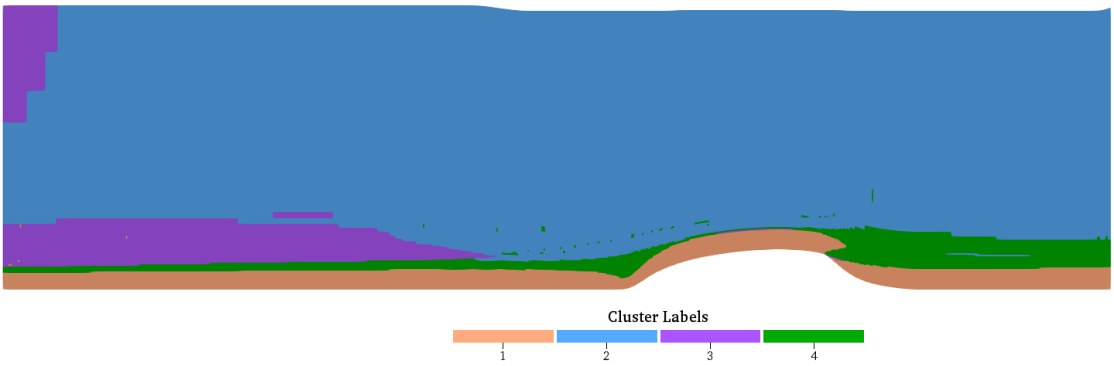


Figure 5.6: Clustering assignment obtained for the NASA-Hump case after applying outlier removal to all features. The Silhouette score is 0.62.

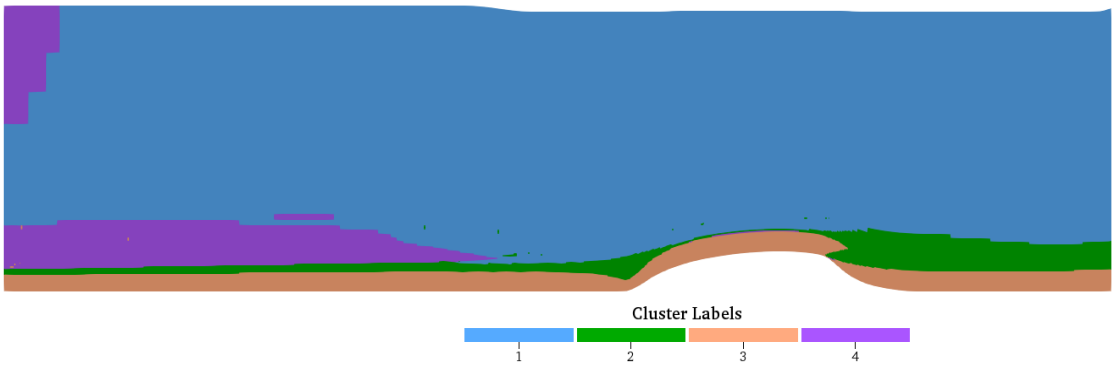


Figure 5.7: Clustering assignment obtained for the NASA-Hump case after applying outlier removal to a selected subset of features (F3 (TS), F7 ($Q_{criterion}$), and F10 (RITA$_{P_k/D_k}$)). The Silhouette score is 0.57.

Upon visual comparison of these clustering assignments, there does not appear to be a significant discrepancy between the two. Furthermore, the difference in Silhouette scores is not substantial. However, this discrepancy becomes apparent later when dimensionality reduction techniques are applied and features are removed. This is because certain features tend to dominate the clustering algorithm; when these features are removed, the importance of other features increases, leading to a shift in the clustering assignment dominated by the remaining features. In this case, it so happens that the features dominating the clustering assignment in both figures did not undergo significant changes in their distributions when the outlier removal methods were applied. Therefore, this makes the generalizability of K-Means even more challenging, as even though outlier removal does not seem to affect the clustering algorithm initially, its effects can become apparent when dimensionality reduction is applied.

Based on the above observations, it is clear that outlier removal applied to features constructed from local flow variables is not necessarily beneficial to the performance or the generalizability of the clustering algorithm. The remaining results in this chapter have therefore been obtained using a feature set from which no outliers were removed.

5.1.3. Number of Centroids

To determine the optimum number of centroids to specify for the K-Means clustering analysis, Figure 5.9 has been created, in which the inertia and Silhouette score are plotted against the number of centroids used for training the K-Means algorithm.

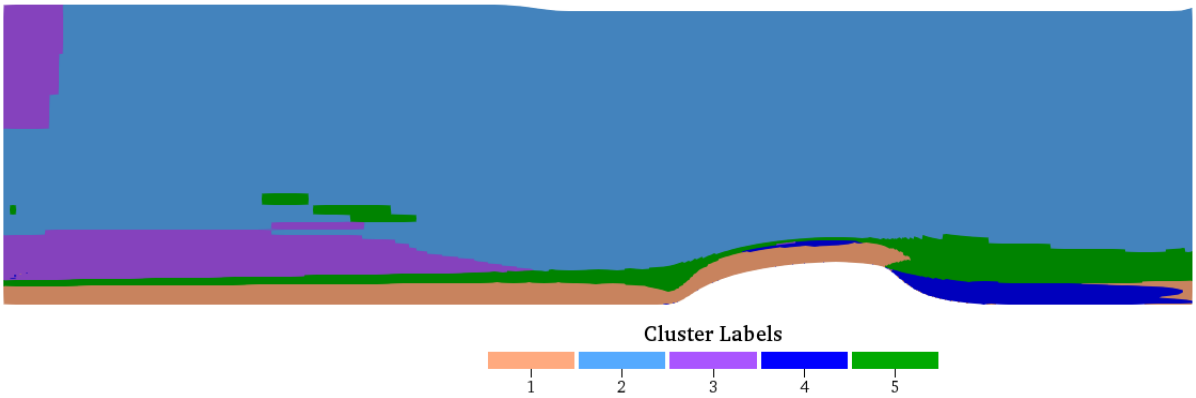


Figure 5.8: Clustering assignment obtained using 5 initial centroids for the NASA-Hump case.

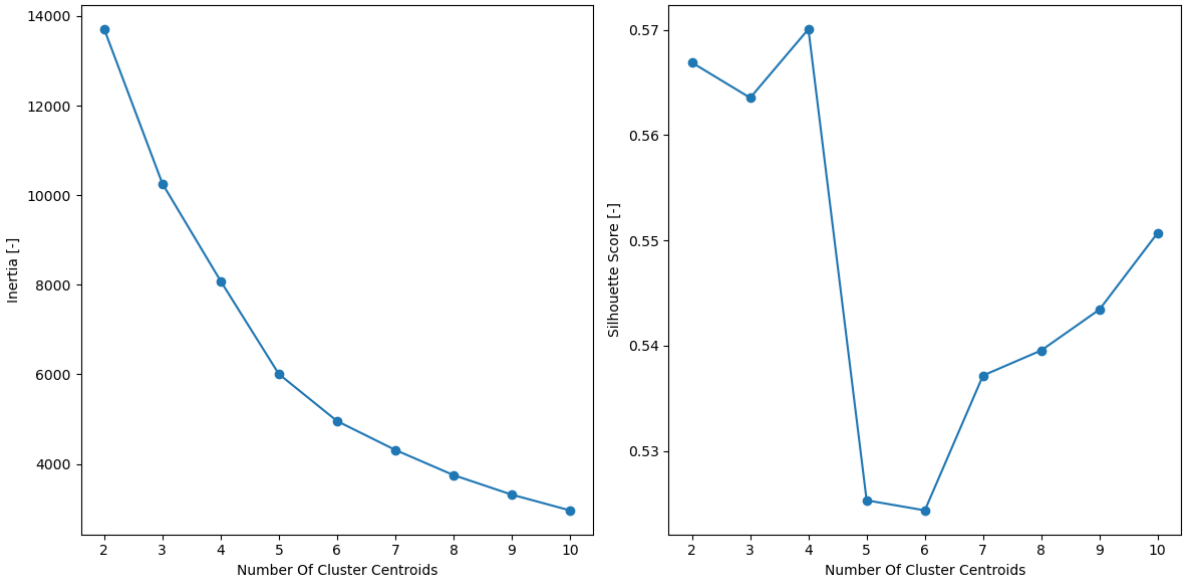


Figure 5.9: Inertia and Silhouette score plotted against the number of centroids used for the training of the K-Means algorithm on the NASA-Hump case.

The 'elbow' point in the inertia plot is found at 5 centroids, while the highest Silhouette score is obtained at 4 centroids. The results of clustering with 4 centroids are already available in the baseline performance plot in Figure 5.1. On the other hand, the results of clustering with 5 centroids are displayed in Figure 5.8. The additional cluster is labeled as cluster 4. While this region is somewhat able to separate the downstream area of the hump from the boundary layer, it remains a non-physical cluster, as it is larger than the recirculation area and groups it together with a thin layer above the hump. Therefore, the number of clusters is kept at 4 for the remaining result analysis.

5.1.4. Dimensionality Reduction

5.1.4.1 Dimensionality Reduction via Filtering

The first attempt at reducing the dimensionality of the feature dataset is to apply the filtering technique based on the PCC coefficients between features. The resulting correlation matrix is displayed in Figure 5.10. As can be seen from this figure, several of the features show a correlation strength higher than the 0.65 threshold. Feature F11 (RITA$_{C_k/D_k}$) has a strong correlation with F12 (RITA$_{D_{f,k}/D_k}$) (0.89) and with feature F7 ($Q_{criterion}$) (0.66), therefore it can be removed from the feature dataset. F1 (Re_d) has a strong correlation with F6 (Re_k) (0.78) and F5 (Re_Ω) (0.88), also justifying its removal from the dataset.

Removing these two features, results in the clustering assignment displayed in Figure 5.11 with a Silhouette coefficient score of 0.57. The artificial cluster created from the free-stream noise (cluster 2) is still present, however, it now includes the near-wall area in the downstream region of the hump and a thin layer above the hump. Cluster 4 which groups together the shear layer with the transitional region between the boundary layer and the free-stream now also contains additional noise. Overall, the clustering performance appears to be worse. However, at this point, it is difficult to conclude that removing these features does indeed harm the clustering performance. This is a similar problem also explained in section 5.1.2. Removing features leads to other features dominating the clustering performance, however, once these features are also removed, the clustering performance can drastically increase. The only conclusion that can be made from these results is that dimensionality reduction via filtering, on its own, does not ensure an improvement in the clustering performance.

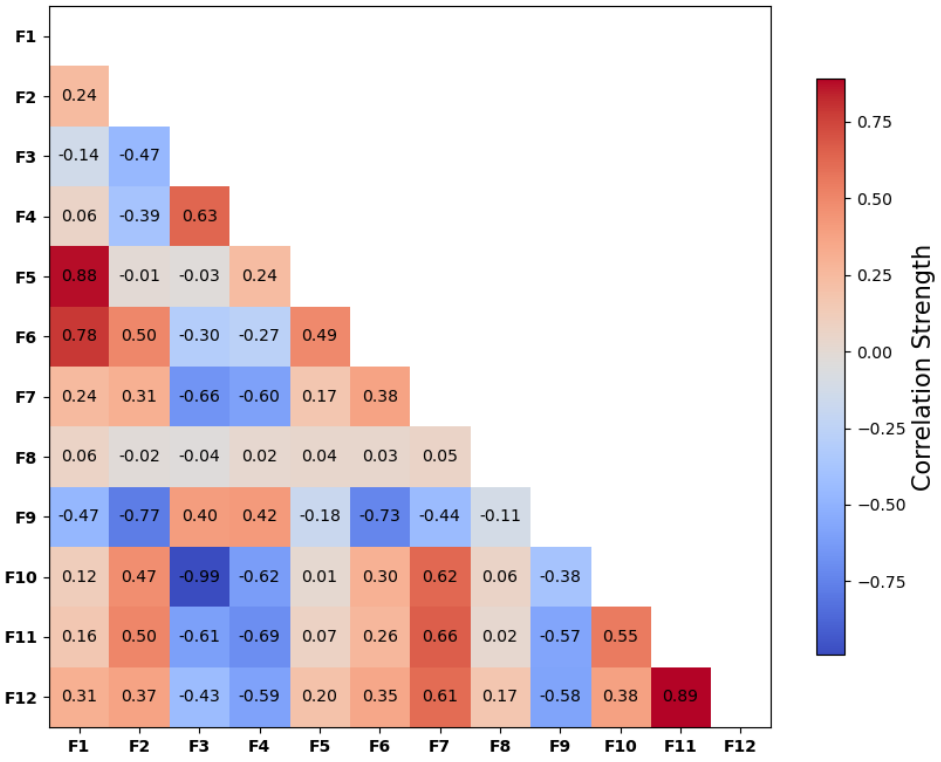


Figure 5.10: Feature correlation matrix obtained for the NASA-Hump feature dataset.

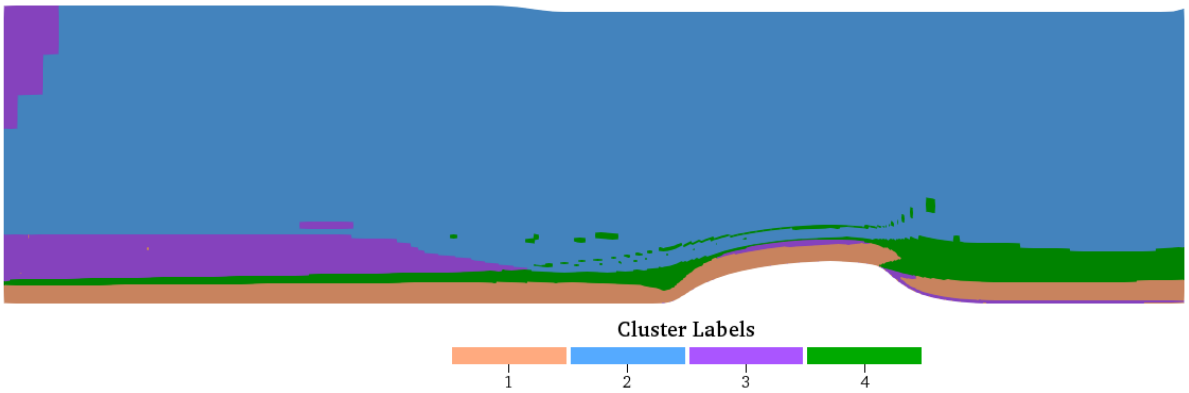


Figure 5.11: Clustering assignment obtained after removing feature F1 (Re_d) and F11 (RITA$_{C_k/D_k}$) from the dataset of the NASA-Hump case. The Silhouette coefficient is 0.57.

5.1.4.2 Dimensionality Reduction via Wrappers

The Forward-SFS method begins with a single feature and incrementally adds more features to the subset if doing so minimizes the inertia score of the K-Means clustering assignment. This process results in a boolean array where 'True' indicates that a feature should be kept, and 'False' indicates that it should be removed from the dataset. Table 5.1 provides a summary of this array.

Table 5.1: Feature selection table obtained from Forward-SFS dimensionality reduction method.

Feature	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
Included	False	True	False	False	True	True	True	True	True	False	False	False

The clustering assignment obtained using the subset of features selected by this method is displayed in Figure 5.12. The Silhouette coefficient for this clustering is 0.60. When comparing this figure with the

baseline, the removal of the five features appears to have negatively impacted the clustering assignment. While the downstream hump area is now completely separated from the rest of the domain via cluster 2, the other three clusters are all artificial. As stated in the previous section, it is difficult to draw a definitive conclusion regarding the removal of the five features, except to say that the Forward-SFS method, on its own, does not necessarily ensure an increase in clustering performance.

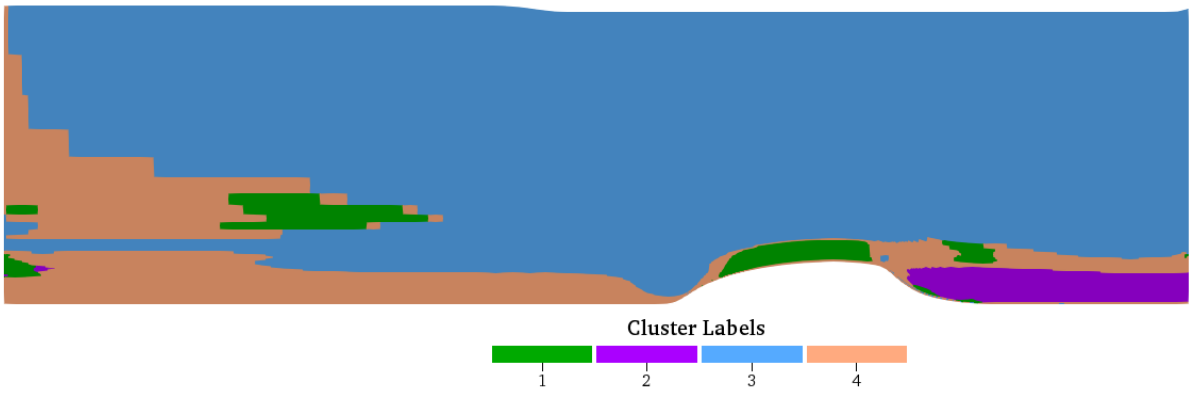


Figure 5.12: Clustering assignment obtained after removing features using the Forward-SFS method from the dataset of the NASA-Hump case. Obtained Silhouette coefficient is 0.60.

5.1.4.3 Dimensionality Reduction via Inspection

The first step in this dimensionality reduction method is to remove the features that are not invariant and Reynolds number dependent: features F1 (Re_d), F2 (TI), F5 (Re_Ω), and F6 (Re_k). As displayed in Figure 5.13, removing these features results in a clustering assignment similar to the baseline. However, the Silhouette score for this clustering is higher, equal to 0.63, and the noise in the free-stream has been reduced.

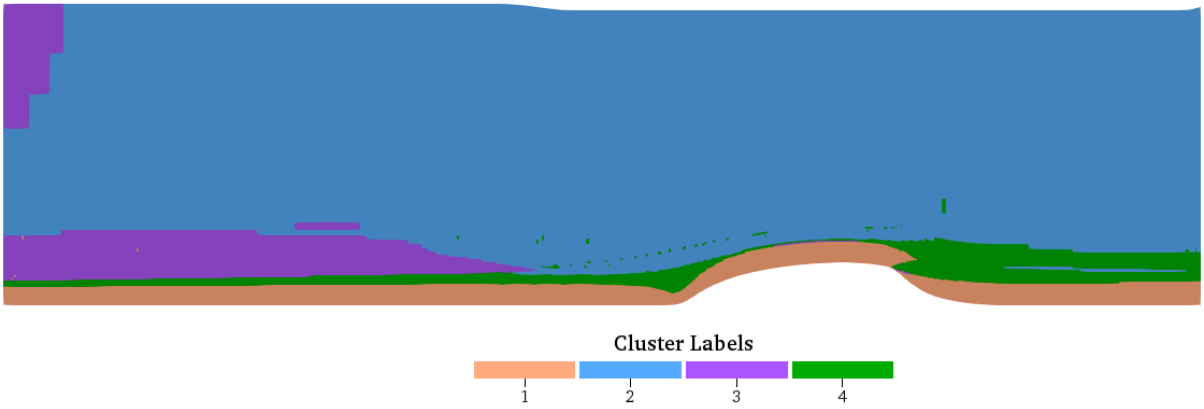


Figure 5.13: Clustering assignment obtained after removing non-invariant and Reynolds number dependent features (F1 (Re_d), F2 (TI), F5 (Re_Ω) and F6 (Re_k)) from the dataset of the NASA-Hump case. The Silhouette coefficient is 0.63.

The next dimensionality reduction step is to remove features whose distribution does not appear visually insightful, this includes very noisy features or those which do not show any recognizable patterns. An overview of all the features distribution is available in Appendix Section A.1. Features F11 (RITA$_{C_k/D_k}$) and F12 (RITA$_{D_{f,k}/D_k}$) show a significant amount of noise in the free-stream, which warrants their removal from the dataset. Similarly, feature F7 ($Q_{criterion}$) has substantial free-stream noise, and it groups part of this noise with the region above the hump, similar to feature F10. The appearance of the thin area cluster in Figure 5.13 after feature removal using filtering techniques, likely stems from the distributions of these two features dominating the clustering analysis. Consequently, F7 ($Q_{criterion}$) and F10 (RITA$_{P_k/D_k}$) are also removed from the dataset. As a result, the final feature set consists of features F3 (TS), F4 (f_d), F8 (PS), and F9 ($\tau_{ij,ratio}$).

The clustering result on this feature subset is displayed in Figure 5.14 with a Silhouette coefficient of

0.81. Comparing this figure to the baseline, there does not seem to be any improvement; if anything, the clustering in the free-stream appears even more noisy.

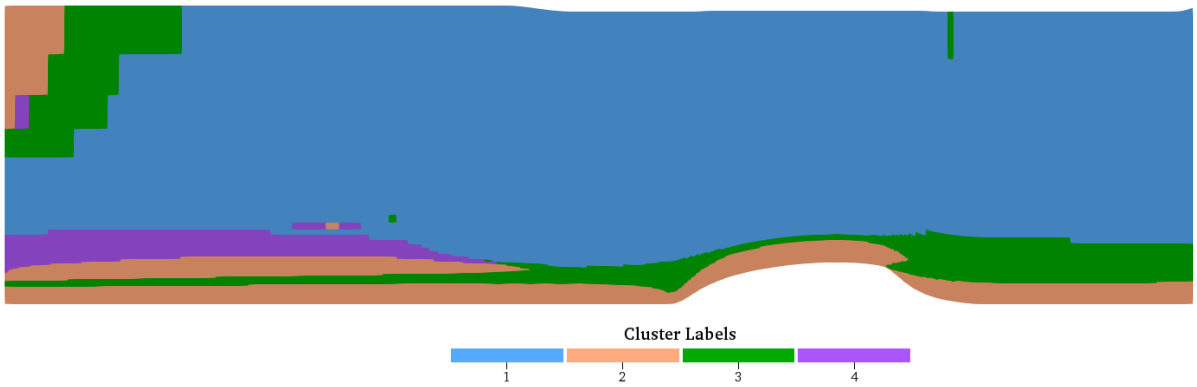


Figure 5.14: Clustering assignment obtained with the feature subset F3 (TS),F4 (f_d), F8 (PS) and F9 ($\tau_{ij,ratio}$) on the NASA-Hump case. The Silhouette coefficient is 0.81.

Modelling Note on the Silhouette Score

The Silhouette score has been consistently used throughout this chapter as a metric to assess the performance of the K-Means algorithm. However, in the final clustering result depicted in Figure 5.14, despite achieving a Silhouette coefficient of 0.81, the clusters appear even noisier and more overlapping compared to the baseline, where the Silhouette coefficient is 0.53. One possible explanation for this discrepancy is that the Silhouette coefficient is computed based on the mean distances between a sample and all other points in the same cluster, as well as between a sample and all points in the nearest cluster that the sample is not a part of. As the dimensionality of the feature set increases, the volume of space also increases exponentially. Consequently, points become increasingly sparse, and distances between points tend to become more uniform or equidistant from each other. This can lead to lower Silhouette coefficient values, even if the clusters do not appear to overlap each other as much. Therefore, relying solely on the Silhouette score may not be reliable in determining whether a particular clustering assignment is good or not, especially when reducing the dimensionality of feature sets.

5.1.4.4 Dimensionality Reduction via PCA

To determine the optimal number of dimensions for the PCA analysis, the dimensions of the input feature dataset were varied within the range of 2 to 10. For each dimension, PCA was conducted, followed by training the K-Means clustering algorithm using the PCA-transformed data. Subsequently, the Silhouette score was computed for each clustering outcome and these scores were then plotted against the respective dimensions used for PCA.

The resulting plot is displayed in Figure 5.15, indicating that the optimum number of dimensions is 3, as the Silhouette score is closest to 1. However, it is important to keep in mind the previously stated modeling note on the Silhouette coefficient, which explains why the Silhouette score may not be the ideal metric to measure clustering performance. In light of this, the clustering results were also visually inspected for each dimension in the range 2 to 10, which confirmed that 3 dimensions lead to the best clustering results.

The clustering outcome obtained by reducing the feature dataset's dimensionality to three dimensions is presented in Figure 5.16. Upon comparing this result with the baseline, it is evident that they are quite similar. However, the former shows slightly less free-stream noise, giving it a small advantage over the baseline. At this stage, this result, along with the one obtained by removing non-invariant and Reynolds number dependent features (F1 (Re_d), F2 (TI), F5 (Re_Ω) and F6 (Re_k)), show the best clustering results. Therefore, it is worth exploring the application of PCA dimensionality reduction to a feature set excluding these features, ensuring their absence from the principal components obtained with PCA.

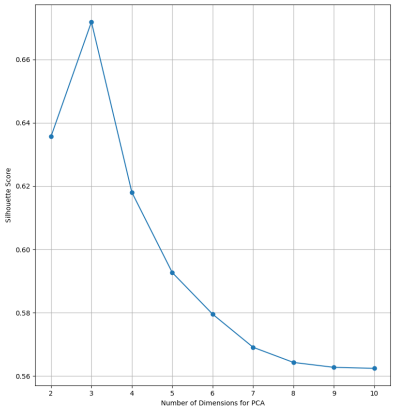


Figure 5.15: Silhouette scores of the K-Means clustering assignment plotted against the number of dimensions used in PCA for dimensionality reduction of the feature set.

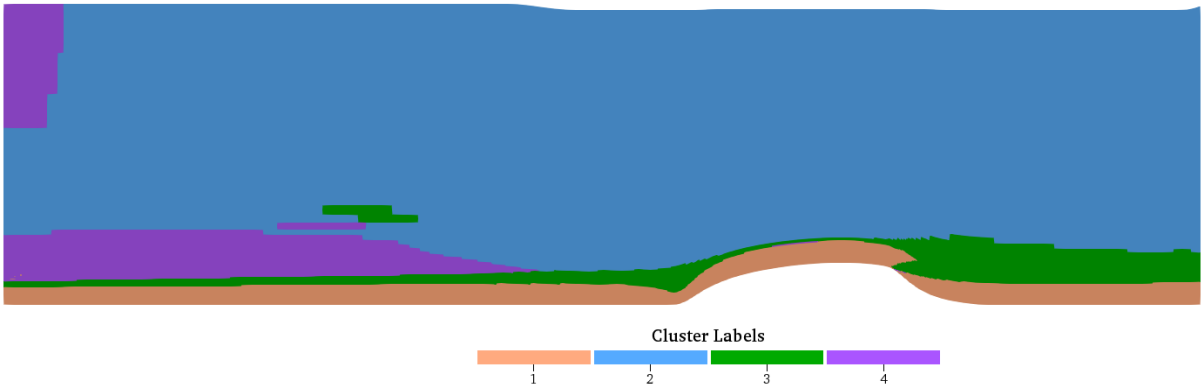


Figure 5.16: Clustering assignment obtained using K-Means on a feature set reduced by PCA using 3 dimensions.

The result of this is displayed in Figure 5.17, accompanied by a Silhouette coefficient of 0.68. From this figure, it is clear that the clustering assignment closely resembles the previous one, apart from a slight increase in the free-stream noise. Considering that eliminating these features greatly enhances the generalizability of the K-Means algorithm, and the increase in free-stream noise is only marginal, it is deemed advantageous to discard these features before applying PCA to the dataset.

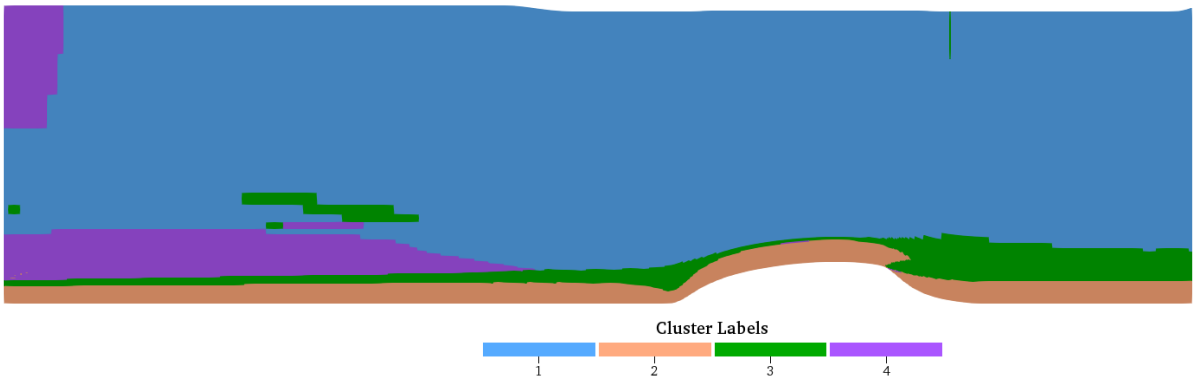


Figure 5.17: Clustering assignment obtained for the NASA-Hump case on a feature set reduced by PCA using 3 dimensions and excluding features F1 (Re_d), F2 (TI), F5 (Re_Ω), and F6 (Re_k). The Silhouette score is 0.68.

Overall, reducing the dataset's dimensionality has not yielded significant benefits across any of the four methods considered (filtering, wrappers, inspection, and PCA). This means that dimensionality reduction did not necessarily aid in aligning the clustering output with the expected patterns depicted in Figure

4.1. However, excluding non-invariant and Reynolds number dependent features, and applying PCA to the remaining features in the dataset, proved to be the best approach. The clustering performance did not deteriorate compared to the baseline, and the removal of these features enhances the K-Means algorithm's generalizability to other cases. As the PCA dimensionality reduction did not show any improvement over simply removing the non-invariant and Reynolds number dependent features, it is not beneficial to apply it to the feature set. This is because it adds a layer of time complexity without any significant added benefits. Therefore the most promising approach to training a K-Means clustering algorithm on the NASA-Hump case is to remove the non-invariant and Reynolds number dependent features from the dataset and to use four cluster centroids.

5.1.5. Generalizibility to Other Cases

The following results have been obtained by training the K-Means algorithm with four clusters on the NASA-Hump case, using a feature set from which the non-invariant (F2 (TI)) and Reynolds number dependent (F1 (Re_d), F5 (Re_Ω), F6 (Re_k)) features have been removed and from which no outliers were removed. The clustering output on this training case can be found in Figure 5.13, but has been re-displayed in this section as Figure 5.18 for clarity, where streamlines are used to help identify the location of the recirculation bubble.

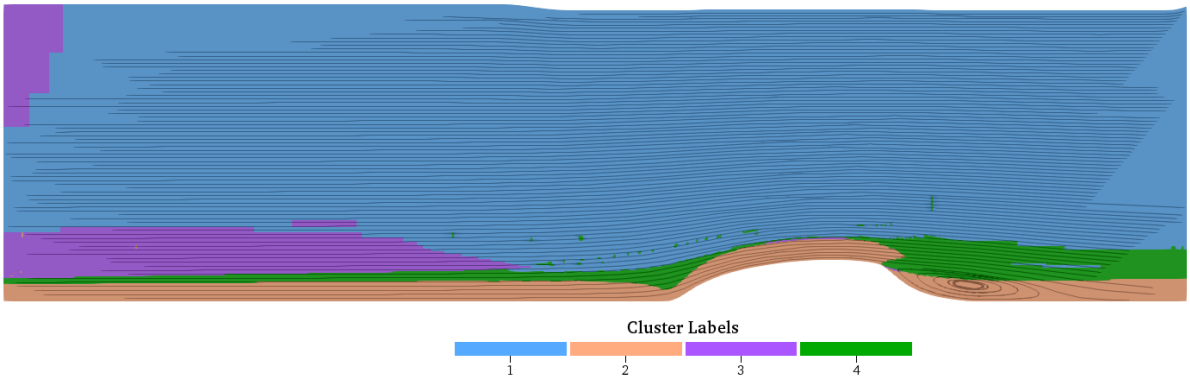


Figure 5.18: Clustering assignment obtained for the NASA-Hump case after the training the K-Means algorithm on a feature set excluding features F1 (Re_d), F2 (TI), F5 (Re_Ω) and F6 (Re_k).

The K-Means algorithm trained on this case has been used to predict the clusters for the other 2D-separated flow cases. The results of this are displayed in Figures 5.19, 5.20, and 5.21 respectively. The purple cluster, present in all three cases and the training case, is an artificial cluster created from the free-stream noise, particularly prominent in the CBFS case. The free-stream blue cluster remains consistent across all cases. The green cluster, which groups together the shear layer with part of the outer boundary layer, is present in all cases. However, for the CBFS case, the shear layer is not included in the green but in the free-stream blue cluster. Furthermore, this cluster shows considerable noise, and its location relative to the recirculation bubble varies. In the Periodic-Hill case, it sits directly atop the bubble, while in the APG case, it is at a significant distance from it.

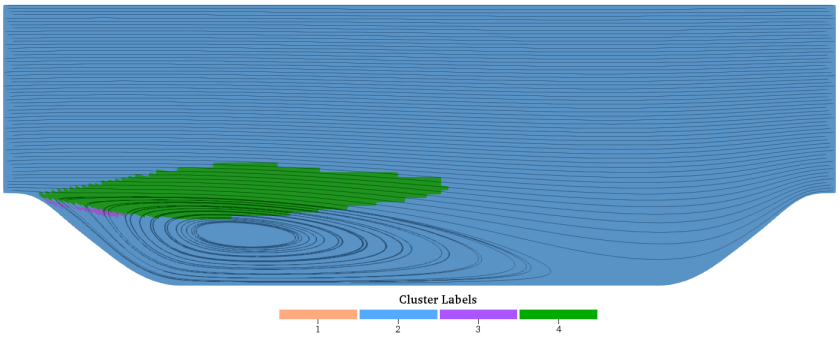


Figure 5.19: Clustering assignment obtained for the Periodic-Hill case after training the K-Means algorithm on the NASA-Hump case, using a feature set excluding features F1 (Re_d), F2 (TI), F5 (Re_Ω), and F6 (Re_k).

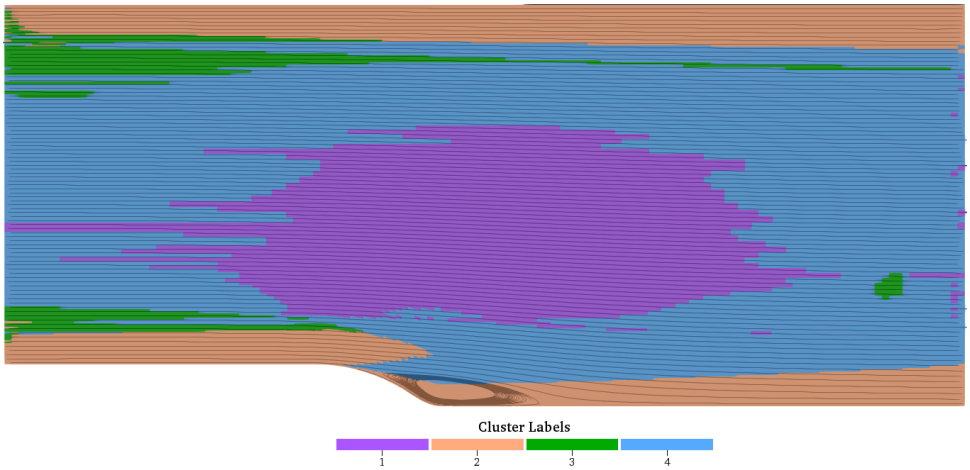


Figure 5.20: Clustering assignment obtained for the CBFS case after training the K-Means algorithm on the NASA-Hump case, using a feature set excluding features F1 (Re_d), F2 (TI), F5 (Re_Ω), and F6 (Re_k).

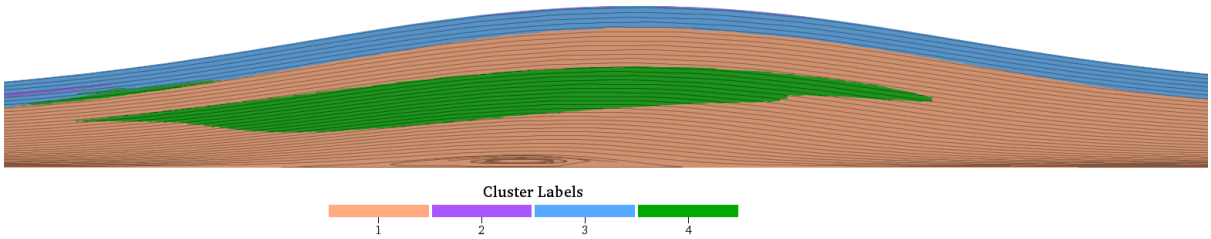


Figure 5.21: Clustering assignment obtained for the APG case after training the K-Means algorithm on the NASA-Hump case, using a feature set excluding features F1 (Re_d), F2 (TI), F5 (Re_Ω), and F6 (Re_k).

Overall, the clustering performance of K-Means is poor. In all cases except for Periodic-Hill, where no distinct boundary layer was identified, the boundary layer is consistently grouped with the recirculation region. Additionally, the shear layer is indistinguishable from the free stream or the outer region of the boundary layer. Lastly, an artificial cluster of free-stream noise persists across all cases. Consequently, the K-Means algorithm is not deemed suitable for the goal of applying local corrections to RANS simulations.

5.2. GMM Results

In this study, the GMM clustering algorithm is explored as a potential improvement over the K-Means clustering algorithm. For comparison purposes, it is trained on the same NASA-Hump feature set as K-Means, which has not undergone any outlier removal and from which the non-invariant and Reynolds number dependent features (F1 (Re_d), F2 (TI), F5 (Re_Ω), and F6 (Re_k)) have been removed. The K-Means clustering output, depicted in Figure 5.18, will serve as the reference clustering performance against which GMM will be evaluated.

5.2.1. Number of Components

To determine the optimal number of components for the GMM model, the number of components was varied from 1 to 10, and the BIC score was plotted for each clustering attempt. The covariance option was set to **'full'**. The result of this analysis is displayed in Figure 5.22, showing that the minimum BIC score is obtained for 10 components. However, considering the expected clustering output shown in Figure 4.1, which identifies four main regions, adding an additional 6 components may not be justified. Therefore, apart from evaluating the clustering performance using the BIC score, the clustering assignments were visually inspected for different numbers of components used. The optimum number of components was found to be 4, consistent with the number of centroids used for the K-Means analysis.

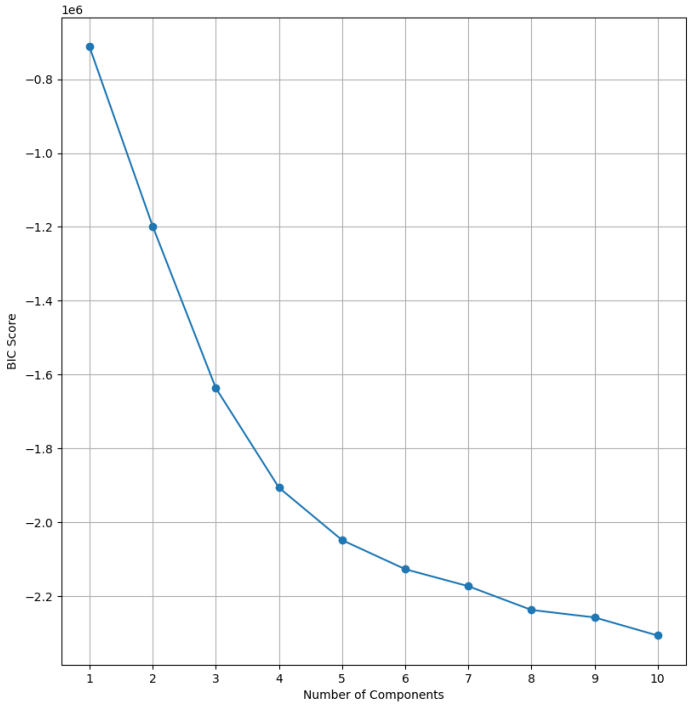


Figure 5.22: BIC score obtained for GMM clustering output against number of components used for the GMM model training.

The clustering result obtained by training the GMM model with 4 components is displayed in Figure 5.23. Comparing this clustering output with the reference K-Means clustering assignment in Figure 5.18 reveals that the GMM model performs even worse in clustering the flow domain. In this case, the free-stream noise cluster (labeled 3) appears even more elongated. Furthermore, cluster 2 groups together the outer boundary layer region and the shear layer with part of the free-stream.

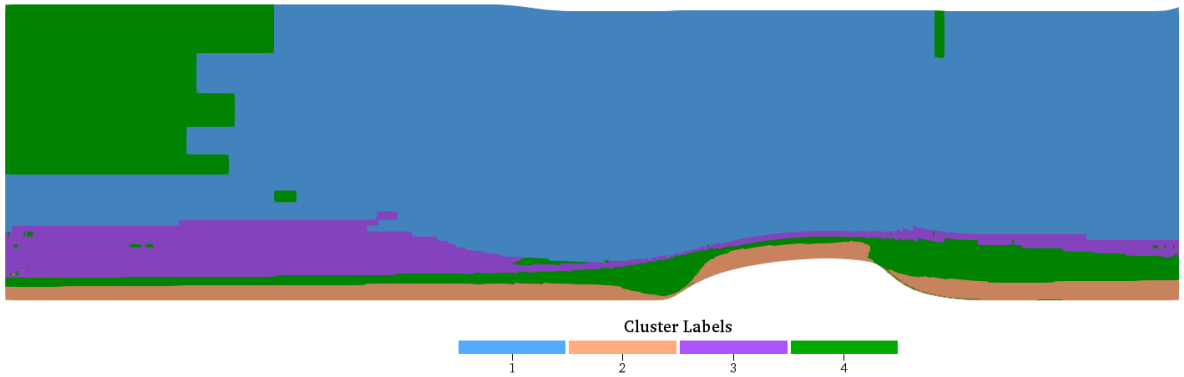


Figure 5.23: GMM clustering output obtained for the NASA-Hump case using a GMM model trained with 4 components and with the **'full'** covariance option.

5.2.2. Covariance Options

The final attempt to enhance the clustering performance involves investigating whether the **'tied'** covariance option offers any advantages over the **'full'** covariance option used in the previous results. The clustering assignment obtained using this option and keeping the number of clusters fixed to 4 is displayed in Figure 5.24. While the free-stream noise cluster (labeled 3) is now less elongated, the overall free-stream noise has increased compared to Figure 5.23. Therefore, this clustering assignment is considered worse than the one obtained using the **'full'** covariance option. This outcome is expected because the **'tied'** covariance option is less flexible, as it assumes that all components share the same covariance matrix, thereby treating features as equally correlated across components.

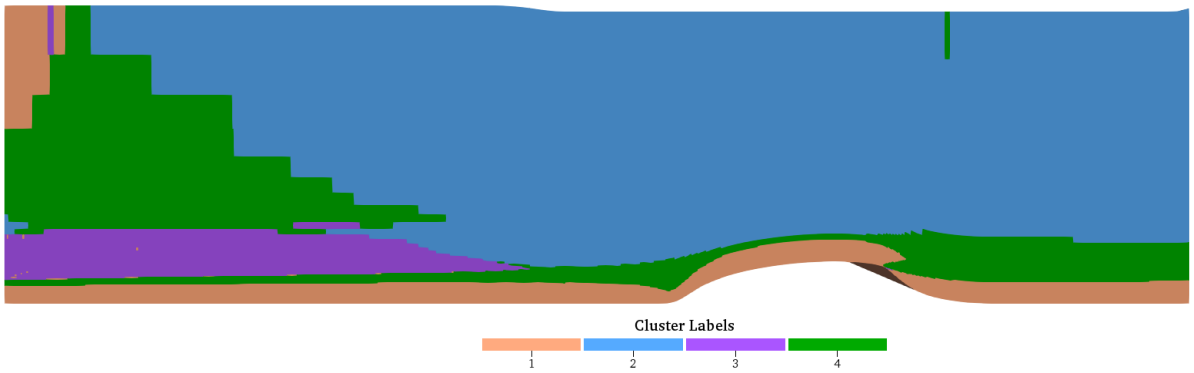


Figure 5.24: GMM clustering output obtained for the NASA-Hump case using a GMM model trained with 4 components and with the **'tied'** covariance option.

Overall, GMM shows even poorer clustering performance than K-Means. Thus, testing its generalizability to other flow cases is redundant. Neither of these algorithms has been able to achieve the main clustering objective, which is to separate the shear layer from the rest of the domain. The final attempt to cluster the domain is to create a classifier manually, rather than rely on unsupervised learning algorithms like K-Means and GMM. The results obtained using this approach are described in the section below.

5.3. RITA Classifier Design

The initial approach used in this study to design a physics-based classifier, relied on defining a threshold for the RITA ratio of production to destruction of k, to match with the expectation that the production should be higher than the destruction in the shear layer:

$$\text{RITA}_{P_k/D_k} = \frac{|D_k|}{(|D_k| + |P_k|)} < 0.5 \tag{5.1}$$

However, regions outside of the shear layer also show values below 0.5 for this ratio. In the NASA-Hump case, this occurs in the upstream region of the hump, at the intersection between the boundary layer and the hump wall, and in portions of the developing boundary layer and the free stream. Similar trends are observed for the other separated flow cases. Although excluding the free stream from the shear layer cluster is possible, by setting the lower threshold value of the ratio to 0.3, such that $0.3 < \text{RITA}_{P_k/D_k} < 0.5$, the other regions continue to be grouped with the shear layer, as illustrated in Figure 5.25 for the NASA-Hump case.

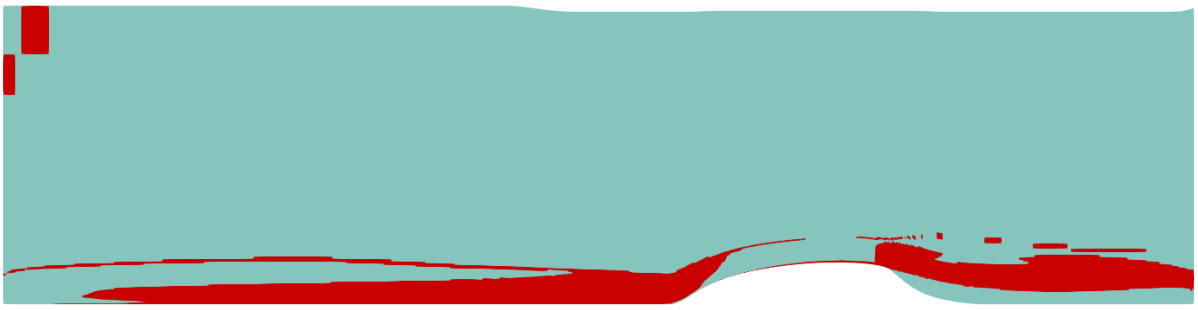


Figure 5.25: Clustering assignment based on $0.3 < \text{RITA}_{P_k/D_k} < 0.5$ classifier for the NASA-Hump case.

The next attempts at designing the shear layer classifier focused on identifying a second RITA ratio or another feature, that could be used in conjunction with RITA_{P_k/D_k} to eliminate these additional regions from the clustering assignment. However, attempts to use other RITA ratios did not lead to any improvements; they encountered the same challenges as RITA_{P_k/D_k}, resulting in the shear layer being grouped with regions outside of it. Nevertheless, one feature included in the K-Means and GMM clustering dataset showed promising results: the turbulence intensity (TI).

5.3.1. RITA and Turbulence Intensity

The distribution of the turbulence intensity (TI), whose formal definition is given in equation 4.23, is displayed in Figure 5.26. This feature shows a clear distinction between the downstream area of the hump where the shear layer is located and the rest of the domain.

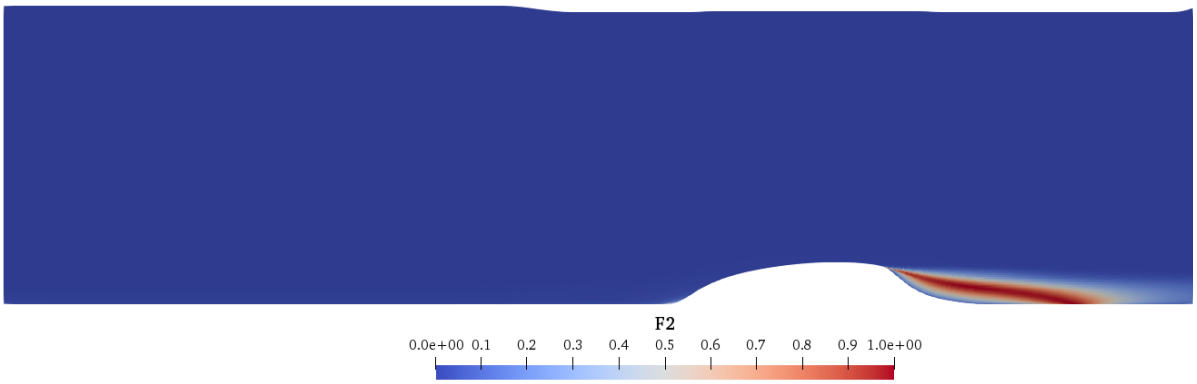


Figure 5.26: The distribution of the turbulence intensity (TI) for the NASA-Hump case.

Setting a threshold on this ratio so that:

$$TI = \frac{k}{k + 0.5\|U\|^2} \geq 0.12 \tag{5.2}$$

and keeping RITA$_{P_k/D_k}$ < 0.5, ensures that only the shear layer region of the domain meets these conditions, while all other regions are excluded. The way the classifier is constructed based on these thresholds is further explained in the modeling note below.

Modelling Note - RITA/TI Classifier

The classifier, here denoted as σ, is based on the thresholds set on the RITA$_{P_k/D_k}$ ratio and the turbulence intensity (TI). It assigns a value of 0 or 1 to every mesh cell in the domain, according to:

If: RITA$_{P_k/D_k}$ < 0.5 and $TI \geq 0.12 : \sigma = 1$ **Else**: $\sigma = 0$.

The mesh cells where $\sigma = 1$ indicate the location of the shear layer. The mesh cells where $\sigma = 0$ represent the rest of the domain.

Applying this classifier to all separated flow cases results in the clustering assignments depicted in Figures 5.27, 5.28, 5.29, and 5.30 for the NASA-Hump, CBFS, Periodic-Hill, and APG cases, respectively. Streamlines have been added to each figure to clarify the position of the shear layer cluster relative to the recirculation region. As can be seen from these figures, the cluster representing the shear layer predominantly resides above the recirculation region in all cases, extending slightly beyond the point where the flow reattaches to the wall. Visually, this clustering aligns well with the expected location of the shear layer, as depicted in Figure 4.1. Furthermore, its consistency across different cases indicates its generalizability to various flow conditions and geometries.

However, it is important to note the gap between the shear layer cluster identified in the APG case and the recirculation bubble. For the other flow cases, the shear layer cluster is directly in contact with the recirculation region. In the APG case, the area above the recirculation region has a RITA$_{P_k/D_k}$ ratio higher than 0.5, indicating that the destruction of k is higher than its production. This is an unusual result, which does not match the expectations regarding the physics of the shear layer.

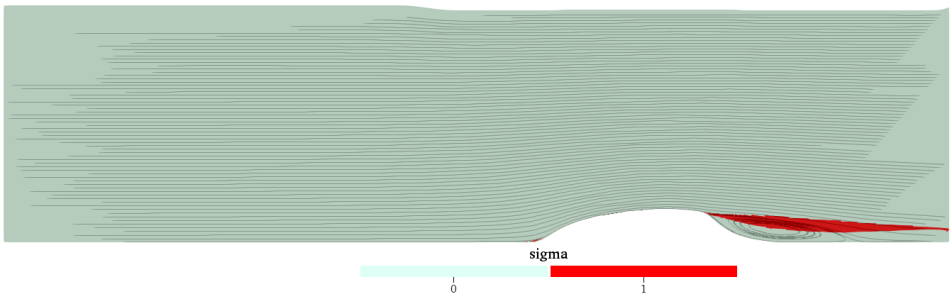


Figure 5.27: σ classifier based on RITA/TI thresholding for the NASA-Hump case.

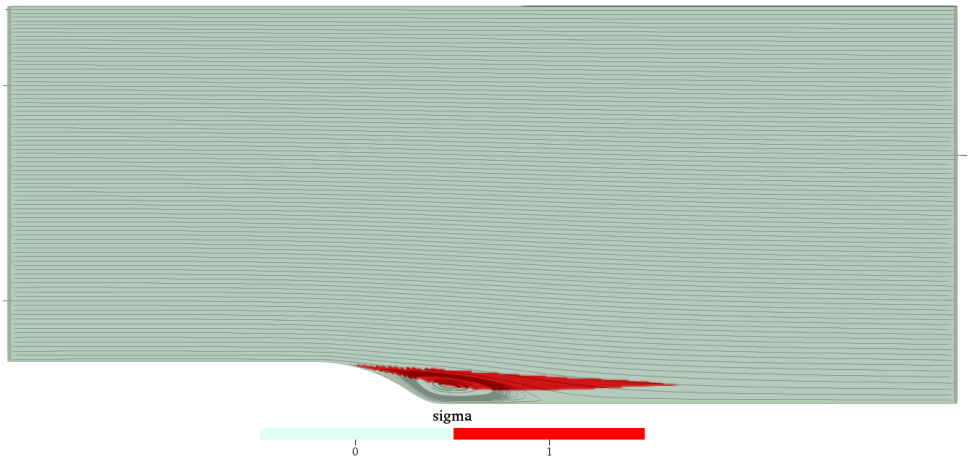


Figure 5.28: σ classifier based on RITA/TI thresholding for the CBFS case.

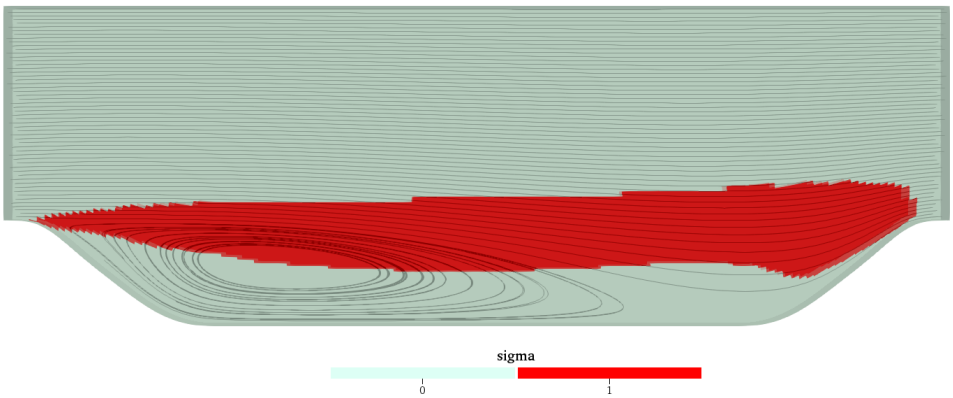


Figure 5.29: σ classifier based on RITA/TI thresholding for the Periodic-Hill case.

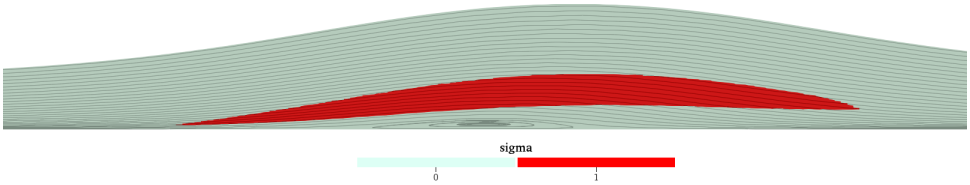
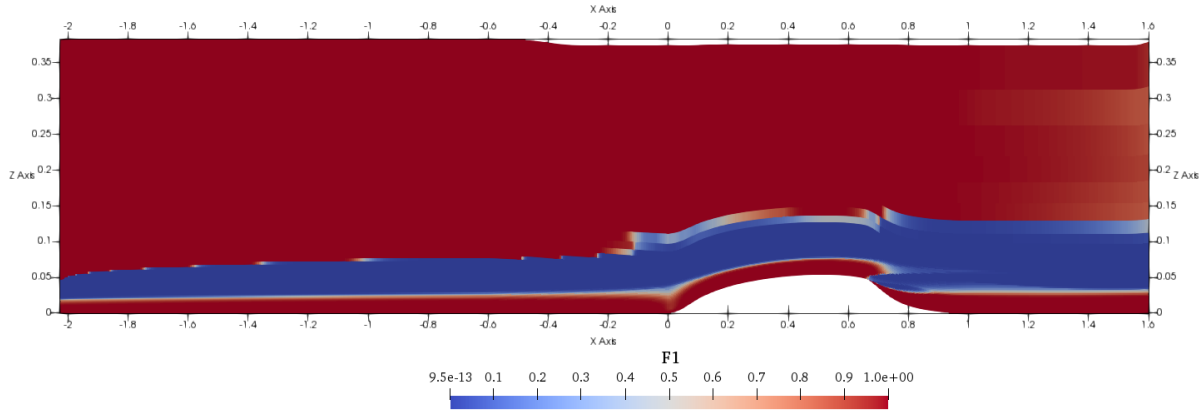


Figure 5.30: σ classifier based on RITA/TI thresholding for the APG case.

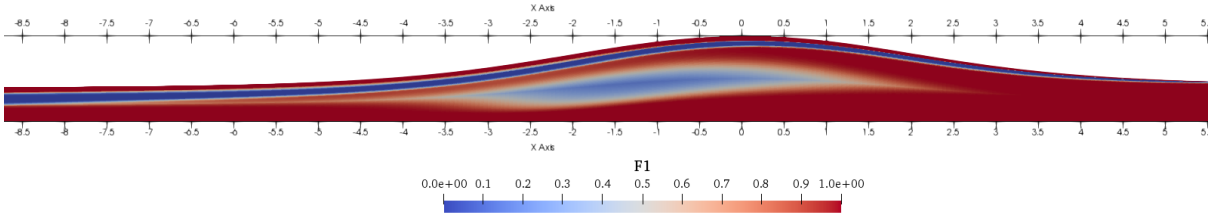
To investigate this further, the F_1 blending function which is used by the $k-\omega$ SST model to switch from its $k-\epsilon$ formulation in the free-stream and its $k-\omega$ formulation in the near-wall region, is plotted for

both the APG case and the NASA-Hump in Figure 5.31. For clarity, when F_1 equals 0, the $k-\epsilon$ model is fully active, while a value of 1 indicates full activation of the $k-\omega$ model. Values between 0 and 1 represent a blend of the two models. Additionally, the F_2 blending function, used by the $k-\omega$ SST model to limit the eddy viscosity to align with Bradshaw's observations regarding the level of turbulent energy production in the outer wake of the boundary layer, is depicted for both the APG case and the NASA-Hump in Figure 5.32

In both the NASA-Hump and APG cases, the $k-\omega$ model is fully active in the near-wall and recirculation regions. However, at the point of separation, there is a sudden switch to the $k-\epsilon$ model for the NASA-Hump, a transition not evident in the APG case. Here, near the separation region and throughout much of the shear layer's initial stretch, the $k-\omega$ model remains active. The $k-\epsilon$ model is known to sometimes overpredict the production of k in the shear layer, leading to higher shear stresses and consequently to earlier reattachment location predictions. The other 2D-separated flow cases show similar trends as the NASA-Hump. For these cases, likely, the rapid switch to the $k-\epsilon$ model in the shear layer helps increase the production of k over its destruction, resulting in a correct prediction of the shear layer cluster. For the APG case, where this switch does not occur abruptly and the $k-\epsilon$ model is never fully activated in the shear layer, it also does not result in enough turbulent production of k in the shear layer. The shear layer therefore resides higher up above the recirculation bubble, near the region where the switch to the $k-\epsilon$ model begins to occur.



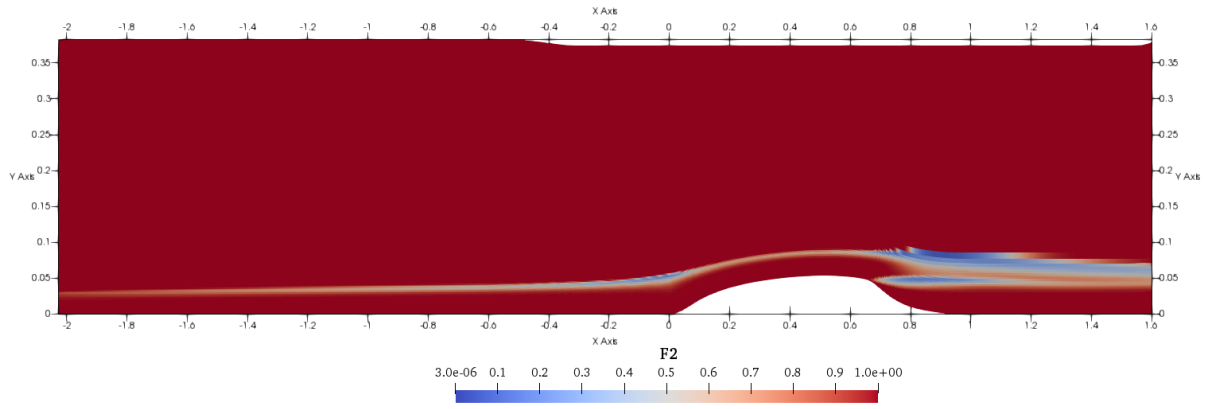
(a) F_1 blending function contour for the NASA-Hump case.



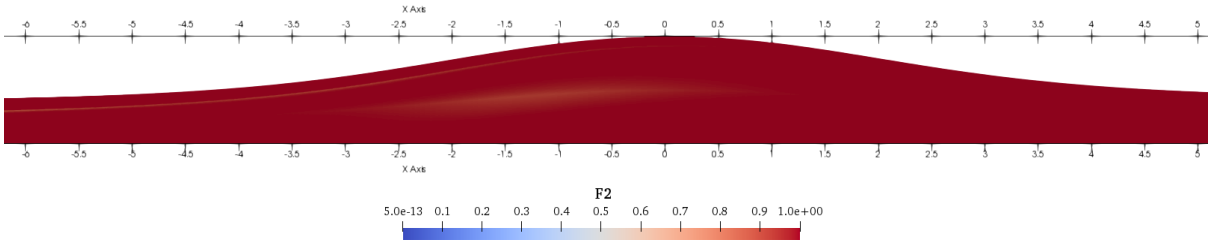
(b) F_1 blending function contour for the APG case.

Figure 5.31: Overview of the F_1 blending function used to switch between the k-ω and $k-\epsilon$ variants within the k-ω SST model framework.

On the other hand, it is also possible that discrepancies observed for the APG case are caused by the F_2 blending function. As shown in Figure 5.32, there is a noticeable difference in how the eddy viscosity is limited between the APG and NASA-Hump cases. When F_2 equals 1, the eddy viscosity is fully limited, whereas when F_2 equals 0, the eddy viscosity is not limited and is computed directly from the ratio of k to ω. For the APG case, the eddy viscosity is limited in the majority of the domain, while in the NASA-Hump case, the limiter decreases to below half of its strength in the shear layer region. Not fully limiting the eddy viscosity in the shear layer allows for higher production of turbulent kinetic energy, which helps the classifier to easily distinguish between these regions and the rest of the domain. In the APG case, the region with slightly less limitation on the eddy viscosity is located higher above the recirculation region, which could explain why the RITA/TI classifier identifies a region higher than where the shear layer is expected to reside for this case.



(a) F_2 blending function contour for the NASA-Hump case.



(b) F_2 blending function contour for the APG case.

Figure 5.32: Overview of the F_2 blending function of the $k-\omega$ SST model, used to determine in which region of the flow the eddy viscosity should be limited.

Overall, the clustering results obtained using the RITA/TI classifier represent the most promising outcomes thus far in the analysis, significantly surpassing the performance of unsupervised clustering methods like K-Means and GMM. One final remark is that the RITA/TI classifier relies on turbulence intensity, which is known to be Galilean invariant, potentially limiting its applicability in cases with moving reference frames. Nevertheless, as the majority of industrial CFD simulations do not involve moving reference frames, this classifier remains applicable in most scenarios. Nonetheless, attempts have been made in this study to identify an invariant alternative, discussed further in Appendix Chapter B.

5.4. Final Conclusions

Throughout this chapter, various methods have been investigated to obtain a generalizable classifier capable of identifying the shear layer region from the rest of the flow domain. Unsupervised clustering methods such as K-Means and GMM yielded poor results. Not only were the trained classifiers not generalizable, but they also failed to identify the shear layer. These methods are highly sensitive to the input feature dataset, and noise can significantly bias the clustering output. Feature datasets constructed from flow variables are notoriously difficult to pre-process; outlier removal often either negatively impacts the distribution of the features or fails to identify outliers. The RITA/TI classifier, designed to match the physical expectations of the turbulence characteristics of the shear layer, demonstrated the best overall performance in identifying the shear layer from the rest of the domain. Consequently, this is the classifier that is used in this study for applying local corrections to the k-ω SST model.

The results of this chapter provide answers to the first two research questions formulated for this study. It is indeed possible to construct a classifier for the shear layer so that model corrections obtained through SpaRTA, as discussed in Chapter 7, can be activated only where necessary. While this classification appears to be relatively consistent across different domain geometries and flow Reynolds numbers, it shows a slight underperformance for the APG case, where it predicts the shear layer to be slightly higher above the recirculation region than expected. This discrepancy seems to be correlated with the trends observed in the F_1 and F_2 blending functions of the $k-\omega SST$ model, which differ as the geometry of the case changes drastically.

6

Comparison of Baseline and Propagation Results

This chapter covers the differences observed in the flow behavior predicted by Baseline RANS simulations and Propagation Simulations. While the residual and convergence probe plots have not been added for each simulation, they have been checked to ensure every simulation has converged successfully. The analysis begins with a comprehensive overview of the NASA-Hump, outlining the methodologies used to compare and interpret the results. Detailed explanations of any additional simulations conducted are presented here. Subsequent sections on the Periodic-Hill, CBFS and APG cases are more concise, highlighting only the most important observations gathered from the results. The chapter concludes with key findings drawn from the presented results and important considerations for SpaRTA model training and evaluation in Chapter 7.

6.1. NASA-Hump

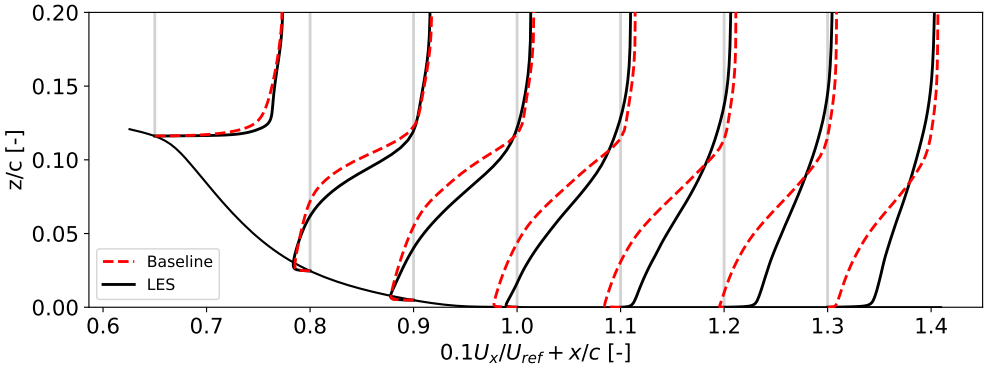
6.1.1. Discrepancies Observed in Baseline RANS

Figure 6.1 illustrates the profiles of axial velocity U_x, turbulent kinetic energy k, and the shear stress component $\overline{u'w'}$ of the Reynolds-stress tensor, downstream of the hump at various x/c locations, obtained from the Baseline simulation and LES data. Scaling factors are applied to these profiles to prevent overlap while ensuring significant deviation from zero for clear visualization. For clarity, the grey vertical lines represent the 'zero' lines, indicating zero values when a profile intersects or aligns with them. Profiles to the left of these lines denote negative values, while those on the right denote positive values.

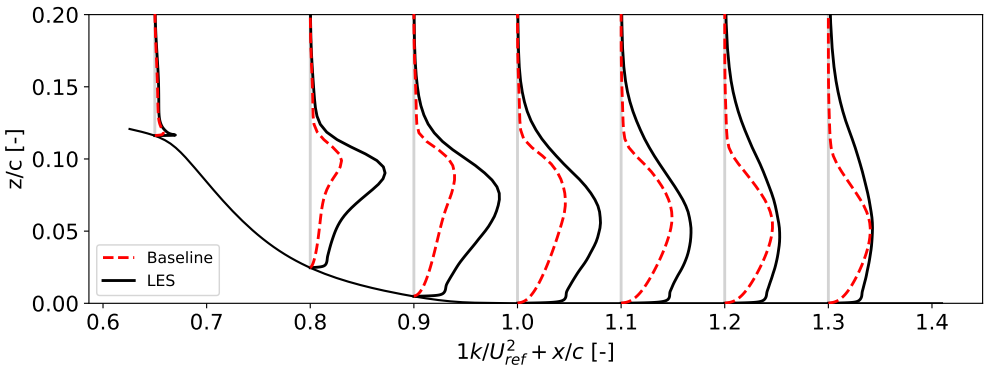
These profiles are supported by the skin friction coefficient C_f and pressure coefficient C_p plots along the full length of the bottom wall boundary of the domain, displayed in Figure 6.2. Note that the y-axis of the C_p plot has been inverted to match displays of this plot found in other literature studies [38]. As the fully developed boundary layer advances toward the hump, it begins to decelerate, as evidenced by the sudden drop in the C_f values and the increase in the C_p values. The match between the Baseline and LES predictions of the profiles in this region is relatively good. However, slightly before $x/c = 0$, which marks the onset of the hump geometry, the Baseline C_f profile drops slightly below 0, indicating a very small incipient separation region, which is not predicted by LES.

The subsequent sudden increase in the C_f coefficient, accompanied by a sudden decrease in the C_p coefficient, marks the onset of flow acceleration over the hump wall due to a favorable pressure gradient developed as the domain constricts. While the Baseline simulation displays a singular plateau atop the hump geometry between $0.1 < x/c < 0.6$, the LES data reveals two plateaus, the first occurring between $0.1 < x/c < 0.2$. This additional plateau is attributed to relaminarization, as described in [38], which is thought to arise due to a strong favorable pressure gradient in this region. However, as any small instability leads to turbulent effects taking charge, the C_f values increase and reach the second

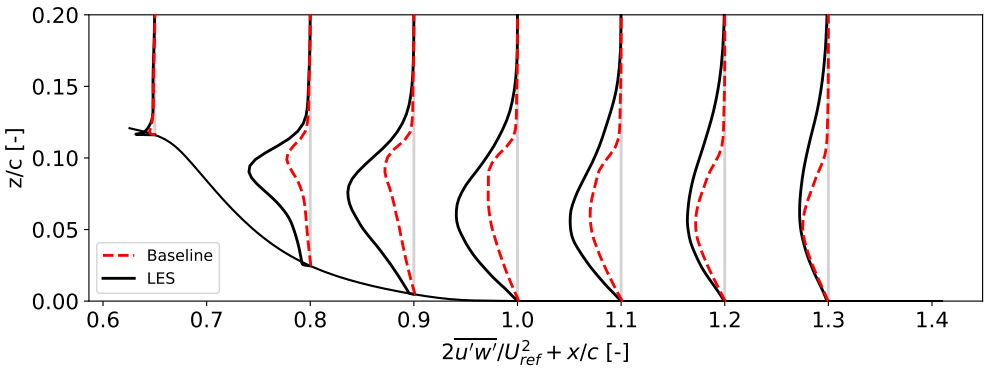
plateau. The Baseline simulation fails to capture this relaminarization as the k-ω SST model cannot predict such behavior.



(a) U_x profiles.



(b) k profiles.



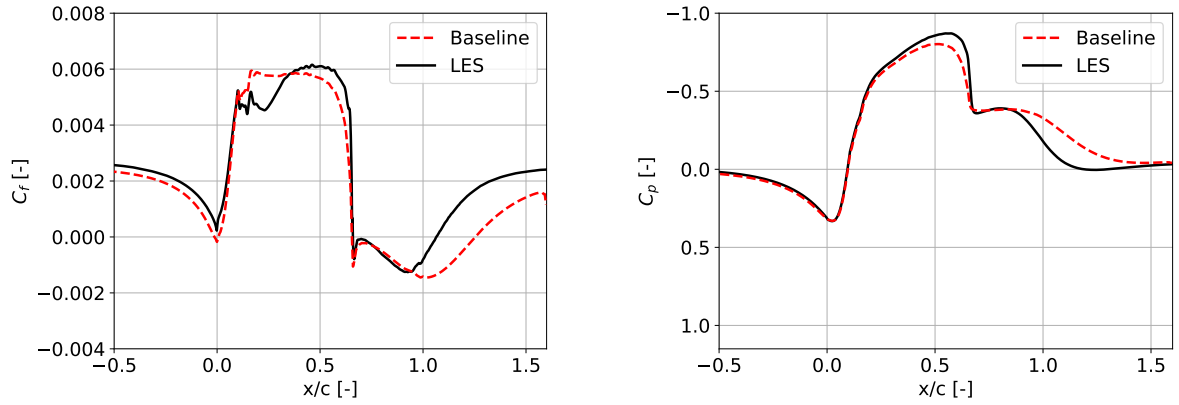
(c) $\overline{u'w'}$ profiles.

Figure 6.1: Comparison of LES and Baseline U_x, k and $\overline{u'w'}$ profiles at various x/c locations downstream of the hump.

The subsequent sudden decrease in the C_f values and increase in the C_p values marks the onset of flow deceleration as the domain expands again after the hump, leading to the development of an adverse pressure gradient, which eventually leads to flow separation. The separation point is predicted very accurately by the Baseline simulation at $x/c = 0.65$, compared to $x/c = 0.66$ by the LES data. The kinks in the C_f profile in the separated region are due to the changes in the contour shape of the hump geometry. The flow is predicted to reattach at $x/c = 1.25$ by the Baseline simulation, which is much delayed compared to the reattachment location predicted by LES at $x/c = 1.06$. This indicates that the recirculation region predicted by the Baseline simulation is much too large, which aligns with expectations regarding the performance of RANS in predicting the flow behavior in 2D-separated flow cases [61][62]. This is also clearly observed when comparing the relative positioning of the U_x profiles

predicted by Baseline and LES relative to each other and to the gray 'zero' lines.

The reason behind the delayed reattachment location in the Baseline simulation can be understood from the k and $\overline{u'w'}$ profiles. The Baseline simulation underpredicts turbulence production and shear stress in the shear layer region above the recirculation region. There are two primary reasons for this: the model-form error in the Reynolds-stress tensor (L2 uncertainty) and the model-form error in the $k-\omega$ SST model formulation (L3 uncertainty). An investigation into which of these model-form errors is more critical to the accuracy of the Baseline RANS simulation for the NASA-Hump case is carried out in Section 6.1.3.



(a) C_f vs x/c **(b)** C_p vs x/c

Figure 6.2: Comparison of LES and Baseline C_p and C_f coefficients along the bottom wall boundary of the domain.

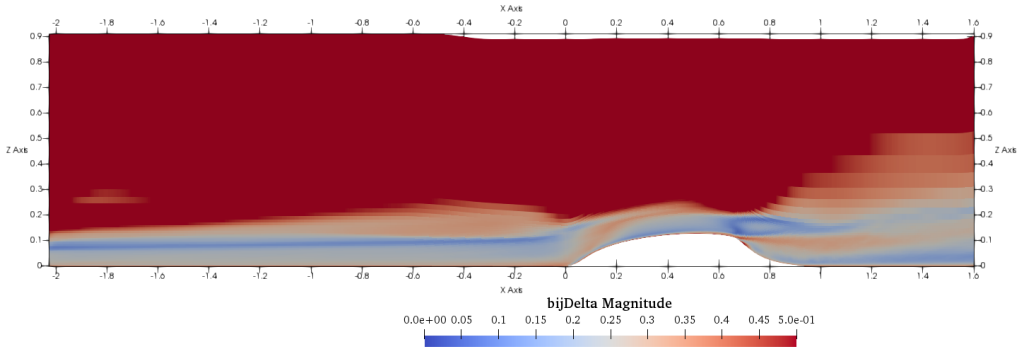
6.1.2. b_{ij}^{Δ} and R Correction Fields

The correction fields for the magnitude of the b_{ij}^{Δ} normalized turbulence anisotropy and the R residual term of the k-ω SST model equations, obtained using the Frozen simulation approach, are depicted in Figure 6.3. The free stream corrections observed in the b_{ij}^{Δ} field result from dividing by very small numbers during its computation from a_{ij}^{Δ} and k_{LES}, which both approach zero in this region. However, these corrections are not expected to influence the Propagation simulations, as the b_{ij}^{Δ} field is always multiplied by k in the Reynolds-stress tensor equation, effectively setting the correction values to zero in the free stream.

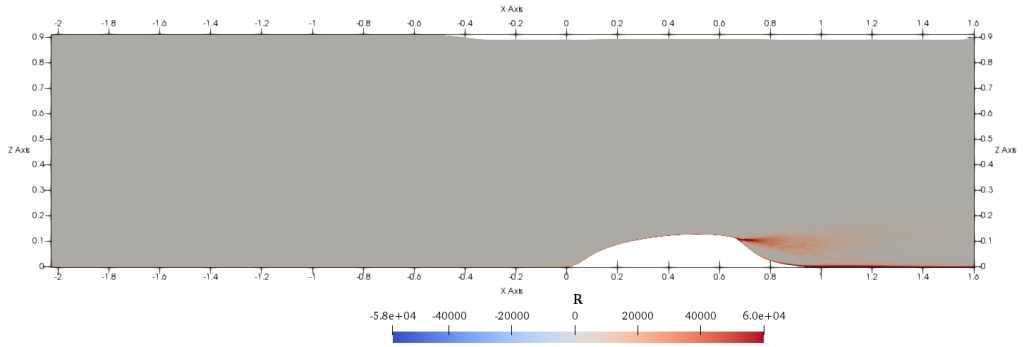
The correction fields show that the most significant discrepancies in the RANS simulations are concentrated in the shear layer region, with the highest magnitude corrections occurring precisely at the edge of the hump where boundary layer separation initiates and in the initial stretch of the shear layer. However, it is important to acknowledge that discrepancies are also evident in other regions of the domain, as can be observed in the b_{ij}^{Δ} correction field at the junction between the incoming boundary layer and the back-side of the hump. This discrepancy is likely attributed to the observed difference in the formation of a small separation region in the Baseline simulation, as seen in section 6.1.1 in the Baseline-LES comparison plots for C_f. Furthermore, the high corrections right on top of this junction region can likely be attributed to the relaminarization captured by LES but not by Baseline RANS.

R can act both as a production term and a destruction term, locally increasing and decreasing the values of turbulent energy production. It is crucial to note that the high positive values observed in the R correction field within the shear layer region indicate that the corrected anisotropy tensor alone cannot accurately predict the actual production of k. This suggests that model-form error in the $k-\omega$ SST model also plays a significant role in predicting the underlying physics correctly. The R correction field shows no significant corrections for the boundary layer upstream of the hump, which aligns with expectations, as the $k-\omega$ SST model was calibrated to accurately predict a developing turbulent boundary layer on a flat plate. The high R corrections in the downstream near-wall area of the domain are more challenging to explain. Comparing the predicted turbulent production rate in this region, the LES data shows negative values for the production of turbulent kinetic energy. This occurs because while most sub-grid-scale (SGS) models used in LES simulations are dissipative, implying that they predict energy to be dissipated from the large scales to the smaller ones, the SGS stresses can still transfer energy back to the large scales through a process known as backscatter. In contrast, RANS Baseline simulations do not account

for backscatter [63]. They predict the dissipation and production of k in this region to be in balance while considering that ω at the wall should be computed according to the boundary condition imposed by Menter and implemented by the `omegaWallFunction` in OpenFOAM. Therefore, the corrections in this region result from the $k-\omega$ SST equations adjusting the balances between their production and dissipation terms to account for the LES behavior.



(a) b_{ij}^{Δ} magnitude correction field. The color bar has been limited to an upper value of 0.5 to bring out the distribution of the corrections.

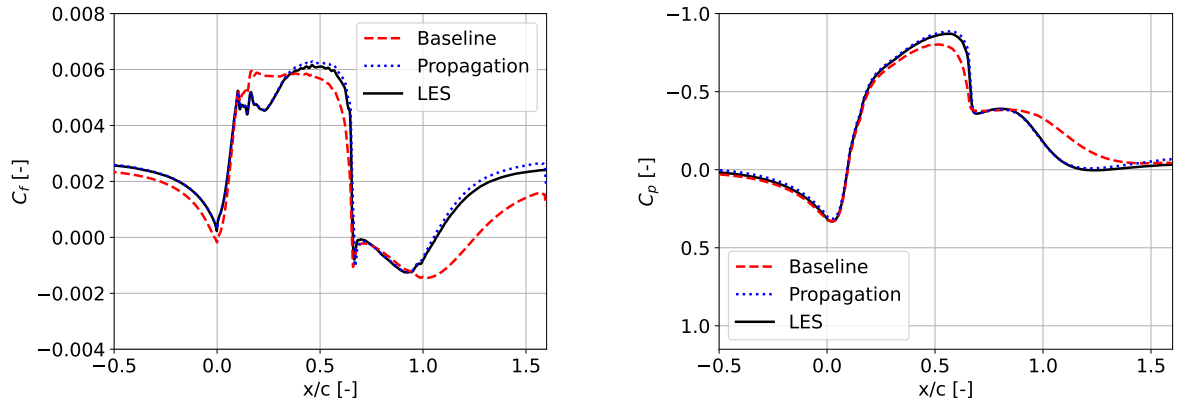


(b) R magnitude correction field. The color bar has been limited to an upper value of 60000 m^2s^{-3} to bring out the distribution of the corrections.

Figure 6.3: NASA-Hump correction fields obtained from Frozen simulations.

6.1.3. Propagation Results

Propagation simulations are always initiated from the converged Baseline simulation to eliminate inconsistencies that can be caused by differences in initial conditions. Furthermore, these simulations apply the b_{ij}^{Δ} and R correction fields in the entire computational domain.

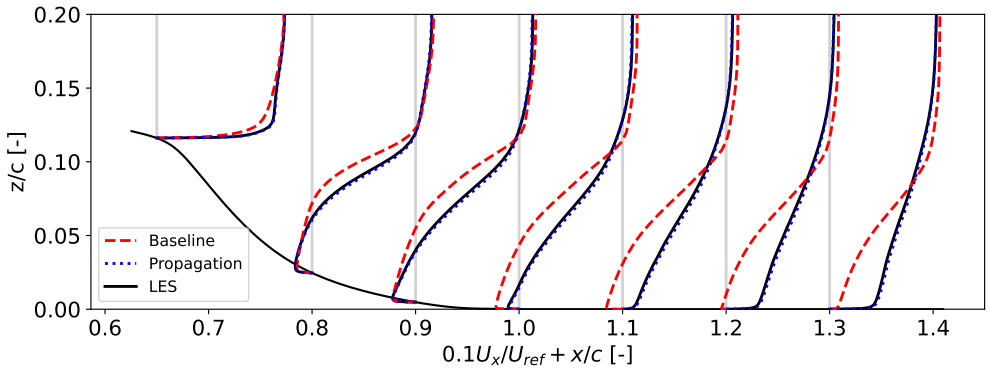


(a) C_f vs x/c

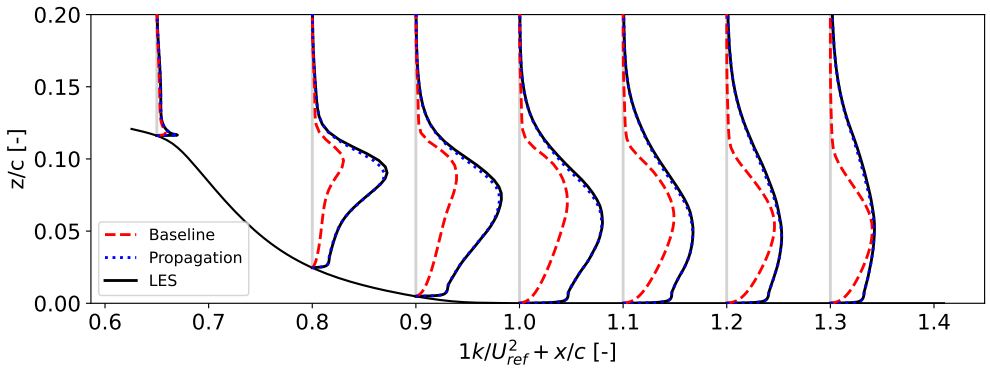
(b) C_p vs x/c

Figure 6.4: Comparison of C_f and C_p coefficients along the bottom wall boundary of the NASA-Hump domain, obtained from Baseline, Propagation and LES simulations.

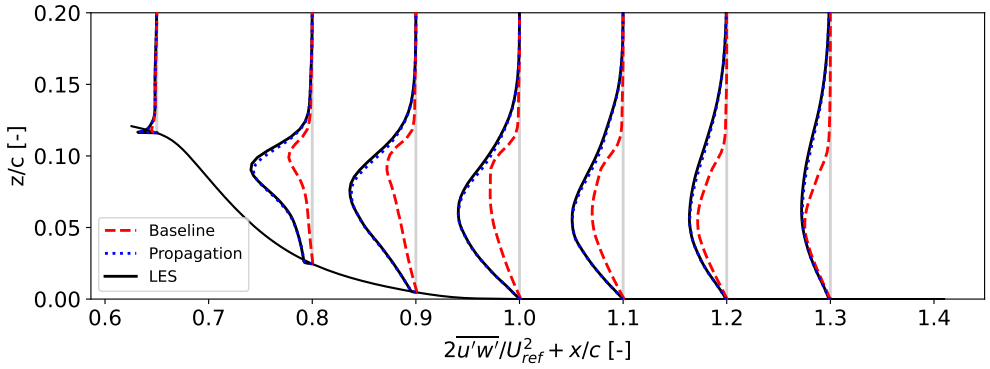
Figure 6.4 compares the C_f and C_p plots obtained from Baseline, Propagation and LES, while Figure 6.5 displays the profile comparisons of U_x, k, and $\overline{u'w'}$ downstream of the hump. Both figures show a near-perfect alignment between the Propagation results and the LES predictions, which validates the correction fields obtained from the Frozen simulation. Nevertheless, near the domain outlet, there is a slight discrepancy visible between Propagation and LES in the C_f and C_p plots. As explained in [8], this is thought to originate from the fact that the RANS domain is too short and does not allow for full boundary layer recovery.



(a) U_x profiles.



(b) k profiles.

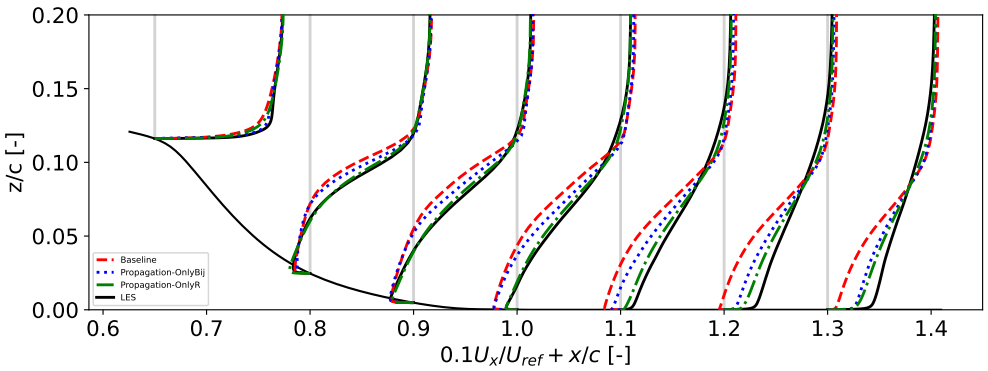


(c) $\overline{u'w'}$ profiles.

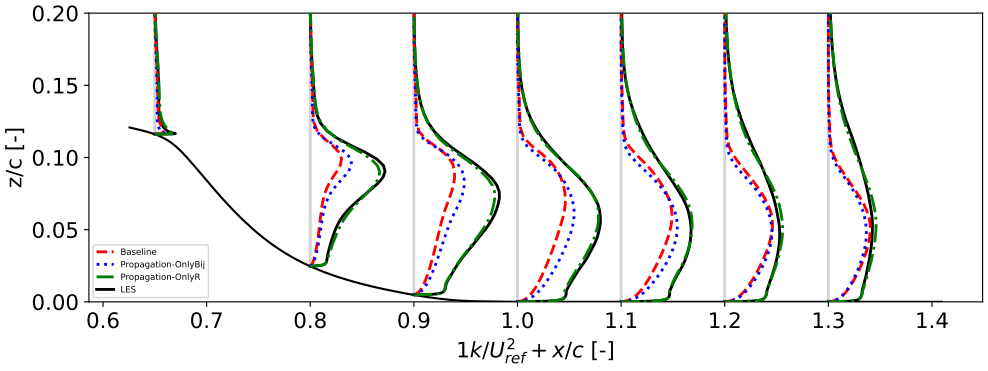
Figure 6.5: Comparison of Baseline, Propagation and LES profiles of U_x, k and $\overline{u'w'}$ at various x/c location downstream of the hump.

To assess the relative importance of each correction field, two additional Propagation simulations have been conducted: one based solely on the b_{ij}^{Δ} correction field, and another based only on the R field. The resulting profiles are displayed in Figure 6.6. It is evident from the U_x profiles that the Propagation simulation based on the R correction field significantly outperforms the one based solely on the b_{ij}^{Δ}

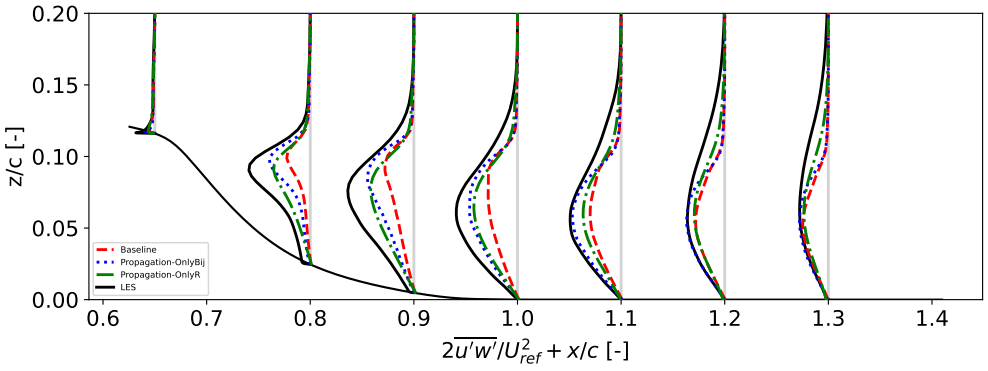
corrections. As seen from the k profiles, almost all the error between the Baseline RANS and the LES data stems from the model-form error in the $k-\omega$ SST model, so when this is corrected for, an almost perfect prediction of k is obtained. However, even with this near-perfect prediction of k, the U_x profiles still do not fully match the LES ones. This discrepancy is caused by the fact that the R corrections still result in an underprediction of the shear stress, as evidenced by the $\overline{u'w'}$ profiles. The model-form error in the Reynolds-stress tensor is primarily caused by the inaccuracy in the prediction of the turbulence anisotropy, as can be observed from the fact that the b_{ij}^{Δ} correction field leads to a more accurate prediction of the shear stress. Nevertheless, both correction fields are needed for the Propagation simulation to match the LES data.



(a) U_x profiles.



(b) k profiles.



(c) $\overline{u'w'}$ profiles.

Figure 6.6: Comparison of U_x, k, and $\overline{u'w'}$ profiles at various x/c locations downstream of the hump, obtained from Propagation simulation with only b_{ij}^{Δ} corrections active and Propagation simulation with only R corrections active.

Thus far, the b_{ij}^{Δ} and R correction fields have been applied across the entire computational domain. However, the main goal of this study is to apply corrections selectively, only within regions where the RANS

model is known to under-perform, i.e., the shear layer in the case of 2D-separated flows. Therefore, it is crucial to assess the impact of limiting corrections solely to the shear layer, which is identified by the RITA/TI classifier discussed in Chapter 5. Furthermore, since the SpaRTA model discovery relies only on the simulation and corrective field data obtained from the shear layer cluster, it is important to have a reliable reference point for comparing the models' performances. This investigation is further elaborated in the section below.

6.1.4. Propagation Restricted To Classifier Region

Before evaluating the effects of applying corrections exclusively in the shear layer cluster obtained through the RITA/TI classifier, an essential aspect of the classification process must be addressed: whether the shear layer cluster should remain static throughout the simulation (i.e., read from a file) or be calculated dynamically (updated at each iteration based on changes in flow variables as corrections are applied). A compelling argument in favor of dynamic clustering arises from the differences observed in the size and location of the recirculation bubble between the Baseline simulations and the Propagation simulations/LES predictions. Both LES and Propagation simulations display a smaller recirculation bubble. Additionally, as demonstrated in Figure 6.5, the peaks in the shear component of the Reynolds-stress tensor, which are indicative of the shear layer's location, differ between the Propagation/LES and the Baseline simulations. Therefore, since the shear layer location is expected to change, the clustering should also be updated to account for the changing physics. The approach used to update the cluster dynamically is documented in the modeling note below.

Dynamic Clustering with RITA/TI classifier

As a reminder, the RITA/TI classifier, here denoted as σ, is based on the thresholds set on the RITA_{P_k/D_k} ratio and the turbulence intensity and assigns a value of 0 or 1 for every mesh cell in the domain, according to:

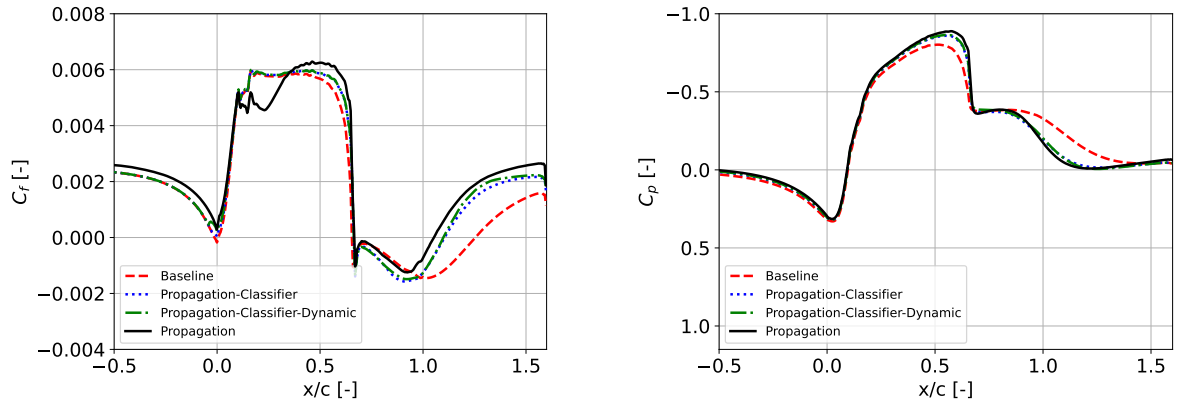
If: $\text{RITA}_{P_k/D_k} < 0.5$ and $TI \geq 0.12 : \sigma = 1$ **Else**: $\sigma = 0$.

The mesh cells where $\sigma = 1$ indicate the location of the shear layer. The mesh cells where $\sigma = 0$ represent the rest of the domain.

Both b_{ij}^{Δ} and R correction fields alter P_k. However, since R can take both positive and negative values, it can act as both a production and a destruction term. Therefore, the RITA_{P_k/D_k} ratio is reformulated to take this into account:

If: $R < 0$, then $\text{RITA}_{P_k/D_k} = \frac{D_k+R}{D_k+P_k+R}$ **Else:** $R > 0$, then $\text{RITA}_{P_k/D_k} = \frac{D_k}{D_k+P_k+R}$

Two Propagation simulations have been conducted to compare the effectiveness of dynamic versus static clustering in capturing the evolving physics of the shear layer region. In the first simulation, corrections are applied in the static shear layer. This type of simulation is referred to as a Propagation-Classifier simulation or PC for short. In the second simulation, the shear layer is dynamically updated, and the simulation is referred to as a Propagation-Classifier-Dynamic simulation, or PCD for short.

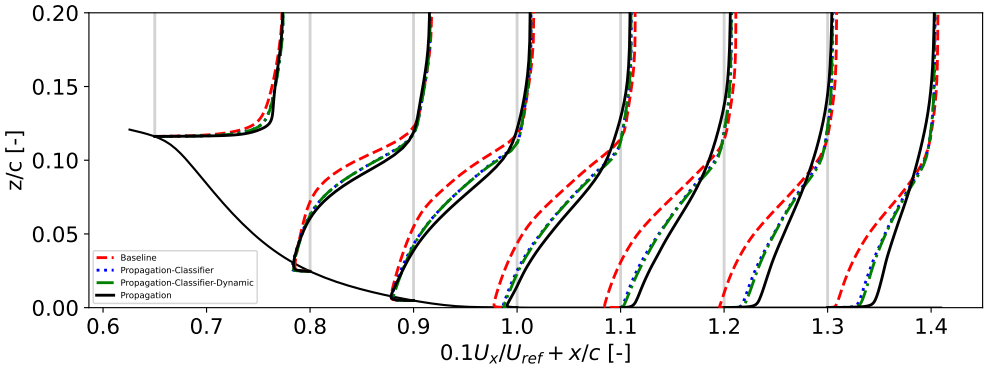


(a) C_f vs x/c

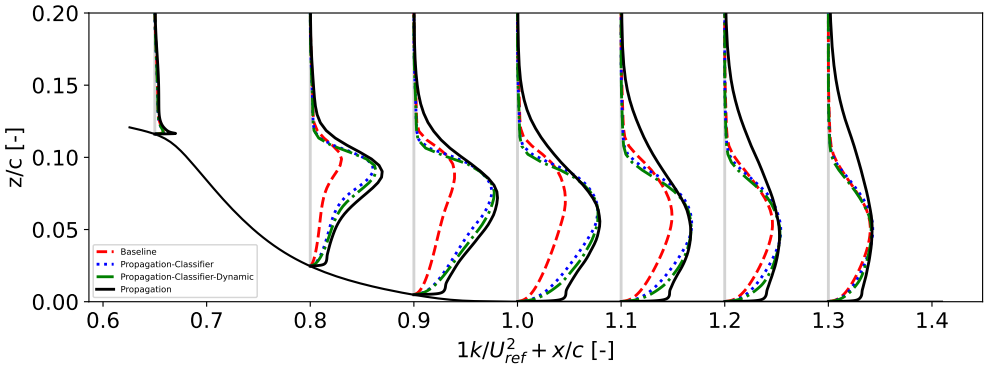
(b) C_p vs x/c

Figure 6.7: Comparison of C_f and C_p coefficients along the bottom wall boundary of the domain obtained from Baseline, Propagation, PC and PCD simulations.

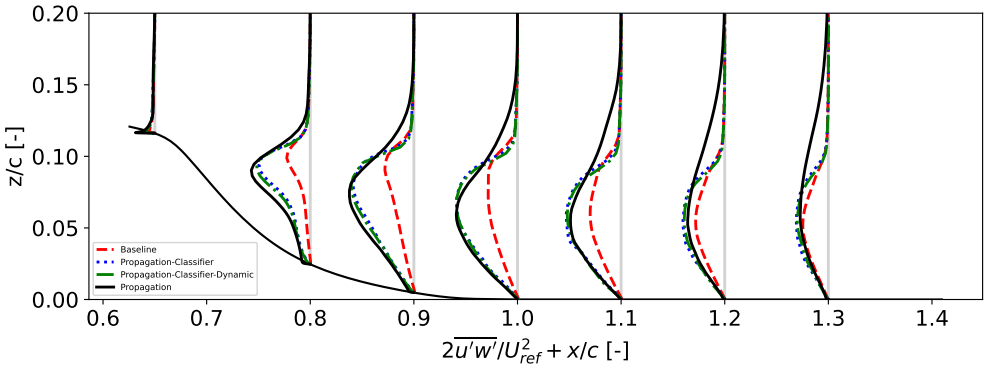
The U_x, k and $\overline{u'w'}$ profiles obtained from these two different types of simulations are displayed in Figure 6.8. The C_f and C_p coefficients are shown in Figure 6.7 and the exact separation and reattachment locations are displayed in Table 6.1.



(a) U_x profiles.



(b) k profiles.



(c) $\overline{u'w'}$ profiles.

Figure 6.8: Comparison of Baseline, Propagation, PC and PCD profiles of U_x, k and $\overline{u'w'}$ at various x/c locations downstream of the hump.

Both PC and PCD simulations show that applying corrections only in the shear layer results in a fairly improved prediction over the Baseline RANS, especially in the shear layer and recirculation region. Nevertheless, in the region directly above the shear layer, the discrepancies start to become more significant as this region is not included in the shear layer cluster. It is important to notice that there are no discontinuities in the profiles, even though the RITA/TI classifier performs a hard classification, so that in the shear layer cluster, full corrections are applied, and directly outside of its boundaries, no corrections are applied. As no convergence problems have been encountered during the simulations, there is no immediate need for a blending function to be applied at the boundaries of the cluster region so that the

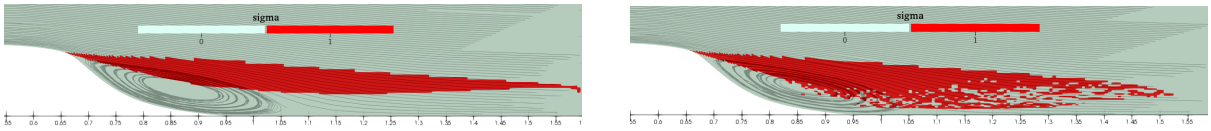
corrections are gradually eliminated.

Table 6.1: Overview of the separation and reattachment locations obtained from LES, Propagation, Baseline, PC and PCD simulations for the NASA-Hump case.

Simulation Type	Separation Location x/c [-]	Reattachment Location x/c [-]
LES	**0.66**	**1.06**
Propagation	0.66	1.05
Baseline	0.65	1.25
PC	0.66	1.10
PCD	0.66	1.09

In the near-wall region downstream of the hump, there is a significant difference in the k profile prediction by both PC and PCD simulations. As seen in the correction fields of R and b_{ij}^{Δ} in Figure 6.3, high corrective values are present for both fields in this near-wall region. Since the classifier simulations do not apply any corrections in this region, discrepancies in the profiles are not unexpected.

The PCD simulation shows only a marginal improvement over the PC simulation, which slightly underpredicts k in the shear layer. Overall, both simulations demonstrate a significant improvement over the Baseline simulation. However, they still overpredict the size of the recirculation region as evidenced by the reattachment locations listed in Table 6.1. To observe the physical changes in the shear layer cluster in the PCD simulation, Figure 6.9 has been created. The size of the shear layer cluster grows and curves around the shrinking recirculation region. The boundaries of the cluster also become less well-defined, which might be due to the interrupted patterns seen in the R correction field shown in Figure 6.3b. Since R is used to compute the RITA classifier dynamically, as documented in the above modeling note, its overall distribution effects are also present in the final classification.



(a) Propagation-Classifier **(b)** Propagation-Classifier-Dynamic

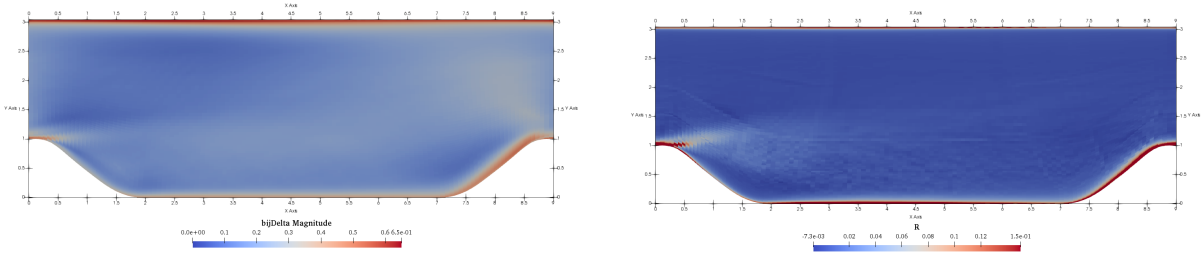
Figure 6.9: Comparison of static and dynamic RITA/TI classification. R and b_{ij}^{Δ} correction are only applied where σ=1.

6.2. Periodic-Hill

6.2.1. b_{ij}^{Δ} and R Correction Fields

In the Periodic-Hill case, similar to the NASA-Hump, flow separation occurs due to an adverse pressure gradient created by a change in the shape of the bottom wall contour. However, the top wall boundary is now a wall, which introduces additional effects on the outer-flow pressure field acting on the boundary layer undergoing separation. As stated in [41], the actual ability of RANS models to predict separation for this case is difficult to interpret because even small modeling errors can translate to significant differences in predictive performance due to the periodic boundary conditions that can amplify these errors.

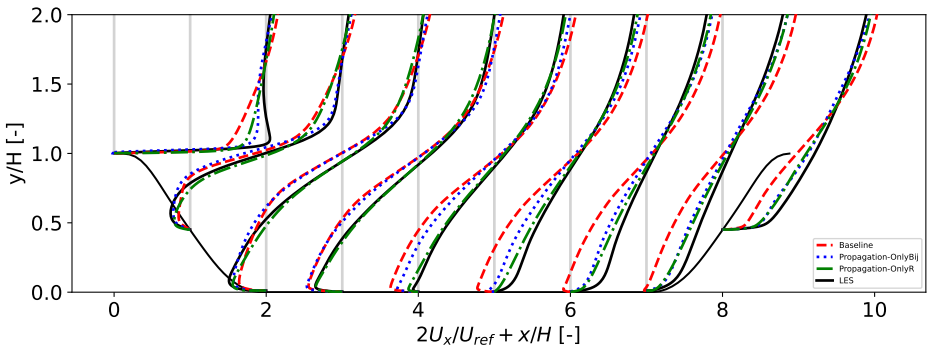
The b_{ij}^{Δ} and R correction fields for this case are displayed in Figure 6.10. To enhance the visualization of the corrections, the color bar for R has been constrained to an upper value of 0.15 m^2s^{-3}. Analogous to the NASA-Hump case, significant corrections are observed at the point of separation, in the shear layer and the near-wall region. To assess the relative importance of each correction field, two distinct Propagation simulations were conducted: one based solely on the b_{ij}^{Δ} correction field, and another only based on the R field. Figure 6.11 presents a comparison of the simulation results with the available LES data for this case.



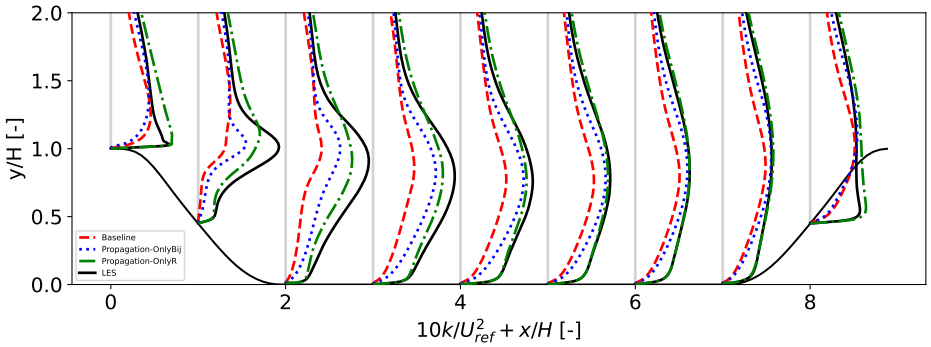
(a) b_{ij}^{Δ} magnitude correction field.

(b) R magnitude correction field.

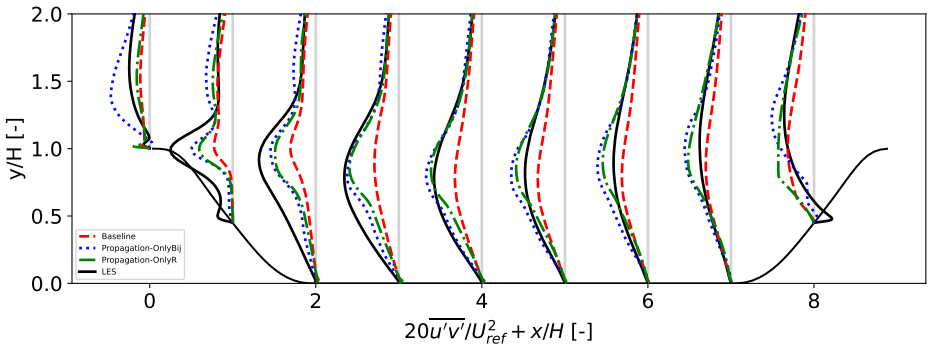
Figure 6.10: Periodic-Hill correction fields obtained from Frozen simulations.



(a) U_x profiles.



(b) k profiles.



(c) $\overline{u'v'}$ profiles.

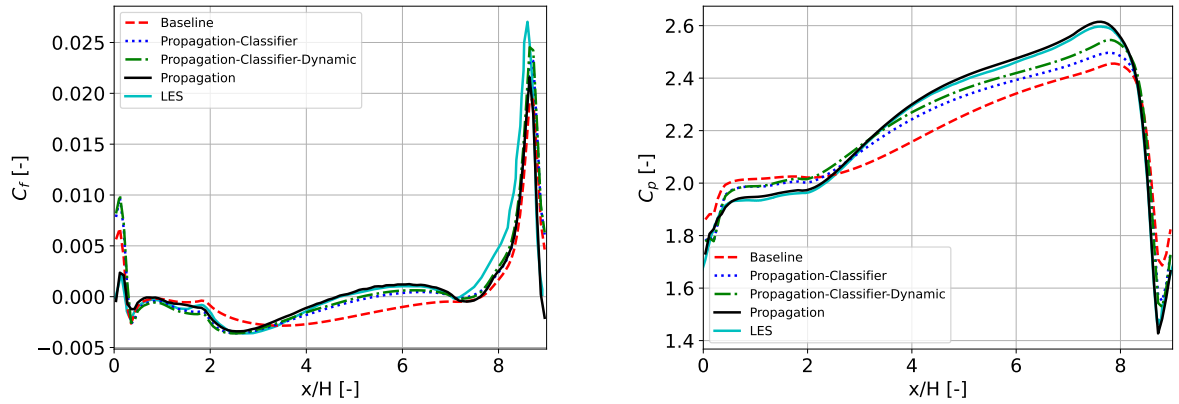
Figure 6.11: Comparison of U_x, k, and $\overline{u'v'}$ profiles at different x/H locations along the Periodic-Hill domain, obtained from Propagation simulations with only b_{ij}^{Δ} corrections active and Propagation simulation with only R corrections active.

The Propagation simulation based on the R correction field shows the best overall performance in predict-

ing the U_x velocity profile. As can be inferred from the k and $\overline{u'v'}$ profiles, while the turbulence anisotropy correction plays a more significant role in predicting the correct magnitude of the Reynolds-stress tensor, it still does not lead to an accurate prediction of the production of k, and therefore k remains underestimated. On the other hand, correcting for the model-form error in the k-ω SST model results in a better prediction of k, which in turn leads to a better prediction of the U_x velocity profile. It is important to note, that these trends align with the ones obtained for the NASA-Hump case.

6.2.2. Comparison of Full Propagation vs Classifier Propagation Simulations

Conducting a full Propagation simulation, incorporating both b_{ij}^{Δ} and R fields, along with the two types of classifier Propagation simulations (PC and PCD), yields the C_f and C_p plots showcased in Figure 6.12, alongside the flow profiles depicted in Figure 6.13. To facilitate the comparison of flow separation and reattachment locations across different simulations, Table 6.2 is provided.



(a) C_f vs x/H **(b)** C_p vs x/H

Figure 6.12: Comparison of C_f and C_p coefficients across the bottom wall boundary of the Periodic-Hill domain, obtained from LES, Baseline, Propagation, PC and PCD simulations.

Table 6.2: Overview of the separation and reattachment locations obtained for the Periodic-Hill case from LES, Baseline, Propagation, PC and PCD simulations.

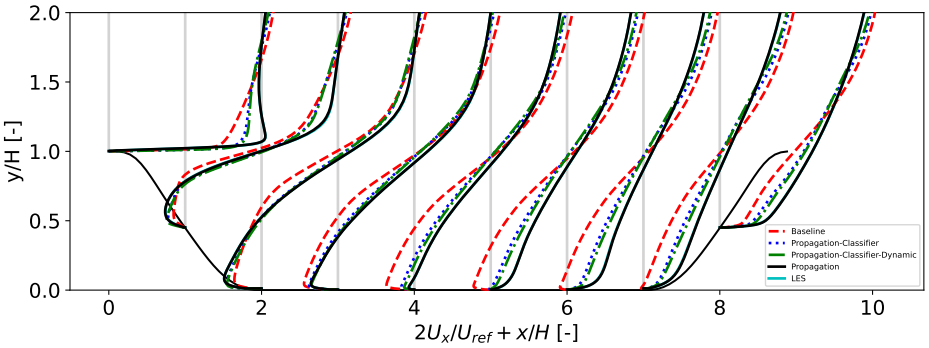
Simulation Type	First Separation Location x/H [-]	First Reattachment Location x/H [-]	Second Separation Location x/H [-]	Second Reattachment Location x/H [-]
LES	**0.22**	**4.69**	**7.01**	**7.22**
Propagation	0.25	4.64	7.06	7.62
Baseline	0.26	7.64	-	-
PC	0.29	5.43	7.04	7.38
PCD	0.30	5.15	7.03	7.48

The C_p plot shows a steep rise immediately after the first hill crest, indicating flow separation, before remaining relatively constant across the downstream half of the recirculation zone. The C_f plot predicts the separation point at around $x/H = 0.22$, with a minimum in skin friction near $x/H = 0.28$ where the region of maximum reverse flow is located. C_f returns to zero at $x/H = 4.69$ as the flow decelerates, consistent with a positive pressure gradient associated with the narrowing of the recirculation zone. Post-reattachment, influenced by a moderately adverse pressure gradient, the developing boundary layer decelerates as it approaches the subsequent hill, leading to a secondary incipient separation around $x/H = 7.22$. A local skin friction minimum at this point coincides with an increase in the C_p plot. Finally, the flow accelerates over the second hill crest as evidenced by a sharp rise in C_f and a sharp decrease in C_p.

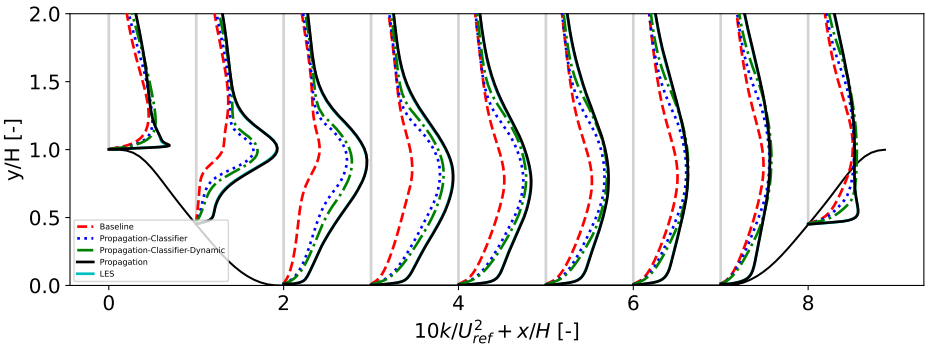
Table 6.2 highlights that the Baseline RANS simulation fails to predict this secondary incipient separation, and places the reattachment of the first separation at a location further downstream compared to the

LES prediction. Additionally, Figure 6.13 illustrates that k and $\overline{u'v'}$ are heavily underestimated in the shear layer of the Baseline simulation, leading to an overly large recirculation region. This aligns with the trends observed for the NASA-Hump case.

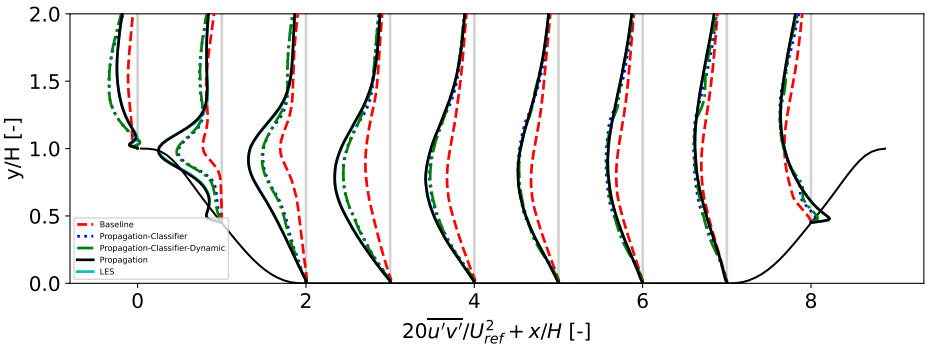
The Propagation simulation profiles align perfectly with the LES profiles, with only minor discrepancies observed in the C_f and C_p plots. These differences likely stem from the fact that the C_f and C_p data were obtained from the original LES mesh, whereas the U_x, k, and $\overline{u'v'}$ LES profiles were obtained by interpolating LES data to the RANS mesh. Consequently, slight variations can arise due to information loss during interpolation.



(a) U_x profiles.



(b) k profiles.



(c) $\overline{u'v'}$ profiles.

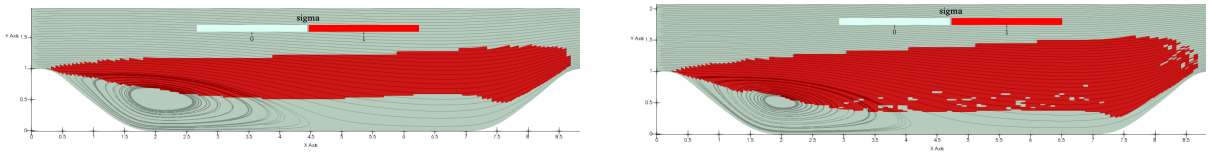
Figure 6.13: Comparison of U_x, k and $\overline{u'v'}$ profiles obtained at different x/H locations along the Periodic-Hill domain from LES, Baseline, Propagation, PC and PCD simulations.

Although the Propagation simulation successfully captures both the primary separation and the secondary incipient separation, it notably overestimates the secondary reattachment location in comparison to LES. Conversely, both PC and PCD simulations demonstrate superior performance in overall separation and reattachment point prediction for this secondary separation region. The underlying reasons for this discrepancy are not discernible from the profiles presented in Figure 6.13, primarily due to the rela-

tively small size of the secondary separation region compared to the primary one, making it challenging to capture the physics accurately. Nevertheless, given that the primary separation is deemed the most critical, the models' performance is primarily assessed based on their predictions in this region. The PC and PCD simulations tend to overpredict the first reattachment location compared to the standard Propagation simulation and LES data. This discrepancy can be explained by the underprediction of the k and $\overline{u'v'}$ profiles by these simulations.

In general, the PCD simulation shows improvement over the PC simulation in predicting the reattachment location. However, it performs even worse than the Baseline simulation in predicting the separation location. This discrepancy can be attributed to the $\overline{u'v'}$ profiles, which show an overestimation at the top of the hill crest. The PCD simulation's overestimation of shear stress at this location leads to the flow remaining attached for longer, delaying separation. Similar behavior is observed in the PC simulation.

This can be traced back to the cluster of the shear layer obtained for these simulations, as depicted in Figure 6.19. The cluster initiates at the edge of the first hill crest and terminates at the onset of the second hill crest. Notably, no corrections are applied right above the hill crest, despite the correction field plots indicating high corrective values in this region (as illustrated in Figure 6.10). Since no corrections are applied in these regions due to their exclusion from the shear layer cluster, this omission is likely the source of the overstress observed in this area.



(a) Propagation-Classifier simulation.

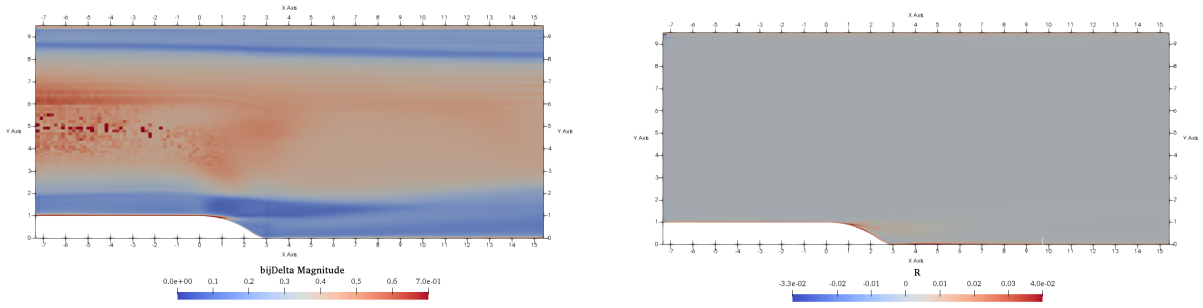
(b) Propagation-Classifier-Dynamic simulation

Figure 6.14: Comparison of static and dynamic RITA/TI classification for the Periodic-Hill case. R and b_{ij}^{Δ} correction are only applied where σ=1.

6.3. Curved Backward Facing Step

6.3.1. b_{ij}^{Δ} and R Correction Fields

In the CBFS case, similar to the NASA-Hump and Periodic-Hill, flow separation initiates due to an adverse pressure gradient caused by a change in the contour shape of the bottom wall. This case stands out due to the challenges involved in simulating flows separating from a gently curved surface, in contrast to the sharper geometries of Periodic-Hill or NASA-Hump [41]. The gradual shift in surface curvature allows the flow to smoothly adapt and trace the surface contour, leading to a slight delay in separation and the formation of a small recirculation region. No abnormalities were encountered in obtaining the correction fields for this case or in the convergence and probe studies. The correction fields for b_{ij}^{Δ} and R are illustrated in Figure 6.10.



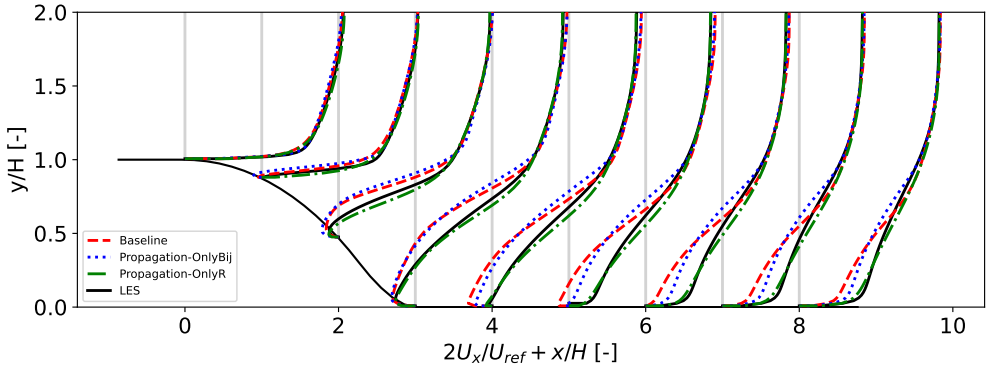
(a) b_{ij}^{Δ} magnitude correction field. The color bar is limited to an upper value of 0.7 to bring out the distribution of the corrections.

(b) R magnitude correction field. The color bar is limited to an upper value of 0.04 m^2s^{-3} to bring out the distribution of the corrections.

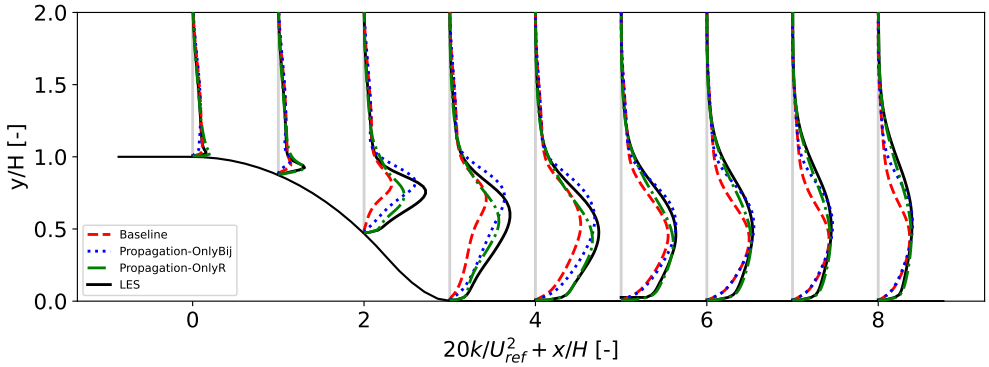
Figure 6.15: CBFS correction fields obtained from Frozen simulations.

Similar to the NASA-Hump and Periodic-Hill cases, there are significant corrections at the point of separation, in the initial stretch of the shear layer and the near-wall region. To assess the relative importance

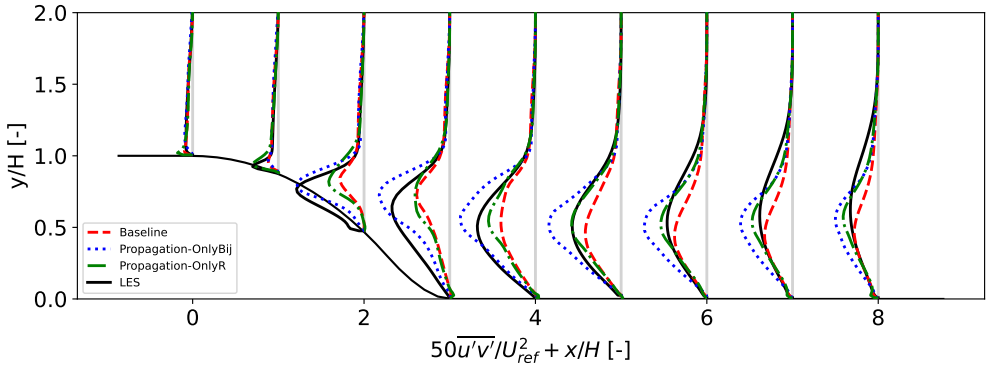
of each correction field, two distinct Propagation simulations were conducted: one based solely on the b_{ij}^{Δ} correction field, and another based only on the R field. Figure 6.16 presents a comparison of the resulting flow profiles.



(a) U_x profiles.



(b) k profiles.



(c) $\overline{u'v'}$ profiles.

Figure 6.16: Comparison of U_x, k, and $\overline{u'v'}$ profiles at different x/H locations along the CBFS domain for Propagation simulation with only b_{ij}^{Δ} corrections active and Propagation simulation with only R corrections active.

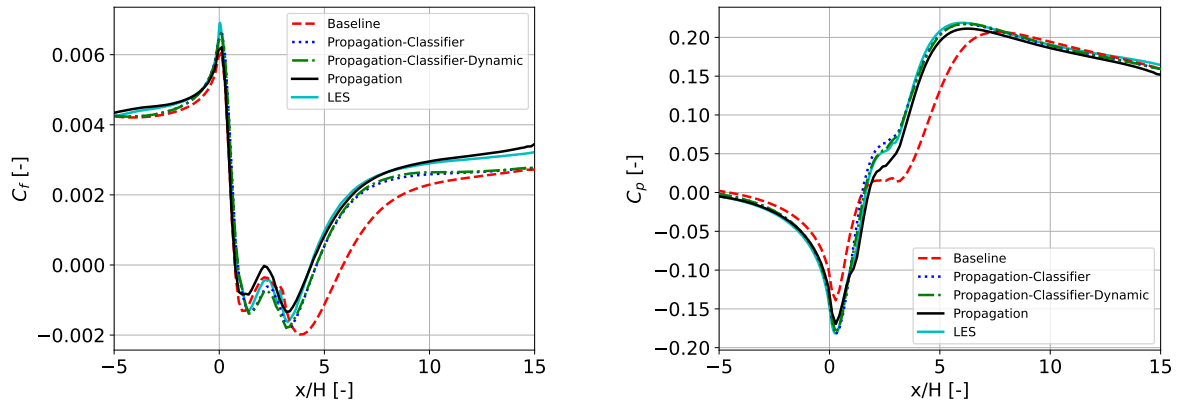
Similar to the NASA-Hump and Periodic-Hill cases, the Propagation simulation based on the R correction fields outperforms the one based solely on the b_{ij}^{Δ} corrections, as deduced from the prediction of the U_x profiles. The $\overline{u'v'}$ profiles indicate that applying only R corrections is sufficient to obtain the correct estimation of the shear stress in the post-reattachment region. However, the shear stress in the shear layer remains largely underpredicted. Further downstream, the $\overline{u'v'}$ profile begins to be overestimated, leading to the overestimation observed in the U_x profiles in this region. The R corrections are also adequate to achieve very good predictions of k in this region, including the near-wall region in the recirculation bubble. However, k also remains slightly underpredicted in the shear layer. On the other

hand, applying only b_{ij}^{Δ} corrections leads to an overprediction of both k and $\overline{u'v'}$ in the shear layer and fails to predict the correct maximum peak location in the $\overline{u'v'}$ profiles.

It is interesting to observe that despite the overpredictions in $\overline{u'v'}$ and k in the b_{ij}^{Δ} Propagation simulation, the resulting U_x profile does not show any significant overprediction, and the solution appears quite similar to the Baseline simulation. This discrepancy can be understood by examining the $\overline{u'v'}$ and k profiles at $x/H = 1$, which is just after the point of separation. While the R corrective simulation yields predictions at these locations similar to the LES data, the b_{ij}^{Δ} simulation resembles the Baseline prediction. Referring back to the correction field plots in Figure 6.15, it is evident that the corrections are most significant in this region so predicting the correct amount of turbulence production here, is crucial for capturing the correct physics in the shear layer and recirculation region. Therefore, because b_{ij}^{Δ} corrections fail to predict the correct turbulence energy production in this region, it has a cascading effect on the predictions of the rest of the flow behavior. Overall, while both b_{ij}^{Δ} and R corrections are vital for the accurate prediction of this simulation, the model-form error in the $k-\omega$ SST model appears to play a more significant role than the error in the anisotropy of the Reynolds-stress tensor. This aligns with the results obtained for the Periodic-Hill and NASA-Hump cases.

6.3.2. Comparison of Full Propagation vs Classifier Propagation Simulations

Conducting a full Propagation simulation, incorporating both b_{ij}^{Δ} and R fields, along with the two types of classifier Propagation simulations (PC and PCD), yields the C_f and C_p plots showcased in Figure 6.17b, alongside the flow profiles depicted in Figure 6.18. To facilitate the comparison of flow separation and reattachment locations across different simulations, Table 6.3 is provided. The incoming boundary layer undergoes a certain level of acceleration upstream of the curved step, as evidenced by the sharp increase and subsequent decrease in the C_f and C_p plots respectively. As mentioned in [41], this acceleration is attributed to distortions in the velocity field in the downstream curved section, where the attached near-wall flow maintains near-constant vorticity as it curves downward over the step. The flow experiences deceleration after x/H = 0, leading to a rise in pressure over the surface and eventual separation estimated at $x/H = 0.83$ by the LES data. In the recirculation region, fluctuations in the C_f plot are attributed to curvature discontinuities in the contour of the step. Surface pressure rises close to the reattachment point, beyond which the boundary layer begins to recover towards an equilibrium channel condition past x/H = 10.



(a) C_f vs x/H **(b)** C_p vs x/H

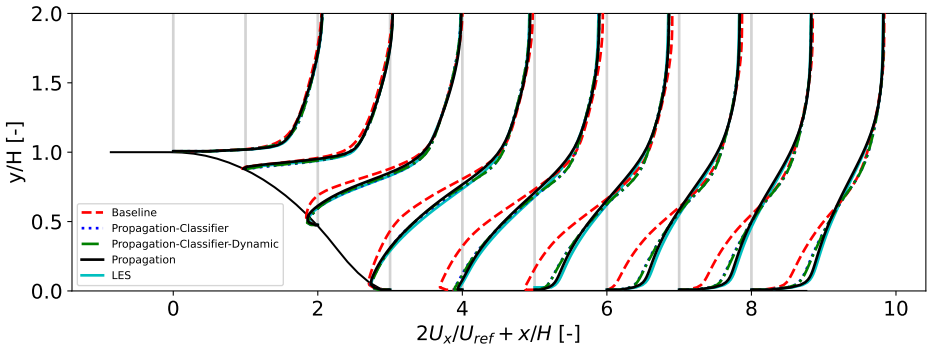
Figure 6.17: Comparison of C_f and C_p coefficients across the bottom wall boundary of the CBFS domain, obtained from LES data, Baseline, Propagation, PC and PCD simulations.

Overall, a slight mismatch between the Propagation and LES data is evident in both the flow profiles and the pressure and the C_f and C_p plots, which is more pronounced compared to the Periodic-Hill and NASA-Hump cases. It is important to note that the computational domain for this case employs a much coarser mesh than the other cases in this study, with an average y^+ value of 3.63 for the bottom wall. As observed from the correction fields and the trends in the Propagation simulations where only one of two correction fields was applied per simulation, accurate predictions near the separation region, especially in the near-wall region, are crucial for correctly predicting the overall flow behavior. Therefore,

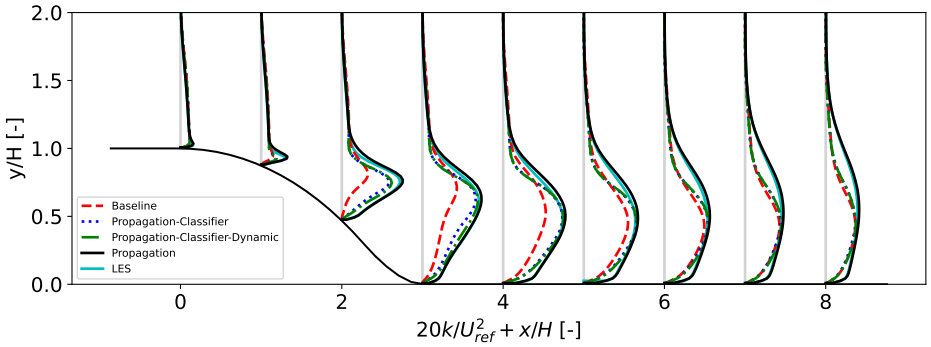
it is possible that because the mesh is coarser, the boundary layer is not as well-resolved, leading to a slight mismatch between the Propagation simulation and LES data. This is further supported by the fact that the Propagation simulation predicts the separation location earlier than the LES data.

Table 6.3: Overview of the separation and reattachment locations obtained for the CBFS case from LES, Baseline, Propagation, PC and PCD simulations.

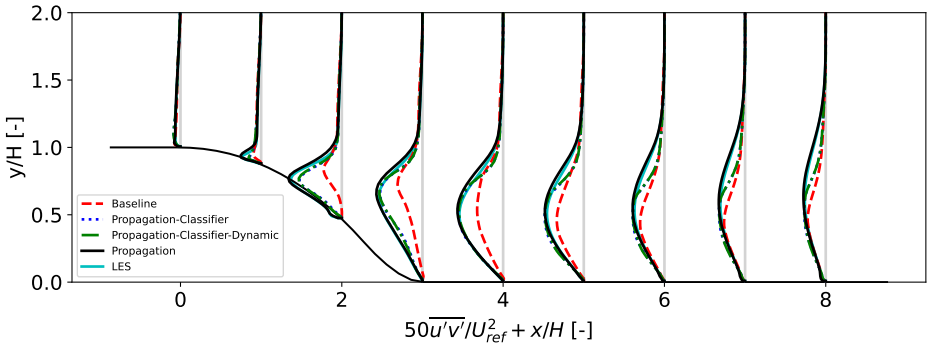
Simulation Type	Separation Location x/H [-]	Reattachment Location x/H [-]
LES	**0.82**	**4.32**
Propagation	0.75	4.35
Baseline	0.74	5.85
PC	0.89	4.59
PCD	0.89	4.58



(a) U_x profiles.



(b) k profiles.

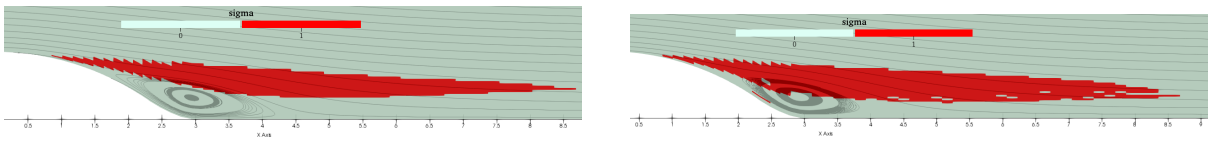


(c) $\overline{u'v'}$ profiles.

Figure 6.18: Comparison of the profiles of U_x, k and $\overline{u'v'}$ obtained at different x/H locations along the CBFS domain from LES data, Baseline, Propagation, PC and PCD simulations.

Nevertheless, the Propagation simulation still manages to obtain a relatively accurate prediction of the reattachment location and, therefore, of the overall size of the recirculation bubble. The slight mismatch observed in the C_f and C_p plots towards the outlet of the domain between LES and Propagation is very likely due to the same reason as the one given for the mismatch observed in the outlet region of the NASA-Hump case, which is that the RANS domain is not long enough to fully capture the boundary layer recovery.

The discrepancies between the PC and PCD simulations are negligible, they both predict the same separation location, with only a slight 0.01 difference in the reattachment location. The most significant difference lies in the fact that the PC simulation slightly underpredicts k in the initial stretch of the shear layer. As depicted in Figure 6.19, similar to the NASA-Hump and Periodic-Hill cases, the shear layer cluster grows and curves over the shrinking recirculation region. Overall, applying corrections solely in the shear layer cluster results in a significant improvement over the Baseline simulation, nearly achieving an exact match with the LES.



(a) Propagation-Classifier simulation.

(b) Propagation-Classifier-Dynamic simulation

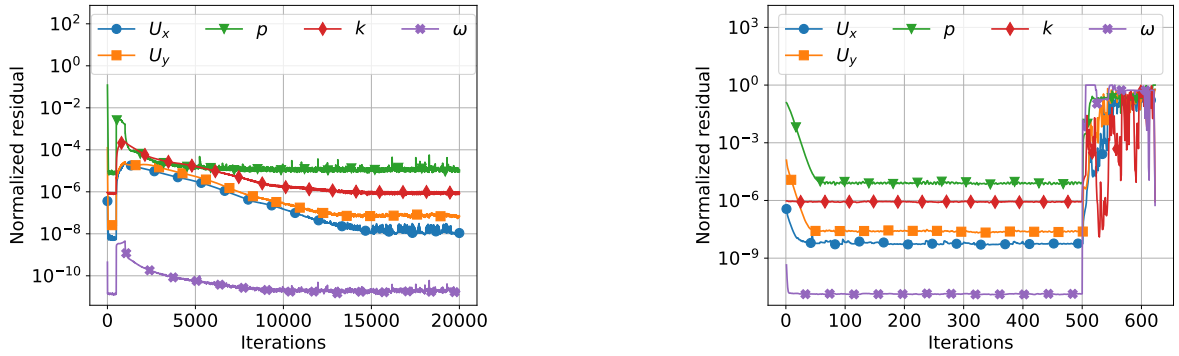
Figure 6.19: Comparison of static and dynamic RITA/TI classification for the CBFS case. R and b_{ij}^{Δ} correction are only applied where $\sigma = 1$.

6.4. APG Case

6.4.1. b_{ij}^{Δ} and R Correction Fields

The APG case is slightly different compared to the other 2D-separated flow cases since the adverse pressure gradient is no longer created by a contour change in the bottom boundary of the domain and there is no flow acceleration prior to separation. In the APG case, separation is induced by imposing an adverse pressure gradient on a fully turbulent zero-pressure gradient boundary layer. This pressure gradient arises from a modification to the top boundary of the domain, resulting in flow expansion. Following the separation region, a favorable pressure gradient is followed by a zero pressure gradient of long enough duration to allow the boundary layer to fully recover [42]. The recirculation bubble that develops in the APG case is extremely small compared to the other 2D-separated flow cases.

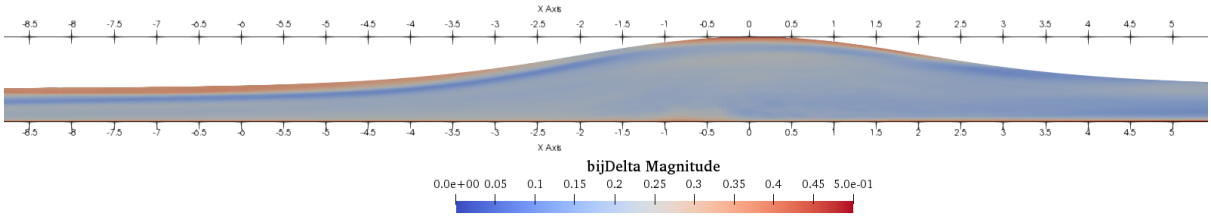
To obtain the R and b_{ij}^{Δ} correction fields for the APG case, the R term had to be removed from the ω equation. If this term is not removed, the Propagation does not converge as can be seen from the residuals in Figure 6.20. The intuition behind removing R from the ω equation comes from previous experience of researchers at the TU Delft aerodynamic group who realized that keeping R in the ω equation when dealing with 3D cases is very unstable.



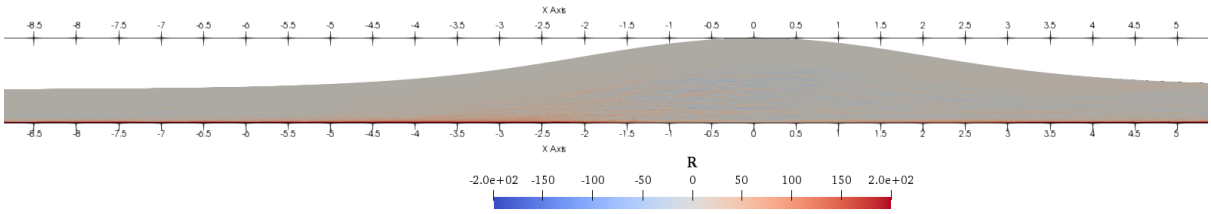
(a) Residuals vs the number of iterations for Propagation simulation with R term **removed** from the ω equation.

(b) Residuals vs the number of iterations for Propagation simulation with R term **not removed** from the ω equation.

Figure 6.20: Residual plots obtained during the APG Propagation simulation run.



(a) b_{ij}^{Δ} magnitude correction field. The color bar has been limited to an upper value of 0.5 to bring out the distribution of the corrections



(b) R magnitude correction field. The color bar has been limited to upper and lower values of 200 and -200 m^2s^{-3} respectively to bring out the distribution of the corrections.

Figure 6.21: APG correction fields obtained from Frozen simulations.

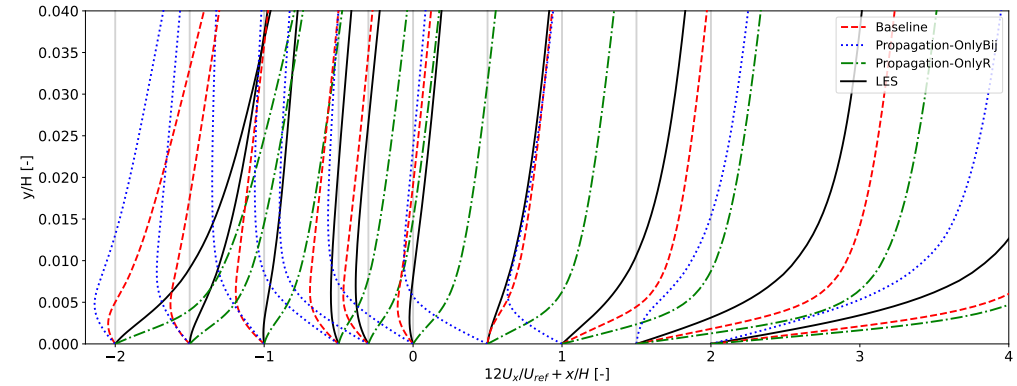
Figure 6.21 displays the b_{ij}^{Δ} and R fields. Similar to the correction fields obtained for the other 2D-separated flow cases, there are high corrections at the wall and near the separation point at around $x/H = -1.47$. Corrections in the shear layer are present in both fields however they are not as recognizable as in the other cases. To assess the relative importance of each correction field, two distinct Propagation simulations were conducted: one based solely on the b_{ij}^{Δ} correction field, and another based only on the R field. Figure 6.22 presents a comparison of the resulting flow profiles.

Both types of simulations result in very poor estimations of the U_x profiles. The corrections applied with b_{ij}^{Δ} lead to severe underestimations, while those with R corrections result in overpredictions. Since these corrections have opposite effects, it helps explain why using both in a Propagation simulation yields near-perfect predictions of the flow, closely resembling the DNS results, as depicted in Figure 6.23 in the section below.

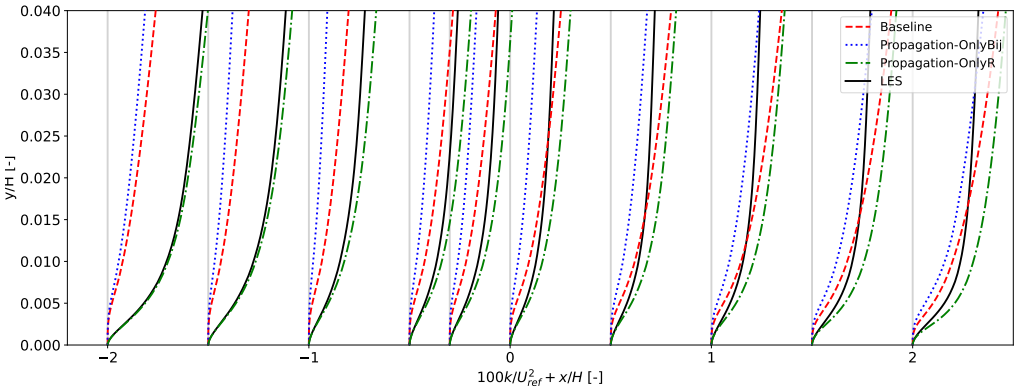
The R corrections lead to overpredictions of turbulence production, as indicated by the k profiles. The cause of this can be related to approaches used to derive the correction fields in the Frozen simulation. The R residual term in the ω equation acts to locally increase or decrease dissipation. However, this term was eliminated from the ω equation during the Frozen simulation. This is problematic because the production term of the ω equation is based on the production term of the k equation as shown in equation 2.42. Therefore, in this case, if the R residual term locally increases the production in the k equation, this is not accounted for in the ω equation, which leads to an underprediction in the amount of dissipation in the system. This ultimately results in an overprediction in k, as the balance between turbulent energy production and dissipation is no longer accounted for correctly.

On the other hand, the b_{ij}^{Δ} corrections severely underpredict both k and $\overline{u'v'}$ profiles, resulting in a poorer performance compared to the Baseline simulation. Once again, this can be attributed to the methods used to derive the correction fields. To correct for turbulence anisotropy, the eddy viscosity needs to be computed since it is not provided by the DNS data. Computing the eddy viscosity requires ω to be known, as indicated in equation 2.31. However, as discussed earlier, the ω field is not correctly computed as there is no term to account for the additional local dissipation. Consequently, the overall turbulence anisotropy correction does not entirely represent the true model-form error as it is biased due to the absence of corrections to the ω field.

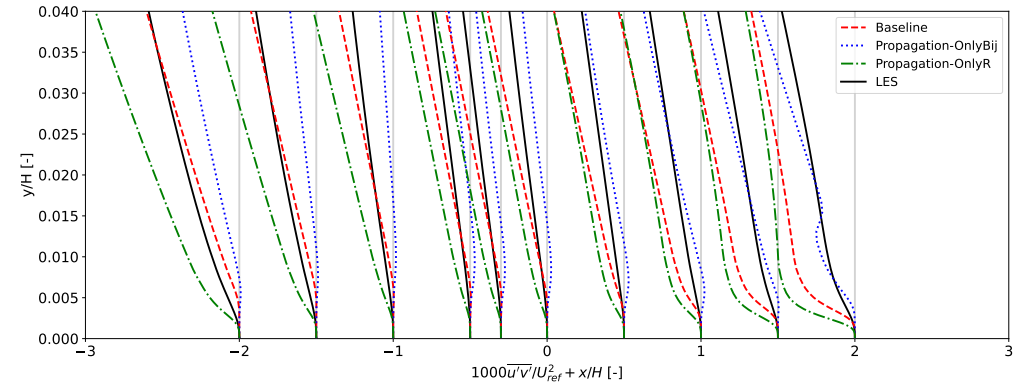
These observations are very important for the SpaRTA model discovery procedure. Since the R and b_{ij}^{Δ} correction fields, in this case, do not fully capture the true model-form errors, integrating these fields into the SpaRTA regression could potentially yield adverse effects on the quality of the models derived through this approach. Consequently, multiple regression runs of SpaRTA were performed without including the APG case alongside the other 2D-separated flow cases. This was done to ascertain whether its inclusion introduces biases into the model predictions. This is further documented in Chapter 7.



(a) U_x profiles.



(b) k profiles.

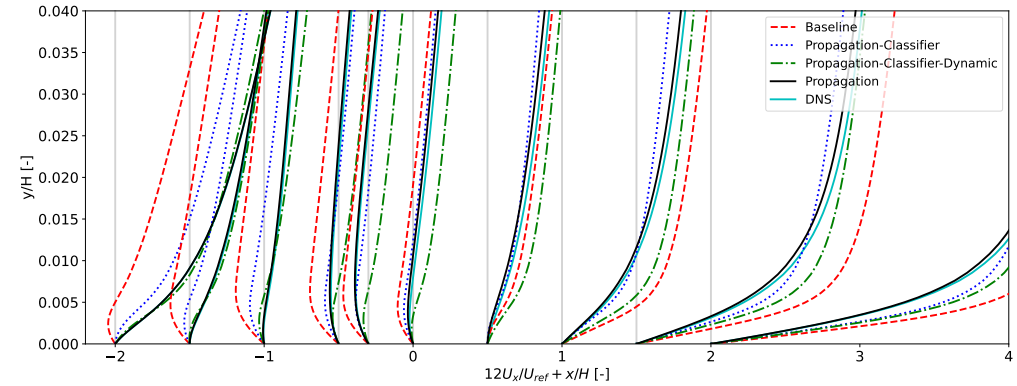


(c) $\overline{u'v'}$ profiles.

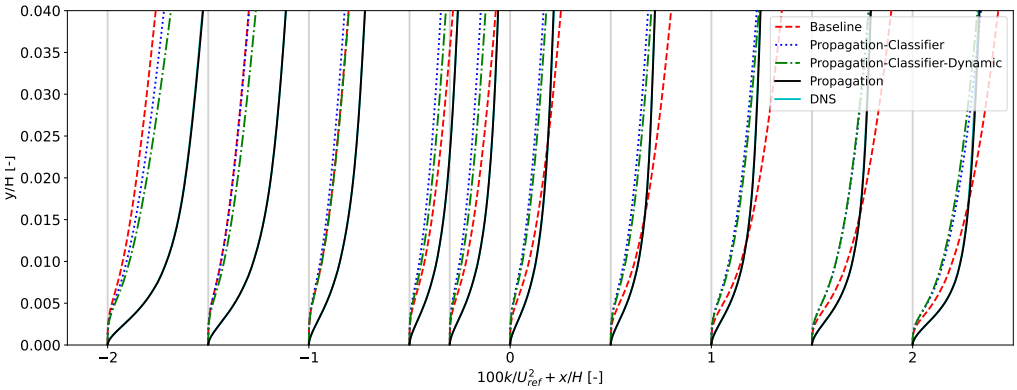
Figure 6.22: Comparison of U_x, k, and $\overline{u'v'}$ profiles at different x/H locations along the APG domain for Propagation simulation with only b_{ij}^{Δ} corrections active and Propagation simulation with only R corrections active.

6.4.2. Comparison of Full Propagation vs Classifier Propagation Simulations

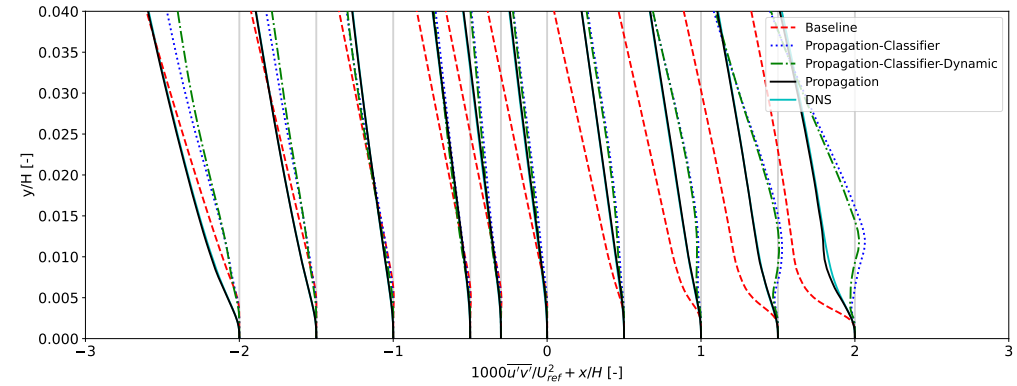
Conducting a full Propagation simulation, incorporating both R and b_{ij}^{Δ} corrections, along with the two types of classifier Propagation simulations (PC and PCD), yields the C_f and C_p plots showcased in Figure 6.24, alongside the flow profiles depicted in Figure 6.23. It is important to take note of the y-axis range of the flow profiles, which is set betweeny/H = 0 and 0.04, meaning that these figures show the profiles in a very small region close to the wall. This was done because the recirculation region is very small and the computational domain is very large. To facilitate the comparison of flow separation and reattachment locations across different simulations, Table 6.4 is provided.



(a) U_x profiles.



(b) k profiles.



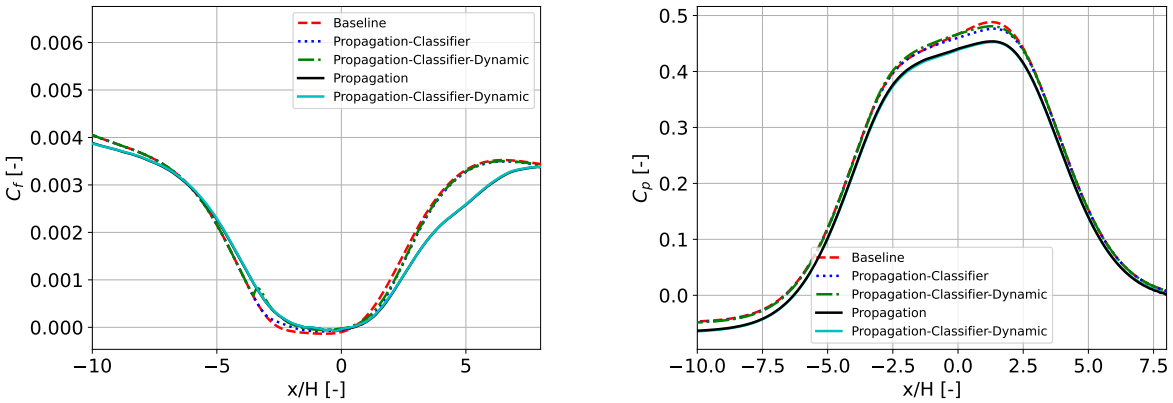
(c) $\overline{u'v'}$ profiles.

Figure 6.23: Comparison of U_x, k and $\overline{u'v'}$ at different x/H locations along the APG domain, obtained from DNS, Baseline, Propagation, PC and PCD simulations.

The C_f and C_p plots reveal that the flow decelerates under an adverse pressure gradient as the surface pressure rises towards the middle of the computational domain. Unlike the other 2D-separated flow cases, there is no flow acceleration prior to separation. As can be seen from Table 6.4, the DNS simulation predicts two separation regions that lie near each other, which is not accounted for by any of the other types of simulations. Therefore, while the Propagation simulation provides a very good match with the DNS flow profile as can be seen from Figure 6.23 and C_f and C_p plots in Figure 6.23, it is still unable to distinguish between the two separation regions.

An interesting observation lies in the difference between the Baseline simulation and DNS data regarding the location of the separation point. The Baseline simulation can predict the reattachment location much more accurately than the separation location. This represents a complete reversal of the trends observed in other 2D-separated flow cases. The Baseline simulation predicts an earlier separation because it

underpredicts the shear stresses near the separation location compared to DNS, which allows the flow to separate earlier. Post-separation, the shear stresses start to become overpredicted in the shear layer, which leads to an overall better prediction of the reattachment location compared to the other cases that underpredict the shear stress in the shear layer. These observations can be traced back to the fact that while the other flow cases have some level of flow acceleration prior to separation, this case does not. Flow acceleration decreases the thickness of the boundary layer which increases the size of the shear stress, keeping the flow attached for longer.



(a) C_f vs x/c **(b)** C_p vs x/c

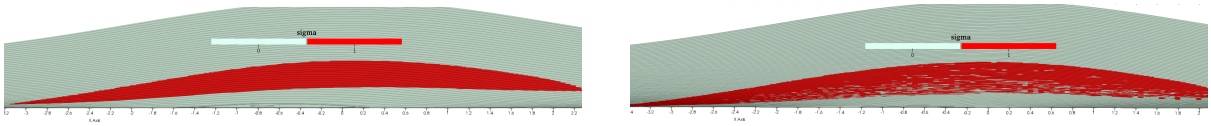
Figure 6.24: Comparison of C_f and C_p coefficients across the bottom wall boundary of the APG domain, obtained from DNS, Baseline, Propagation, PC and PCD simulations.

Table 6.4: Overview of the separation and reattachment locations obtained for the APG case from DNS, Baseline, Propagation, PC and PCD simulations.

Simulation Type	First Separation Location x/H [-]	First Reattachment Location x/H [-]	Second Separation Location x/H [-]	Second Reattachment Location x/H [-]
DNS	**-1.47**	**-1.25**	**-1.14**	**0.40**
Propagation	-1.47	0.48	-	-
Baseline	-2.51	0.48	-	-
PC	-1.95	0.48	-	-
PCD	-1.64	0.28	-	-

The U_x profiles reveal that in the pre-separation region, applying corrections only in the shear layer cluster results in an improved prediction over the Baseline simulation, with the PCD simulation showing the highest accuracy in predicting the separation location. On the other hand, while the PC simulation predicts an identical reattachment location as the Propagation simulation, the PCD predicts a much earlier position, thus excessively shrinking the recirculation bubble. This difference is easily observed in Figure 6.25, which illustrates the changes in the shear layer cluster due to the R and b_{ij}^{Δ} corrections applied in the shear layer. The recirculation bubble for the PCD simulation is barely visible in this figure.

The cluster in the PCD simulation elongates towards the inlet direction of the domain, bringing it closer to the wall. This helps explain why the separation point is more accurately predicted by the PCD simulation. Another important aspect to note from this figure is that the shear layer cluster is not computed to lie directly on top of the recirculation region, unlike in other 2D-separated flow cases. This observation may help explain the discrepancies observed between the PC and PCD simulations and the DNS data for this case. Overall, the PC simulation leads to a better prediction of the velocity profile in the reattachment and post-reattachment regions compared to the PCD simulation, and both types of simulations show an improvement over the Baseline.



(a) Propagation-Classifier **(b)** Propagation-Classifier-Dynamic

Figure 6.25: Comparison of static and dynamic RITA/TI classification for the APG case. R and b_{ij}^{Δ} correction are only applied where σ=1.

6.5. Final Conclusions

The Baseline simulations for all 2D-separated flow cases confirm the well-known weakness of RANS in predicting the correct behavior of a boundary layer subjected to an adverse pressure gradient. RANS fails to predict the turbulent shear stress above the recirculation region which overpredicts the size and location of the recirculation bubble. For the Periodic-Hill, NASA-Hump, and CBFS cases, where the adverse pressure gradient is created by a change in the contour shape of the bottom boundary of the domain and there is flow acceleration prior to separation, RANS primarily fails to predict the correct flow reattachment location. In the APG case, where the adverse pressure gradient is created by a change in the contour shape of the top boundary of the domain and there is no flow acceleration prior to separation, RANS primarily fails to predict the correct separation location. In all cases, this results in an overprediction of the size of the recirculation bubble.

The b_{ij}^{Δ} and R correction fields have been successfully obtained and validated for all cases. The Propagation simulations have shown a near-perfect match with the high-fidelity data in all cases. Slightly larger mismatches were observed for the CBFS and APG cases than for the Periodic-Hill and NASA-Hump cases. The model-form error in the $k-\omega$ SST model, accounted for by the R residual term, has been identified as the primary cause of failure in RANS simulations of 2D-separated flows. However, the correction for turbulence anisotropy, accounted for by b_{ij}^{Δ}, is also crucial in obtaining an accurate flow prediction.

The correction fields for the APG case could only be obtained by removing the R residual term from the ω equation. This holds significant implications for the SpaRTA model discovery. While the b_{ij}^{Δ} and R correction fields derived for the other 2D-separated flow cases represent the true model-form error in the k-ω SST model and the turbulence anisotropy computed under the Boussinesq approximation, the correction fields for the APG case do not. Consequently, attempting to derive a model for b_{ij}^{Δ} and R by fitting it to data obtained from all cases is unlikely to yield promising results, or at least, may not accurately represent the true model-form error. For this reason, SpaRTA regression runs have been performed excluding the data from this case in the analysis, as further discussed in Chapter 7.

Finally, it has been shown that applying corrections to a dynamically updated shear layer cluster throughout the simulation yields slightly improved flow predictions compared to applying these corrections to a static shear layer computed from the converged Baseline result. Therefore, when testing the models obtained through SpaRTA in Chapter 7, the shear layer cluster will be dynamically computed throughout the simulations. Furthermore, these results also answer the third research question formulated for this study, demonstrating that exclusively applying corrections in the shear layer region results in improved flow predictions in separated-flow scenarios compared to Baseline RANS.

7

Modelling Results

Regressing models using SpaRTA typically results in the identification of at least five models for both b_{ij}^{Δ} and R per training session. Only the most promising models are selected for further investigation, based on a compromise between the R^2 value of the fitted model and the number of terms. The minimum acceptable R^2 value is set to 0.9 for R models and 0.7 for b_{ij}^{Δ} models. This criterion is based on the fact that regressing models for R is much easier since it involves a scalar field. Conversely, b_{ij}^{Δ} represents a symmetric tensor field with six components, making regression a more complex task. The maximum number of terms is limited to four for each model, as having a large number of terms can lead to convergence issues when applied in a simulation. The models are validated in Model Propagation simulations, where the model equations are only activated in the shear layer cluster. The shear layer cluster is updated dynamically throughout the simulation to account for the effects introduced by the corrections, similar to the Propagation-Classifier-Dynamic (PCD) simulations introduced in Chapter 6.1.4. The models listed in this chapter have been selected across various SpaRTA training runs, all performed using different input conditions: using all available features and invariants, using a subset of features selected based on the outcome of a Mutual Information analysis and the removal of non-invariant and Reynolds number-dependent features, training only on the NASA-Hump case and excluding the APG case from the training set. This latter condition was used following the recommendations derived from the result analysis of the APG case Propagation results in Chapter 6 Section 6.4.1.

This chapter is structured as follows: Section 7.1 provides an overview of the models discovered for b_{ij}^{Δ}. These models have been chosen to provide a general overview of the ability of SpaRTA to regress model equations, as well as to provide an understanding of why some model equations, although showing the same R^2 coefficient value, perform very differently when applied in a simulation. Each b_{ij}^{Δ} model is tested in isolation (meaning that R field is read from its respective file obtained from the Frozen simulation). Section 7.2 provides a similar analysis of the identified R model equations. Finally, in section 7.4, the models for b_{ij}^{Δ} and R, which showed the highest performance in isolation testing, are combined in the same Model Propagation simulation and compared to similar models obtained from literature.

7.1. b_{ij}^{Δ} Models

7.1.1. Overview of Discovered Model Forms

In total, four b_{ij}^{Δ} models have been selected for further analysis. Table 7.1 provides an overview of the R^2 values of these models, the number of terms they consist of, and the SpaRTA training conditions used to obtain them. There is a clear relationship between the R^2 value of a model and the SpaRTA training conditions. For example, the R^2 value of Model1-Bij is the highest, as it was trained solely on data from the NASA-Hump case. The second highest R^2 value belongs to Model3-Bij, which was obtained by training on a dataset excluding the APG case. The number of terms appears to have a negligible effect on the R^2 value, as demonstrated by Model2-Bij and Model4-Bij, which exhibit the same R^2 despite having 1 and 2 terms respectively.

Table 7.1: General information regarding the obtained b_{ij}^{Δ} models.

Model I.D.	R^2 Value	Nr. of Terms	SpaRTA Training Conditions
Model1-Bij	0.93	3	NASA-Hump case: selected features and invariants.
Model2-Bij	0.7	1	All cases: selected features and invariants.
Model3-Bij	0.77	2	All cases excluding APG: selected features and invariants.
Model4-Bij	0.7	2	All cases: all features and invariants.

The formulations of the four b_{ij}^{Δ} models are given below, together with Table 7.2 which provides further insights into the different features and functions that appear in the model equations.

b_{ij}^{Δ} Model Formulations

Model1-Bij:
$$b_{ij}^{\Delta} = \underbrace{17.43\text{rdiv}\left(\frac{I_1}{0.0087}\right)\mathbf{T_{ij}^{(2)}}}_{\text{Term 1}} + \underbrace{36.73\text{rdiv}\left(\frac{I_1}{0.0087}\right)\mathbf{T_{ij}^{(3)}}}_{\text{Term 2}} + \underbrace{4.223\sqrt{\tau_{ij,ratio}}\mathbf{T_{ij}^{(2)}}}_{\text{Term 3}}$$

Model2-Bij:
$$b_{ij}^{\Delta} = 6.767\text{rdiv}\left(\tfrac{I_1}{0.0257}\right)\mathbf{T_{ij}^{(2)}}$$

Model3-Bij:
$$b_{ij}^{\Delta} = \underbrace{2.539\text{RITA}_{P_k/D_k}\mathbf{T_{ij}^{(2)}}}_{\text{Term 1}} - \underbrace{5.092\text{rdiv}\left(\frac{I_2}{0.0125}\right)\mathbf{T_{ij}^{(2)}}}_{\text{Term 2}}$$

Model4-Bij:
$$b_{ij}^{\Delta} = \underbrace{5.243\text{rdiv}\left(\frac{I_1}{0.0257}\right)\mathbf{T_{ij}^{(2)}}}_{\text{Term 1}} + \underbrace{1.280\text{rlog}\left(\frac{Re_k}{0.2405}\right)\mathbf{T_{ij}^{(2)}}}_{\text{Term 2}}$$

Table 7.2: Overview of the different features and functions used in the b_{ij}^{Δ} model formulations.

Symbol	Meaning	Additional Notes
I_1	Invariant: $I_1 = S_{ij}S_{ij}$	S_{ij} is the mean strain rate.
I_2	Invariant: $I_2 = \Omega_{ij}\Omega_{ij}$	Ω_{ij} is the mean rotation rate.
$\tau_{ij,ratio}$	Ratio of total to normal Reynolds stresses	Feature F9 - equation 4.30
$T_{ij}^{(2)}, T_{ij}^{(3)}$	Pope Tensor Basis	
RITA_{P_k/D_k}	RITA ratio of production to destruction of k	Feature F10 - equation 4.31
Re_k	Turbulence based Reynolds number	Feature F6 - equation 4.27
$\text{rdiv}(x)$	Function of the form: $\text{rdiv}(x) = \frac{x}{(1+x^2)}$	Plot in Figure 7.1
$\text{rlog}(x)$	Function of the form: $\text{rlog}(x) = \log(\lvert x\rvert + 1)$	Plot in Figure 7.1

The regression appears to have identified the $T_{ij}^{(2)}$ tensor basis as the most important for reconstructing the b_{ij}^{Δ} corrections. This is evident from the fact that almost all terms among all models use this basis, except for term 2 of Model1-Bij. Furthermore, many of the models rely on applying the rdiv() function to the invariant I_1. This invariant is computed from the mean strain rate (see Table 7.2). In the shear layer, the mean strain rate is expected to be quite high due to the large velocity gradients. At least one of the features listed in Section 4.4 is used in every model, except for Model2-Bij. This demonstrates the utility of using these types of features, which were specifically chosen to reflect the local turbulence characteristics in the regression.

The general shapes of the rdiv() and rlog() functions are displayed in Figure 7.1. The rdiv() function ensures that its output remains small even if its input is very large. In the models, rdiv() is used in terms where the I_1 and I_2 invariants are present. These invariants have small values in the range of [-1,1], so the rdiv() function will have a maximum and minimum value of 0.5 or -0.5 respectively. Therefore, even when terms have large coefficient values multiplied in front of the rdiv() functions, such as 17.43 and 36.73 for terms 1 and 2 of Model1-Bij respectively, they will be halved at minimum.The rdiv() function

also allows for negative outputs, unlike the rlog() function. This is important when considering term 2 of Model3-Bij. The I_2 invariant, based on the rotation rate tensors (see Table 7.2), has negative values, resulting in negative values for rdiv(I_2). Consequently, this makes the term positive due to the negative sign multiplying the coefficient in front of rdiv(I_2). Therefore, although it might seem that term 2 subtracts from the Model3-Bij corrections, it actually adds to them.

The rlog() function can output slightly higher values than rdiv(), depending on the input provided. However, the rlog() function is only used in term 2 of Model4-Bij, which depends on the turbulence based Reynolds number Re_k. As shown in Figure A.6 in Appendix Section A, which displays this feature's raw distribution[1], Re_k never exceeds a value of 0.6 in the shear layer. Thus, the rlog() function outputs fall within a comparable range to that of rdiv() function outputs based on the invariants. It is important to note that Re_k is a Reynolds number-dependent feature, which puts Model4-Bij at a disadvantage compared to the other models regarding its overall generalizability.

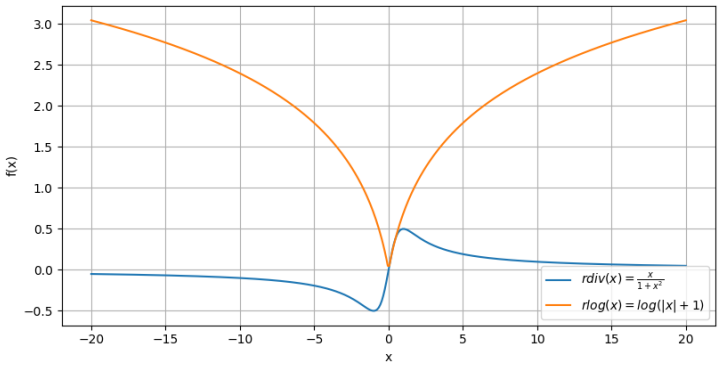


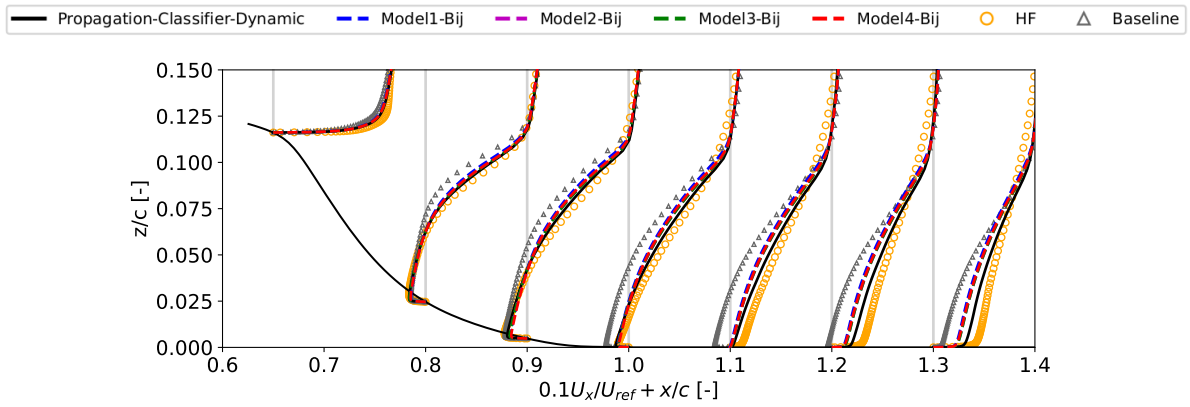
Figure 7.1: General shape of rdiv(x) and rlog(x) functions.

Term 3 of Model1-Bij and term 1 of Model3-Bij are the only two terms that do not have an rdiv() or rlog() function in front of the tensor basis. The features used for these terms, namely RITAP_k/D_k and $\tau_{ij,ratio}$, have values in the range of [0,1]. Therefore, multiplying these features by the coefficients in front of the terms does not result in large values. An important note regarding the models is that because SpaRTA can only perform linear regression, it can only regress coefficients outside the rdiv() and rlog() functions. Therefore, the approach used in SpaRTA is to divide all the features and invariants used in rdiv() and rlog() by their standard deviation. This is why, for example, in Model1-Bij term 1 and term 2 have the same factor in the denominator inside their rdiv() functions.

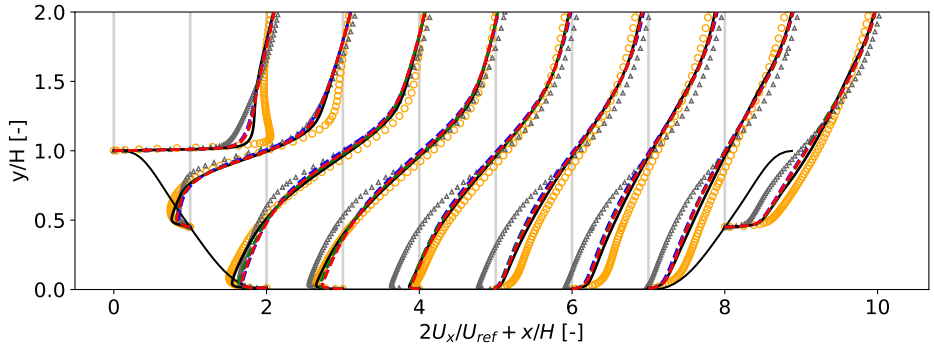
7.1.2. Modelling Results

This section presents the results obtained from hybrid Model Propagation simulations, where b_{ij}^{Δ} was modeled following the formulations discussed in the previous section, and R was read from its Frozen correction field. This setup allows for the analysis of b_{ij}^{Δ} models in isolation. All corrections are applied only in the shear layer cluster identified by the RITA/TI classifier, which is dynamically updated during the simulation. For comparison purposes, the results of the Baseline $k-\omega$ SST, the Propagation-Classifier-Dynamic (PCD) simulation, and the high-fidelity data (HF) (LES or DNS depending on the case) are also displayed. Profiles of the axial velocity U_x, the turbulent kinetic energy k, and the shear stress component of the Reynolds-stress tensor $\overline{u'v'}$ are displayed in Figures 7.2, 7.3, and 7.4 respectively. The separation and reattachment locations predicted by each Model Propagation simulation are listed in Table 7.3. As discussed in Chapter 6, some cases show both primary and secondary recirculation regions, where one region is more significant in terms of size and physical significance than the other. Therefore, only the separation and reattachment locations of the most relevant separation regions are listed in this table. Finally, Figure 7.5 presents a comparison of the eddy viscosity ν_t profiles.

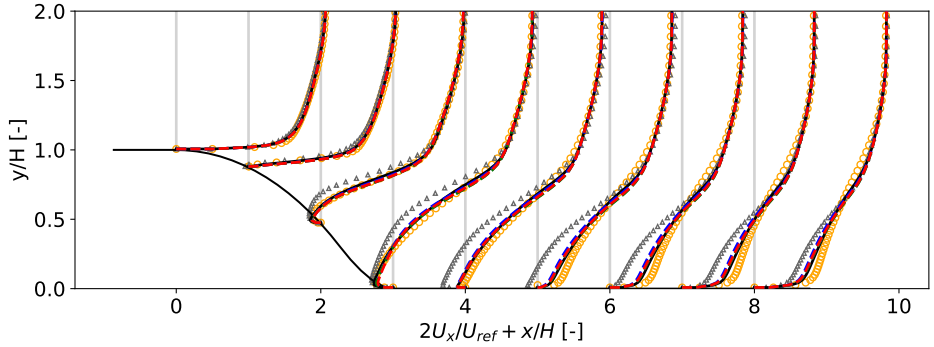
[1] Features like Re_k are used in SpaRTA regression after normalization to a [0,1] range. However, the plots of these features in the Appendix are presented in their unnormalized form.



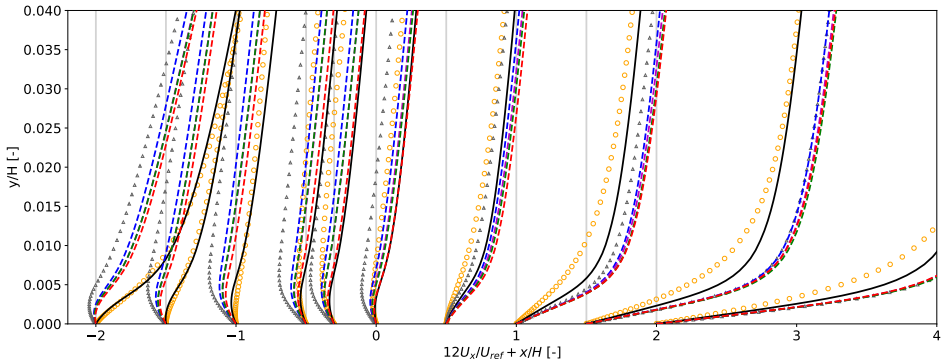
(a) NASA-Hump case.



(b) Periodic-Hill case.

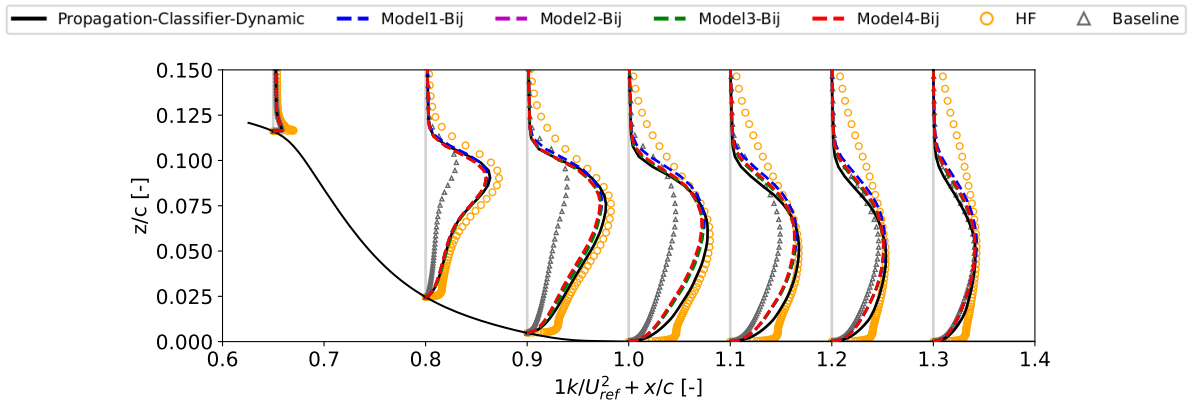


(c) CBFS case.

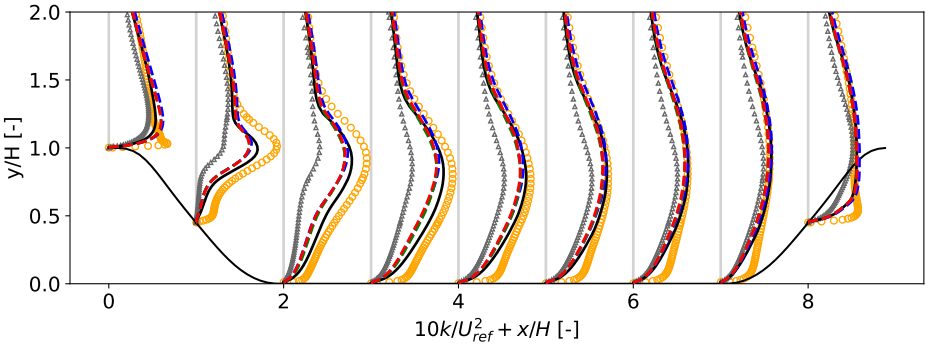


(d) APG case.

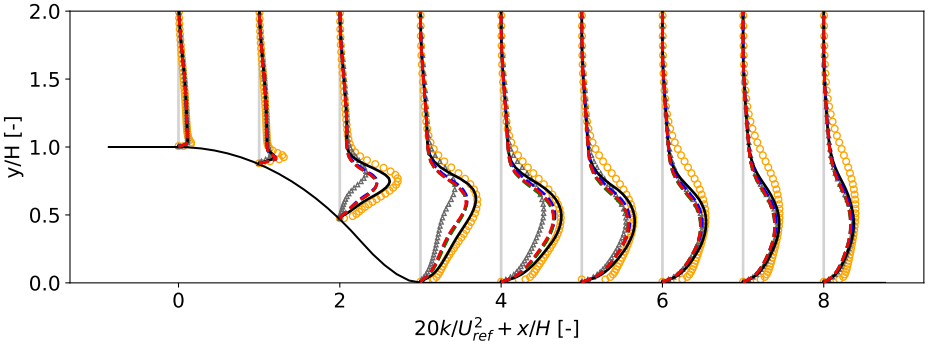
Figure 7.2: Comparison of U_x profiles obtained from the hybrid Model Propagation simulations based on the b_{ij}^{Δ} models run in isolation.



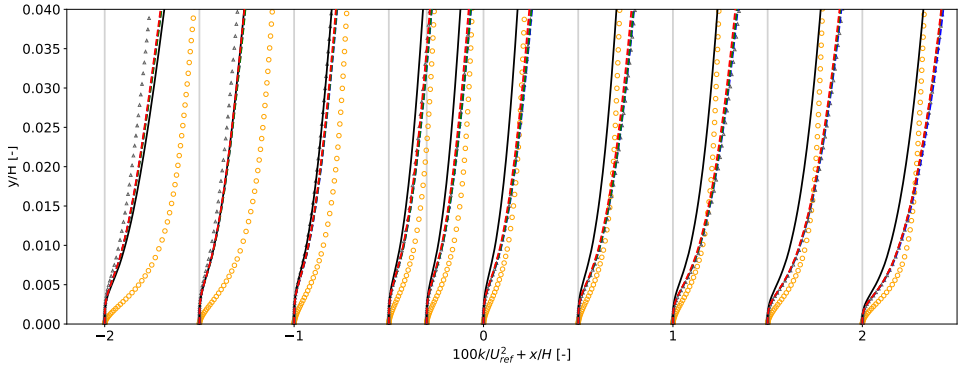
(a) NASA-Hump case.



(b) Periodic-Hill case.

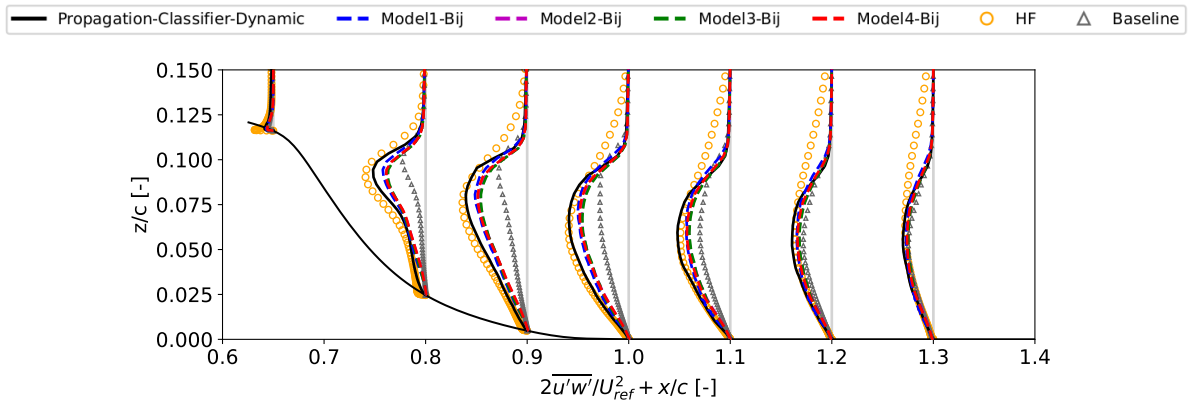


(c) CBFS case.

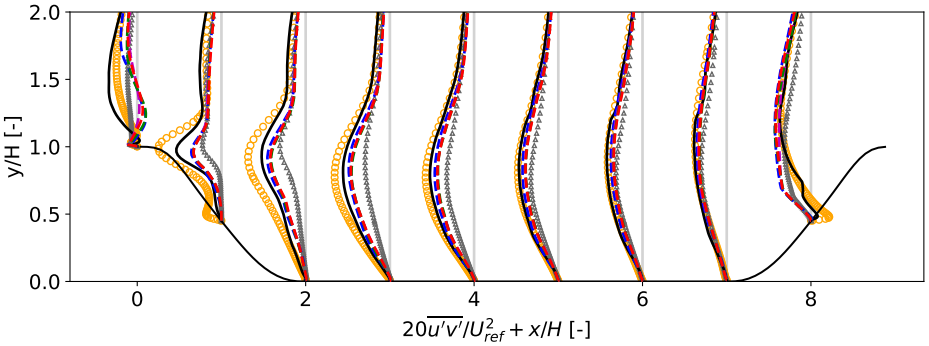


(d) APG case.

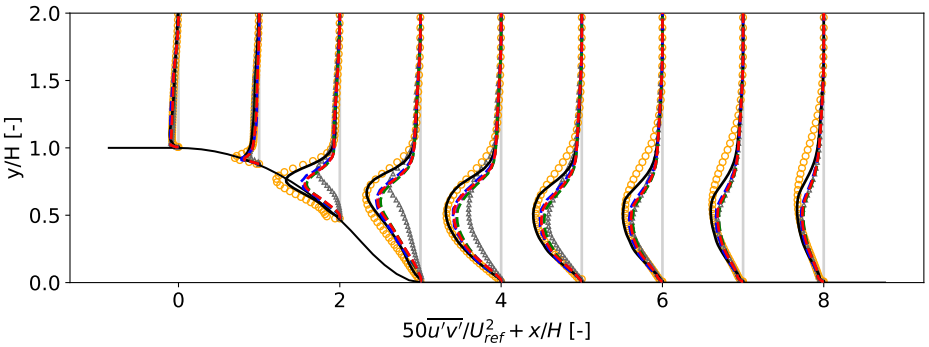
Figure 7.3: Comparison of k profiles obtained from the hybrid Model Propagation simulations based on the b_{ij}^{Δ} models run in isolation.



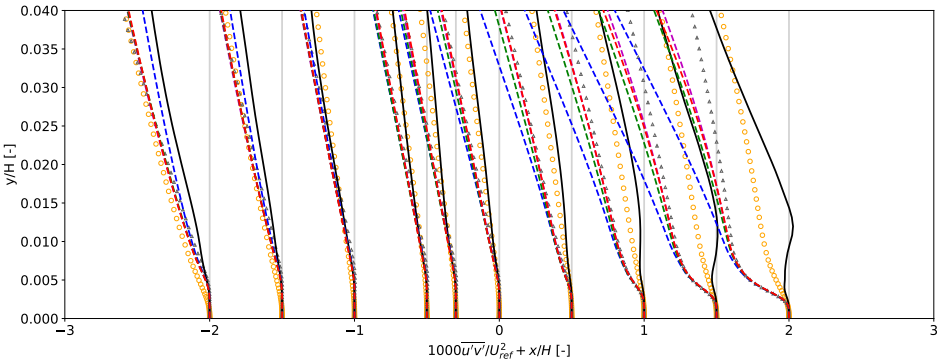
(a) NASA-Hump case.



(b) Periodic-Hill case.



(c) CBFS case.

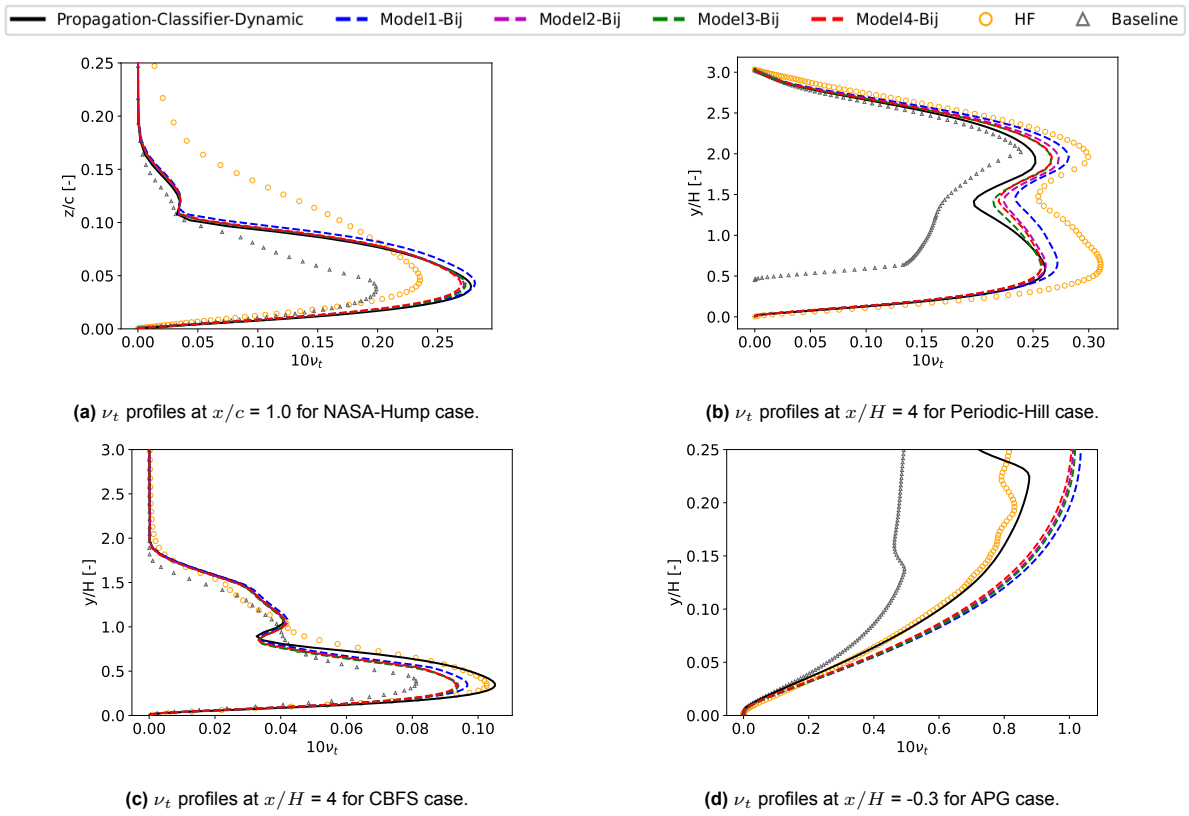


(d) APG case.

Figure 7.4: Comparison of the $\overline{u'v'}$ ($\overline{u'w'}$ in the NASA-Hump case because the domain is designed in the x-z plane rather than the x-y plane) profiles obtained from the hybrid Model Propagation simulations based on the b_{ij}^{Δ} models run in isolation.

Table 7.3: Comparison of the separation and reattachment location predictions obtained from the hybrid Model Propagation simulations based on the b_{ij}^{Δ} models run in isolation.

Cases	NASA-Hump		Periodic-Hill		APG		CBFS	
	Separate x/c [-]	Reattach x/c [-]	Separate x/H [-]	Reattach x/H [-]	Separate x/H [-]	Reattach x/H [-]	Separate x/H [-]	Reattach x/H [-]
Model1-Bij	0.66	1.12	0.30	5.44	-2.27	0.33	0.93	4.72
Model2-Bij	0.66	1.11	0.29	5.35	-2.22	0.29	0.95	4.69
Model3-Bij	0.66	1.11	0.29	5.33	-2.22	0.29	0.95	4.66
Model4-Bij	0.66	1.11	0.29	5.30	-2.17	0.22	0.95	4.70
PCD	0.66	1.09	0.30	5.15	-1.64	0.28	0.89	4.58
HF	0.66	1.06	0.22	4.69	-1.47	0.40	0.82	4.32
Baseline	0.65	1.25	0.26	7.64	-2.51	0.48	0.74	5.85



(a) ν_t profiles at x/c = 1.0 for NASA-Hump case.

(b) ν_t profiles at x/H = 4 for Periodic-Hill case.

(c) ν_t profiles at x/H = 4 for CBFS case.

(d) ν_t profiles at x/H = -0.3 for APG case.

Figure 7.5: Comparison of ν_t profiles at locations near the reattachment region, obtained from the hybrid Model Propagation simulations based on the b_{ij}^{Δ} models run in isolation.

7.1.3. Model Performance Analysis

The results presented in the previous section reveal that the four b_{ij}^{Δ} models show very similar flow prediction performance, despite the different formulations these models have and the different SpaRTA training conditions used to discover them. While these models are not able to entirely reproduce the true b_{ij}^{Δ} corrections, they still demonstrate much better performance compared to the Baseline RANS, as can be seen from the U_x profiles in Figure 7.2. The reason why these models perform better is that they help reduce the underprediction of k and $\overline{u'v'}$ as can be seen from Figures 7.3 and 7.4 respectively. Given that SpaRTA can only perform linear regression and the majority of the models have an R^2 value below 0.8, the obtained results are very satisfactory, generally providing a close prediction to the PCD simulation and the high-fidelity data. It is also important to note that the models perform well even for higher Reynolds number cases like the NASA-Hump case and seem to provide good results even for a change in geometry.

The best match between the models and the PCD and the high-fidelity data profiles occurs near the separation location and the initial stretches of the shear layer. Post-reattachment, the discrepancies become larger, and in the APG case, the models start showing worse predictions than the Baseline RANS. This is very much related to the shear layer cluster where the corrections are being applied. Figures of this cluster for the different cases are available throughout Chapter 6 for the PCD simulations and have not been reshown here, as there are no real differences between those and the shear layer clusters for the hybrid Model Propagation simulations. For the APG case, the shear layer cluster resides too high above the recirculation region and extends too far downstream, so that overcorrections start to occur in the post-reattachment region. For the other cases, this region becomes progressively thinner downstream of the blunt body geometries so that corrections are only being applied in a very small region.

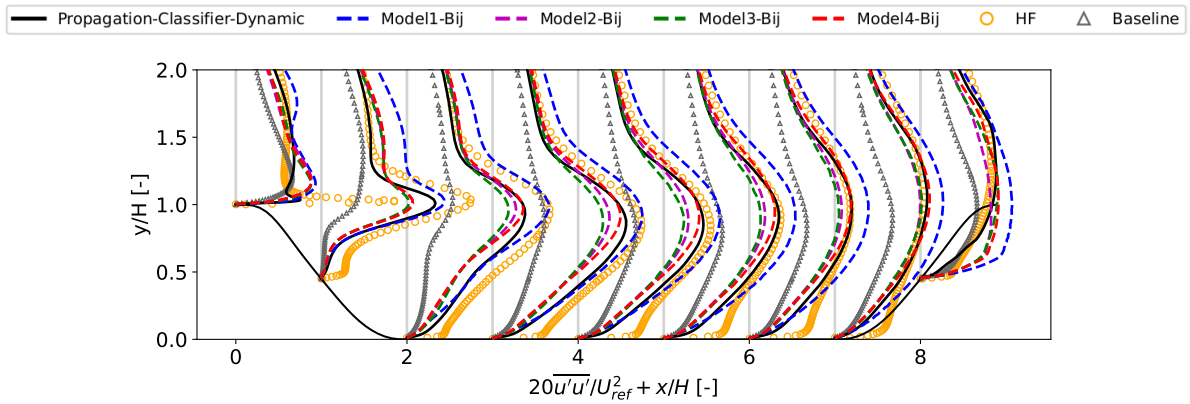
Model1-Bij performs the worst in predicting the reattachment location for all cases except APG. This finding is particularly intriguing because the model was specifically trained on the NASA-Hump case, leading to the expectation that it would outperform all other models for this case. However, this demonstrates that while a model may be trained on a single test case and achieve a high R^2 value, it does not guarantee its a-posteriori performance in a CFD solver. Although this model performs better at capturing the reattachment location for the APG case, it also predicts the separation location much earlier than the other models. Overall, Model1-Bij shows the poorest performance. Therefore, it is important to understand the reasons behind this outcome.

The U_x velocity profiles predicted by this model reveal that the flow velocity, particularly in the shear layer region, is slightly lower compared to the other models. For the APG case, this lower velocity is advantageous as it leads to an earlier flow reattachment location, closer to the high-fidelity data and PCD predictions. On the other hand, the k profiles for this model appear much closer to the high-fidelity data and PCD predictions than those of the other models in the initial stretch of the shear layer. However, in the later stretches of the shear layer, the model tends to overpredict k, which leads in the NASA-Hump case to overpredictions in ν_t, as can be seen from Figure 7.5a. This overprediction is also reflected in the $\overline{u'v'}$ profiles, where the model overestimates the shear stresses in this region. For the APG case, this results in a better prediction of the reattachment location because the additional shear stress shortens the length of the recirculation region. Despite the overpredictions in both k and $\overline{u'v'}$, the other models still show a delayed reattachment location, indicating a longer recirculation region.

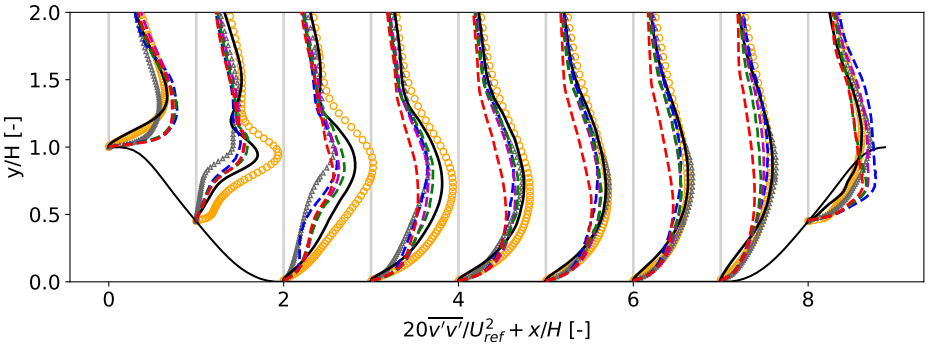
To investigate this further, the normal stress components of the Reynolds-stress tensor $\overline{u'u'}$ and $\overline{v'v'}$ for the Periodic-Hill case have been plotted in Figure 7.6. The Periodic-Hill case was chosen because Model1-Bij shows the poorest performance for it. These plots reveal that the simulation based on Model1-Bij overpredicts the streamwise stresses $\overline{u'u'}$ in the mean-flow region and in the later stretches of the shear layer, while underpredicting the transverse stresses $\overline{v'v'}$ in the initial stretch of the shear layer and the recirculation region. This indicates that Model1-Bij primarily fails compared to the others due to its inability to accurately predict the stress distribution among shear, streamwise, and transverse components. The overprediction in streamwise stress decelerates the flow, which is reflected in the U_x velocity profiles, consequently affecting the pressure distribution and resulting in a delayed reattachment location.

To understand the difference in stress predictions, one can consider the structural form of Model1-Bij. Term 2 of this model is based on the basis tensor $T_{ij}^{(3)}$, which, unlike the basis tensor $T_{ij}^{(2)}$, has only non-zero diagonal components, meaning it only contributes to the normal stresses of the Reynolds-stress tensor. Additionally, this term has the highest coefficient value (36.73) multiplying its rdiv() function, which explains the overprediction in the streamwise normal stresses. Due to its poor performance, this model is excluded from further analysis.

The next two models to consider are Model2-Bij and Model4-Bij. Their formulations show that their first terms are almost identical. However, Model4-Bij includes an additional term based on the rlog() function of the turbulence based Reynolds number Re_k. As discussed earlier, Re_k does not exceed 0.6 in the shear layer region, making the contribution of this additional term small. Nevertheless, the contribution is significant enough to help the model predict slightly better separation and reattachment locations for the APG case and Periodic-Hill case, respectively. Since the difference between these two models is not substantial, the first term based on the rdiv() function of the I_1 invariant is considered to be the more significant term in the model formulations.



(a) $\overline{u'u'}$ profiles.



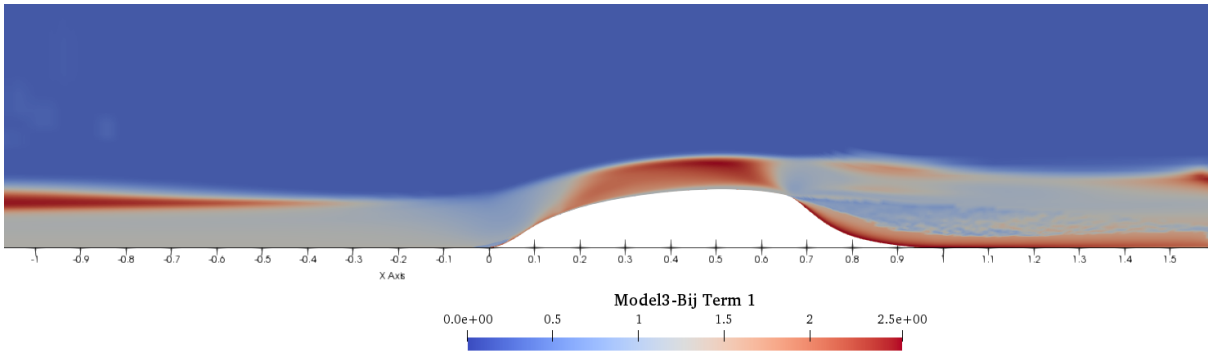
(b) $\overline{v'v'}$ profiles.

Figure 7.6: Comparison of the normal Reynolds-stress tensor components profiles obtained from the hybrid Model Propagation simulations based on the b_{ij}^{Δ} models run in isolation for the Periodic-Hill case.

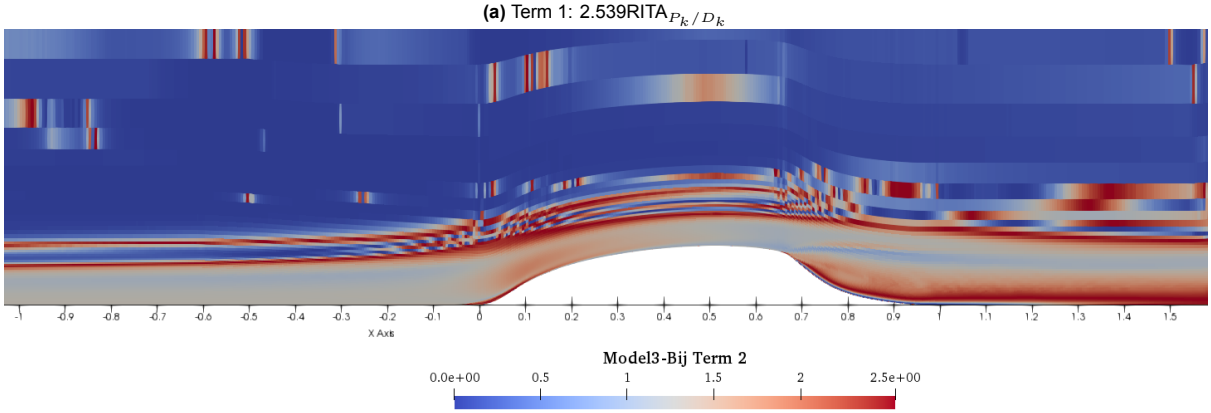
The largest discrepancy between the models can be traced back to the normal Reynolds-stress components $\overline{u'u'}$ and $\overline{v'v'}$, which are displayed for the Periodic Hill case in Figure 7.6. From the other flow profiles (U_x, k, and $\overline{u'v'}$), it is almost impossible to differentiate the two models from each other. Model4-Bij is better at predicting the streamwise $\overline{u'u'}$ stresses, while Model2-Bij is better at predicting the transverse $\overline{v'v'}$ stresses. This explains why Model4-Bij shows an improvement over Model2-Bij in terms of predicting an earlier reattachment location. However, since this model depends on the Re_k feature which is Reynolds number dependent due to its reliance on molecular viscosity, and the slight improvements that Model4-Bij shows over Model2-Bij are not significant enough, Model4-Bij is avoided. It is better to have a Reynolds number independent model that slightly underperforms than one that only performs marginally better but is Reynolds number dependent.

The final model to consider is Model3-Bij, which was obtained after removing the APG case from the regression run. With Model1-Bij and Model4-Bij excluded from the final model selection, Model3-Bij is compared to Model2-Bij. From the separation and reattachment locations listed in Table 7.3, Model3-Bij shows the best overall performance. This is also reflected in the k, U_x, and $\overline{u'v'}$ plots, which show a very slight improvement over Model2-Bij when examined closely. Similar to the other models, it also overpredicts the shear stress after the initial stretch of the shear layer. In terms of the normal stress components, it underpredicts both the streamwise and transverse stresses. Nevertheless, it is considered the best-performing model and is selected for further analysis in Section 7.4, where the best R and b_{ij}^{Δ} models are combined in a simulation.

Now that the best-performing model for b_{ij}^{Δ} has been identified, it is interesting to consider how this model is correcting for the missing physics that the Boussinesq approximation of the Reynolds-stress tensor is not able to reproduce. Model3-Bij has two terms, one based on the RITA$_{P_k/D_k}$ ratio and the other based on the rdiv() function of the I_2 invariant. A plot of these two quantities is displayed in Figure 7.7 for the NASA-Hump case.



(a) Term 1: 2.539RITA$_{P_k/D_k}$



(b) Term 2: -5.092rdiv$\left(\frac{I_2}{0.0125}\right)$

Figure 7.7: Distribution of the features and invariants multiplying the basis tensor $T_{ij}^{(2)}$ in Model3-Bij for the NASA-Hump case.

A clear outline of the shear layer cluster in the distribution of the RITA$_{P_k/D_k}$ term is visible in Figure 7.7a. The values in this region are close to 1, which indicates that RITA$_{P_k/D_k}$ has values below 0.5 in this region. This makes sense as the production of k dominates its destruction in the shear layer. The corrections provided by the I_2 invariant-based term are much higher in this region (generally above 1.5). The I_2 invariant is computed from the rotation rate tensor, which has higher values in this region and the recirculation region compared to the rest of the flow domain. Since both terms are based on the $T_{ij}^{(2)}$ tensor basis, this indicates that the majority of the corrections to the turbulence anisotropy are coming from term 2 based on the I_2 invariant.

Both terms show a relatively uneven corrective pattern, characterized by fluctuations in corrections from one domain cell to the next, resulting in distributions that appear almost cloudy. Unlike the Boussinesq approximation, which fails to fully capture the turbulence anisotropy, these corrections are designed to account for more of the underlying physics, addressing the missing anisotropic effects and capturing the complex interactions within the turbulent flow. This ability to capture the missing physics is achieved by modeling b_{ij}^{Δ} with the observed uneven, yet more accurate, distributions of the corrective terms.

7.2. R Models

7.2.1. Overview of Discovered Model Forms

In total, four R models have been selected for further analysis. Table 7.4 provides an overview of their R^2 values, the number of terms they consist of, and the SpaRTA training conditions used to obtain them. All models have very high R^2 values, which are much higher than the R^2 values obtained for the b_{ij}^{Δ} models. This difference is because regressing models for the R scalar field is a much simpler problem than fitting the b_{ij}^{Δ} 6 component tensor field.

For the b_{ij}^{Δ} models, Model4-Bij was selected from a SpaRTA regression session that involved training on all cases with all features and invariants available. In contrast, for the R models, these same conditions resulted in a model dependent on the non-invariant turbulence intensity feature, which did not show

superior performance compared to the other models. Consequently, this model has not been included for further analysis. Instead, a different regression condition was used to obtain Model4-R listed below.

An interesting observation can be made by comparing the formulations of the first three R models in the color box below: they all depend on the feature representing the ratio of total to normal Reynolds stresses, $\tau_{ij,ratio}$. This feature tends to dominate over the other features when included in the regression. A Mutual Information analysis also identified this feature to be one of the most relevant for the regression of R. However, in terms of model performance, it is informative to compare models based on this feature to those which are not. Therefore, to obtain Model4-R, the regression was run excluding this feature from the dataset. As expected, a completely different model emerged, one with a slightly lower R^2 value and an extra term. The exact model formulation will be further clarified below.

Table 7.4: General information regarding the obtained R models.

Model I.D	R^2 Value	Nr. of Terms	SpaRTA Training Conditions
Model1-R	0.96	2	NASA-Hump case: selected features and invariants.
Model2-R	0.96	3	All cases: selected features and invariants.
Model3-R	0.96	3	All cases excluding APG: selected features and invariants.
Model4-R	0.95	4	All cases excluding APG: $\tau_{ij,ratio}$ feature removed.

The formulations of the R models is given below, together with Table 7.5 which provides further insights into the different features and functions that appear in the model equations.

R Model Formulations

Model1-R: $$R = \underbrace{0.033\boldsymbol{\epsilon}}_{\text{Term 1}} + \underbrace{1.074\tau_{ij,ratio}\mathbf{G_3}}_{\text{Term 2}}$$

Model2-R: $$R = \underbrace{0.020\boldsymbol{\epsilon}}_{\text{Term 1}} + \underbrace{0.047\tau_{ij,ratio}\boldsymbol{\epsilon}}_{\text{Term 2}} + \underbrace{0.589\sqrt{\tau_{ij,ratio}}\mathbf{G_1}}_{\text{Term 3}}$$

Model3-R: $$R = \underbrace{0.002\boldsymbol{\epsilon}}_{\text{Term 1}} + \underbrace{0.807\sqrt{\tau_{ij,ratio}}\mathbf{G_1}}_{\text{Term 2}} + \underbrace{0.038\tanh\left(\frac{PS}{0.0007}\right)\boldsymbol{\epsilon}}_{\text{Term 3}}$$

Model4-R: $$R = \underbrace{0.008\boldsymbol{\epsilon}}_{\text{Term 1}} \underbrace{- 32290\text{rdiv}\left(\frac{I_3}{2.580\times10^{-5}}\right)\mathbf{G_9}}_{\text{Term 2}} \underbrace{- 0.089\text{rdiv}\left(\frac{I_2}{0.011}\right)\boldsymbol{\epsilon}}_{\text{Term 3}} + \underbrace{0.006\text{RITA}^{0.5}_{C_k/D_{f,k}}\boldsymbol{\epsilon}}_{\text{Term 4}}$$

The regression appears to have identified the ϵ scalar basis as the most important one for reconstructing the R corrections, which matches the results obtained by Steiner et al. in their study [33]. The tendency of the regression to find models that rely on this basis is quite important because it shows that the modeling uncertainties in the $k-\omega$ SST model equations are different in nature than the ones for the turbulence anisotropy, which are accounted for by the b_{ij}^{Δ} models. The R models mainly rely on the features identified in Chapter 4 Section 4.4. Therefore, the selected list of features for the regression must match the physics that R is trying to account for, for the regressed R models to be able to properly correct for the model-form errors.

As discussed previously, the first 3 models are based on the $\tau_{ij,ratio}$ feature, which is computed from the ratio of total to normal Reynolds stresses. In the shear layer, this feature takes values of 0.5 and below, as can be seen from its distribution in Figure A.12 in Appendix Section A.1. Other features, such as the ratio of pressure normal stresses to shear stresses (PS) and the RITA ratio of convection to diffusion ($RITA_{C_k/D_{f,k}}$) also appear in these models. This demonstrates the utility of using features specifically chosen to reflect the local turbulence characteristics in the regression. The R models use similar functions to the b_{ij}^{Δ} models for their features, with only Model3-R based on an additional tanh() function. As shown in Figure 7.8, $\tanh(x)$ asymptotically approaches 1 even for large x, so the coefficient in front of ϵ for this term will not exceed 0.038.

Table 7.5: Overview of the different variables used in the R model formulations.

Symbol	Meaning	Additional Notes
ϵ	Dissipation rate of k	Additional basis for R models.
G_1	$G_3 = 2kT_{ij}^{(3)} \frac{\partial U_i}{\partial x_j}$	Based on Pope Tensor Basis
G_3	$G_1 = 2kT_{ij}^{(1)} \frac{\partial U_i}{\partial x_j}$	Based on Pope Tensor Basis
G_9	$G_9 = 2kT_{ij}^{(9)} \frac{\partial U_i}{\partial x_j}$	Based on Pope Tensor Basis
$\tau_{ij,ratio}$	Ratio of total to normal Reynolds stresses	Feature F9 - equation 4.30
PS	Ratio of pressure normal stresses to shear stresses	Feature F8 - equation 4.29
$RITA_{C_k/Df_k}$	RITA ratio of convection to diffusion of k	Feature F13 - equation 4.34
I_2	Invariant: $I_2 = \Omega_{ij}\Omega_{ij}$	Ω_{ij} is the mean rotation rate.
I_3	Invariant: $I_3 = S_{ij}S_{ij}S_{ij}$	S_{ij} is the mean strain rate.
rdiv(x)	Function of the form: rdiv$(x) = \frac{x}{(1+x^2)}$	Plot in Figure 7.1
tanh(x)	Hyperbolic tangent of x	Plot in Figure 7.8

Terms 2 and 3 of Model4-R are based on the rdiv() functions of the invariants I_2 and I_3. Despite the minus signs before both of these terms, they still add to the R corrections. This is because computing these invariants leads to negative values that cancel out the negative signs in front of the terms. It is important to note the very large coefficient in front of term 2, meaning that this term essentially dominates the corrections. Consequently, this model provides the largest corrections for R out of all four models. The effect this has on the flow behavior will be further clarified in the upcoming section.

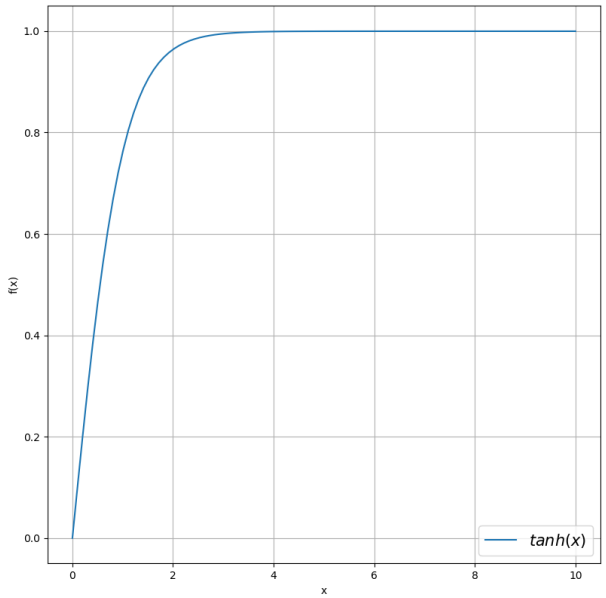
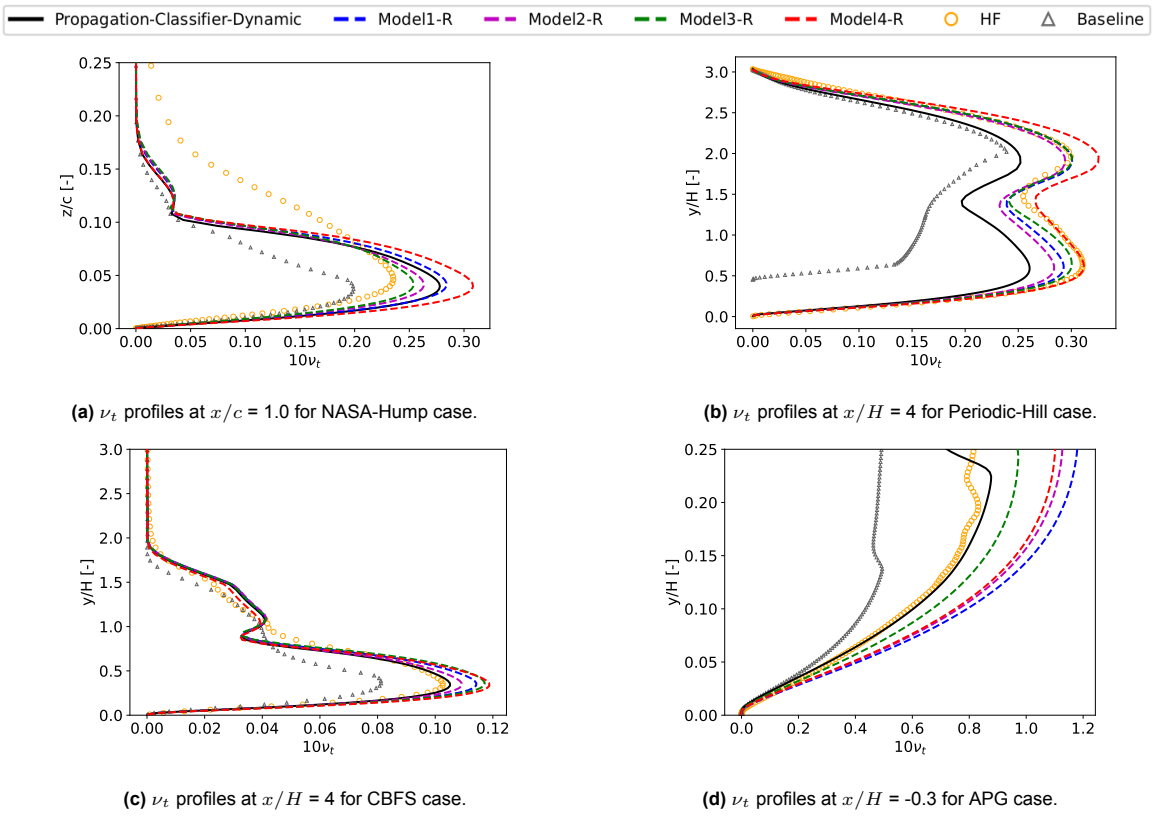


Figure 7.8: General shape of the tanh(x) function.

7.2.2. Modelling Results

This section presents several figures and data obtained from hybrid Model Propagation simulations, where R was modeled following the formulations discussed in the previous section, and b_{ij}^{Δ} was read from its Frozen correction field. This setup allows for the analysis of R models in isolation. All corrections are applied only in the shear layer cluster identified by the RITA/TI classifier, which is dynamically updated during the simulation. For comparison purposes, the results of the Baseline RANS, the Propagation-Classifier-Dynamic (PCD) simulation, and the high-fidelity data (HF) (LES or DNS depending on the case) are also displayed. Profiles of the axial velocity U_x, the turbulent kinetic energy k, and the shear stress component of the Reynolds-stress tensor $\overline{u'v'}$ are displayed in Figures 7.10, 7.11, and 7.12 respectively.

The separation and reattachment locations predicted by each Model Propagation simulation are listed in Table 7.6. As previously discussed, some cases show both primary and secondary recirculation regions, where one region is more significant in terms of size and physical significance than the other. Therefore, only the separation and reattachment locations of the most relevant separation regions are listed in this table. Finally, Figure 7.9 presents a comparison of the eddy viscosity ν_t profiles.



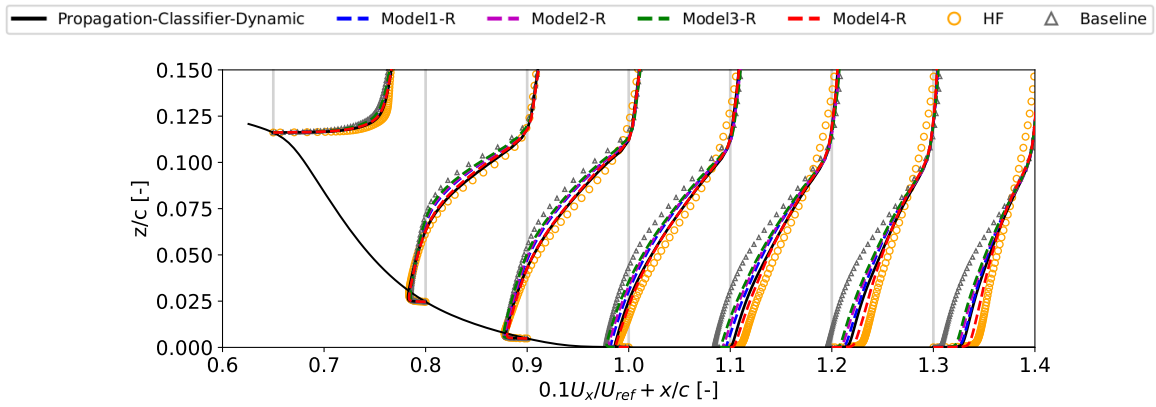
(a) ν_t profiles at x/c = 1.0 for NASA-Hump case.

(b) ν_t profiles at x/H = 4 for Periodic-Hill case.

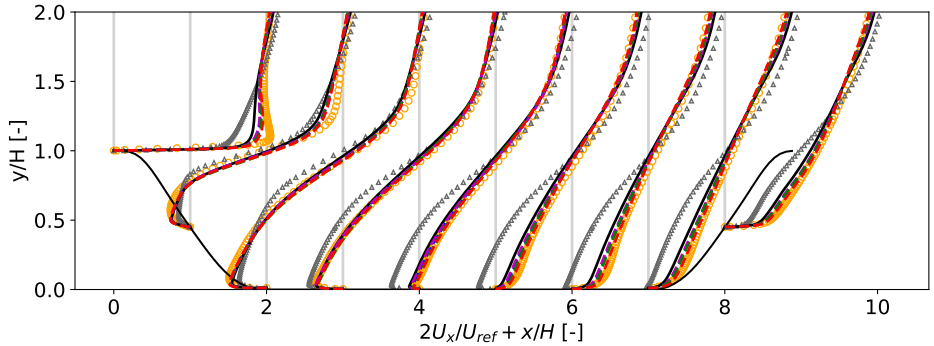
(c) ν_t profiles at x/H = 4 for CBFS case.

(d) ν_t profiles at x/H = -0.3 for APG case.

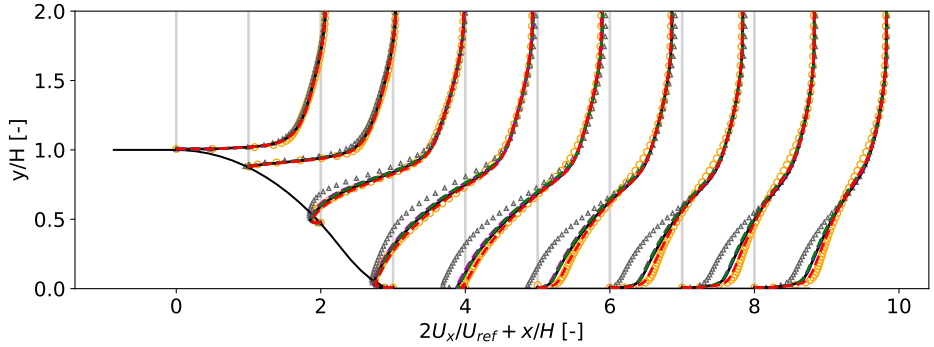
Figure 7.9: Comparison of ν_t profiles at locations near the reattachment region, obtained from the hybrid Model Propagation simulations based on the R models run in isolation.



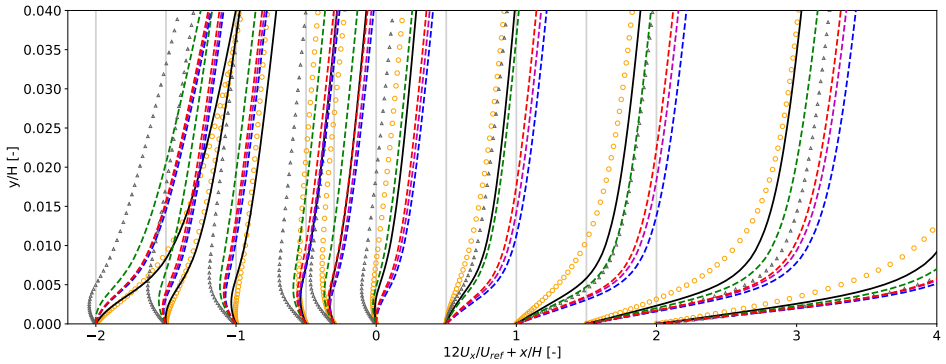
(a) NASA-Hump case.



(b) Periodic-Hill case.

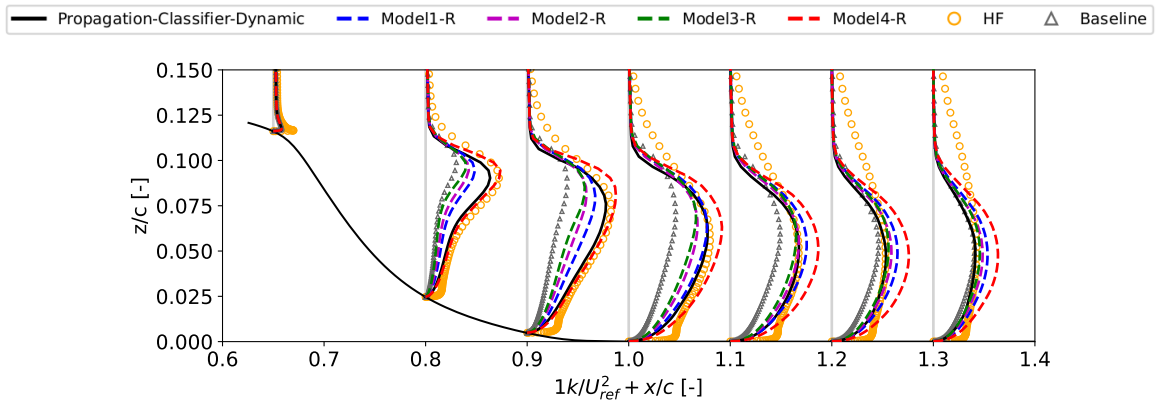


(c) CBFS case.

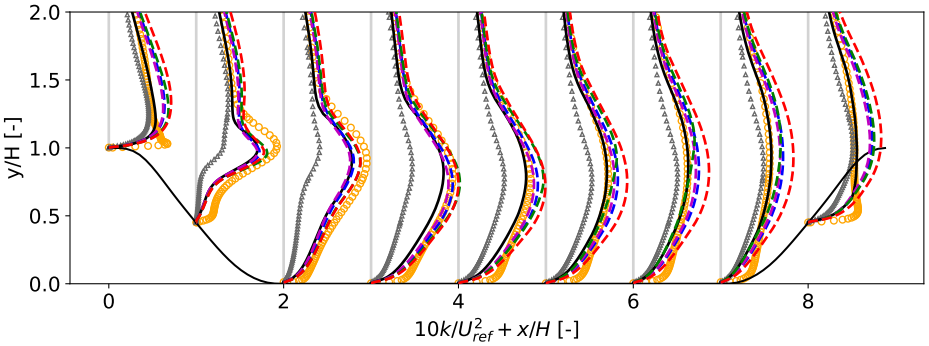


(d) APG case.

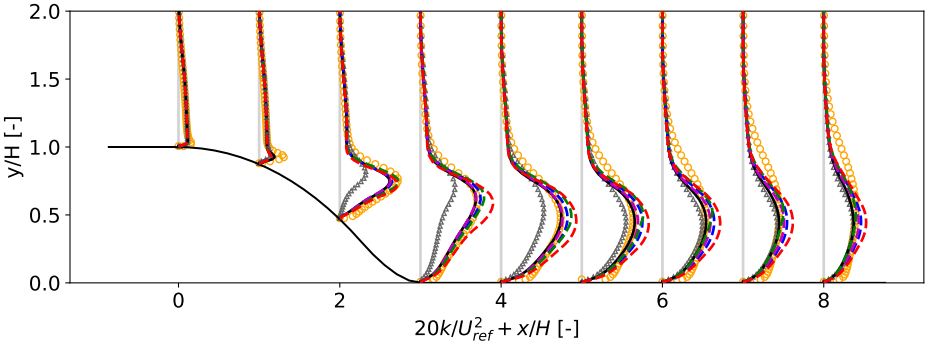
Figure 7.10: Comparison of U_x profiles obtained from the hybrid Model Propagation simulations based on the R models run in isolation.



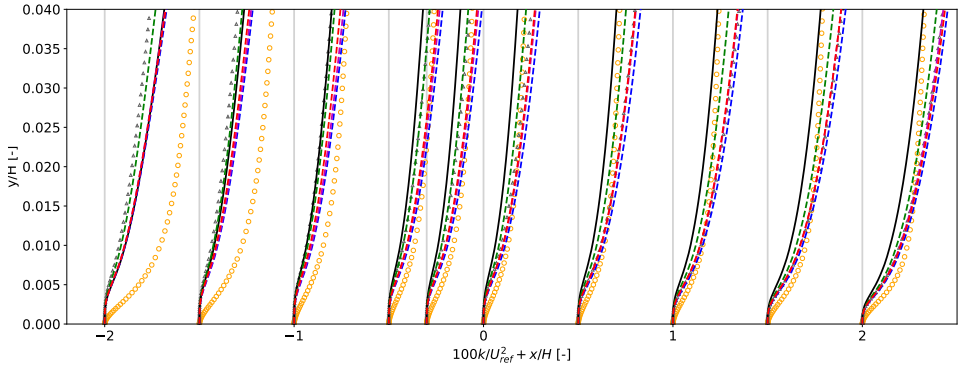
(a) NASA-Hump case.



(b) Periodic-Hill case.

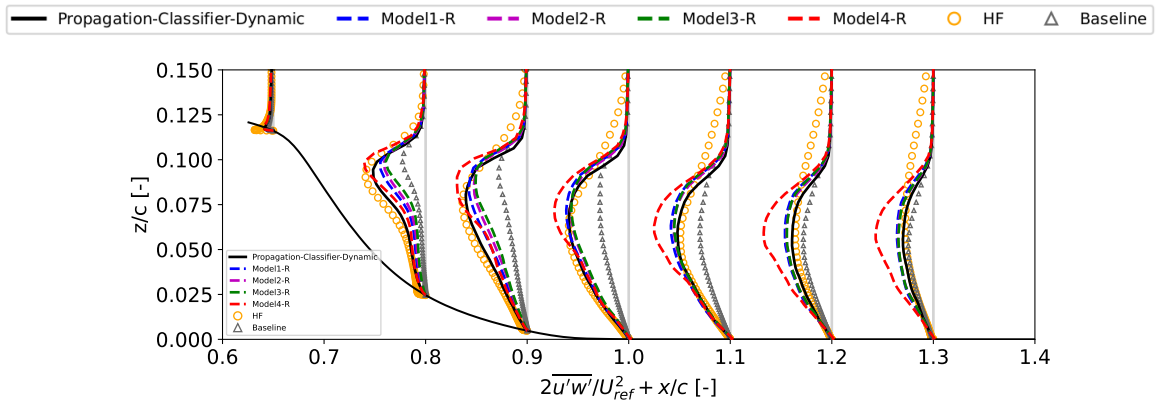


(c) CBFS case.

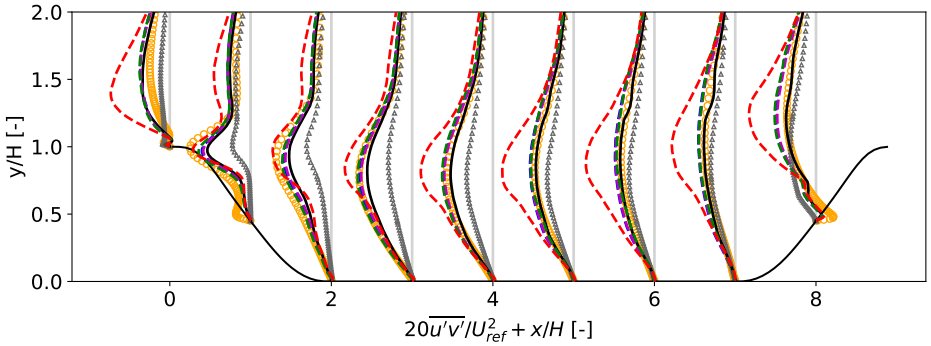


(d) k profiles APG.

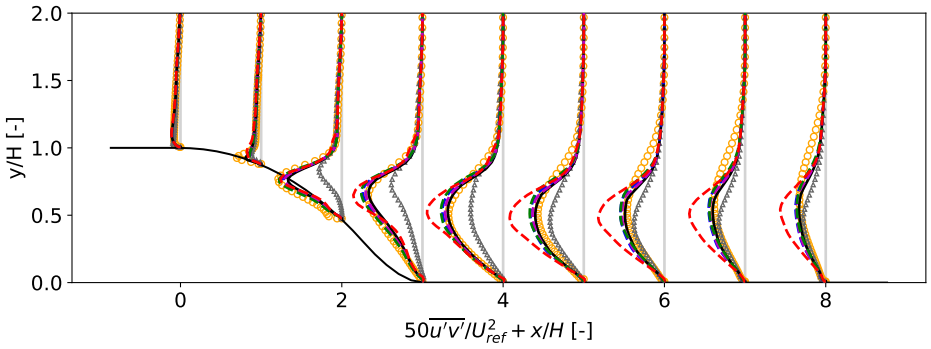
Figure 7.11: Comparison of k profiles obtained from the hybrid Model Propagation simulations based on the R models run in isolation.



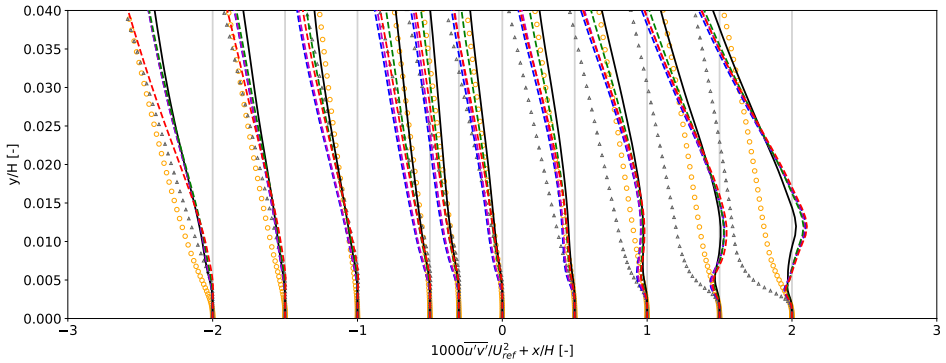
(a) NASA-Hump case.



(b) Periodic-Hill case.



(c) CBFS case.



(d) APG case.

Figure 7.12: Comparison of $\overline{u'v'}$ ($\overline{u'w'}$ in the NASA-Hump case because the domain is designed in the x-z plane rather than the x-y plane) profiles obtained from the hybrid Model Propagation simulations based on the R models run in isolation.

Table 7.6: Comparison of the separation and reattachment location predictions obtained from the hybrid Model Propagation simulations based on the R models run in isolation.

Cases	NASA-Hump		Periodic-Hill		APG		CBFS	
	Separate x/c [-]	Reattach x/c [-]	Separate x/H [-]	Reattach x/H [-]	Separate x/H [-]	Reattach x/H [-]	Separate x/H [-]	Reattach x/H [-]
Model1-R	0.66	1.14	0.27	4.83	-1.84	-0.003	0.82	4.64
Model2-R	0.66	1.15	0.27	4.93	-1.87	0.05	0.82	4.70
Model3-R	0.66	1.17	0.28	4.69	-2.12	0.30	0.82	4.60
Model4-R	0.66	1.09	0.28	4.64	-1.87	0.11	0.88	4.29
PCD	0.66	1.09	0.30	5.15	-1.64	0.28	0.89	4.58
HF	0.66	1.06	0.22	4.69	-1.47	0.40	0.82	4.32
Baseline	0.65	1.25	0.26	7.64	-2.51	0.48	0.74	5.85

7.2.3. Model Performance Analysis

The results presented in the previous section reveal that although the four R models do not completely reproduce the R corrections, they still perform significantly better compared to the Baseline $k-\omega$ SST simulation. This improvement can be observed from the U_x profiles shown in Figure 7.10 and the predictions of separation and reattachment locations in Table 7.6. The models enhance the flow predictions by reducing the underprediction of k and $\overline{u'v'}$, as shown in Figures 7.11 and 7.12, respectively.

Model4-R is by far the best-performing model in terms of predicting the flow velocity profiles and the reattachment locations. Its superior performance can be understood from the plots of k and $\overline{u'v'}$. Unlike the other three models, Model4-R can almost exactly predict the profiles of these two quantities in the initial stretch of the shear layer. However, this accuracy comes at the cost of overpredicting these profiles in the later stretches of the shear layer. These results are intriguing as they highlight the importance of predicting the correct amount of turbulent kinetic energy and shear stress near the separation region to obtain accurate reattachment locations.

The later overestimations in these quantities, although quite large, do not significantly impact the flow reattachment location. Only in the CBFS case does this result in a slightly earlier reattachment location. The reason for Model4-R's overprediction compared to the other models is the large coefficient in front of its term 2, as discussed in Section 7.2.1. It is important to note that the predictions of separation locations are similar to those obtained from the other R models. Near the separation location, corrections are applied in only a very thin region for all models due to the shape of the RITA/TI classifier. Therefore, the differences in separation predictions among the different models are not substantial. Furthermore, the exact separation location is also influenced by the incoming boundary layer upstream.

The other R models have significantly worse predictions of velocity and reattachment locations compared to Model4-R, as these models still underpredict k and $\overline{u'v'}$ quite significantly in the initial stretches of the shear layer. In the later stretches of the shear layer, these models also begin to overpredict these two quantities, but not nearly as much as Model4-R. Overall, these observations reveal that using the SpaRTA regression method with the given list of features and invariants, along with the modeling ansatz created for R and the shear layer cluster, it does not seem possible to obtain a model that accurately predicts the production of k and the Reynolds-stress tensor in both the initial and subsequent stretches of the shear layer. For industrial applications, such as Formula 1 car design, this might not necessarily be a problem since the most critical factor is the ability to predict the correct drag and downforce. However, from the perspective of the turbulence modeling community, which aims to find model corrections that represent the true missing physics from turbulence models, having a model that achieves good results while severely overpredicting the model quantities in certain regions of the domain is not ideal. In any case, Model4-R is selected for further analysis in Section 7.4 where the b_{ij}^{Δ} and R models are coupled together.

A model from the first three R Models is also selected to gauge the overall achievable improvements when using an R model that has the advantage of not overpredicting k and $\overline{u'v'}$ in the later stretches of the shear layer, albeit at the cost of underpredicting these quantities in the initial stretches of the shear layer. Among these first three R Models, Model3-R shows the best performance and is thus chosen for further analysis. It is noteworthy that, similar to the b_{ij}^{Δ} models, the best performing R models were

also identified when the APG case was excluded from the training dataset. It is therefore interesting to further investigate the formulations of the two selected models — Model3-R and Model4-R. Figures 7.14 and 7.15 display the computed values of each term in the equations of these two models. Term 1 of Model4-R, equal to 0.008ϵ, has been omitted as its distribution is essentially identical to Term 1 of Model3-R, which is equal to 0.002ϵ. To serve as a reference, the plot of the production term of the k equation (P_k), which incorporates the b_{ij}^{Δ} frozen corrections in the shear layer, is also shown in Figure 7.13.

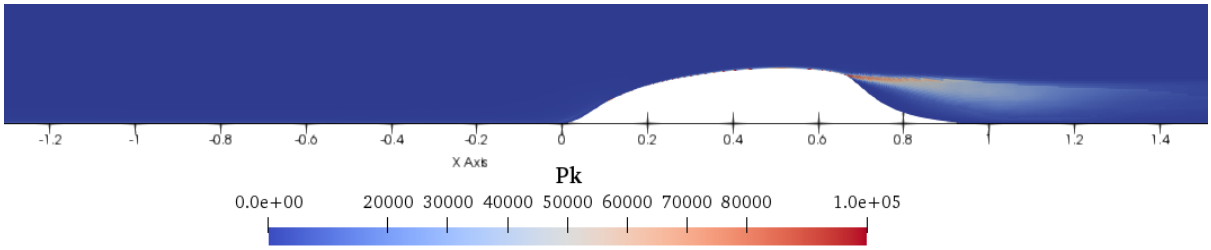
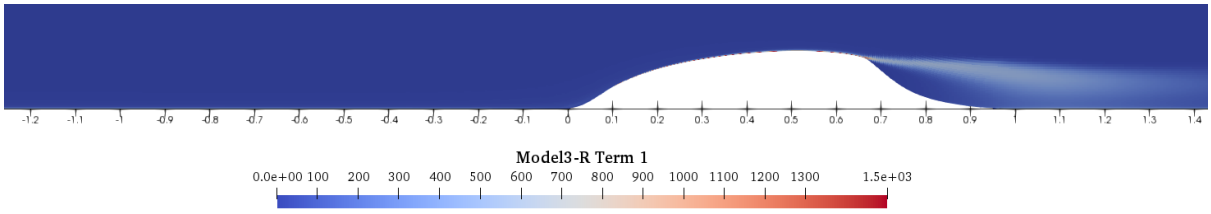


Figure 7.13: Distribution of the production of turbulent kinetic energy across the NASA-Hump domain.

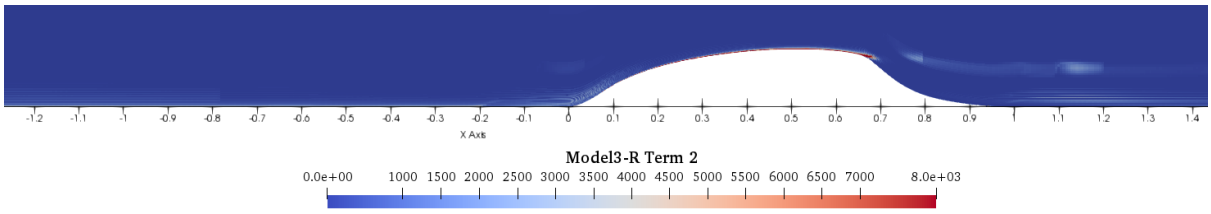
Comparing the relative magnitudes of the terms in Model3-R to those of Model4-R and the P_k plot in Figure 7.13, it becomes evident why Model3-R underpredicts the production of k in the initial stretches of the shear layer compared to Model4-R, and why Model4-R starts to significantly overpredict the production of k in the later stretches of the shear layer. The corrections contributed by each term in Model3-R are quite small in magnitude and are all positive, which helps to slightly increase the production of k in the shear layer (thereby adding to the P_k term). It appears that the contribution of Term 3 in Model3-R is essentially insignificant, as its values range from 30 to 50, while P_k has a magnitude of around 45,000 to 50,000 in the shear layer.

It is interesting to note that term 2 of Model3-R, based on the $\tau_{ij,ratio}$ and the $T_{ij}^{(1)}$ Pope tensor basis, also does not contribute much to the overall corrections apart from near the separation location, where it shows very high values. However, these corrections at this point are crucial as they significantly boost the overall production of k right in the initial stretch of the shear layer, leading to better reattachment predictions. Term 1 of Model3-R reveals why the ϵ scalar basis is important during R SpaRTA model regression. Since there is always a balance between the production and dissipation of k, the magnitude of ϵ is comparable to that of P_k. In the shear layer, ϵ shows higher values than in the rest of the domain, as the production of k is most significant in this region. Tuning the coefficient in front of ϵ is therefore beneficial in locally adjusting the amount of corrections required to obtain accurate predictions of the production of k. Overall, the distributions of the terms discovered by the regression for this R model suggest that the missing physics this model aims to account for is primarily the initial burst in the production of k right at the point of separation and in the initial stretches of the shear layer. This becomes even clearer when comparing the terms of Model4-R, which all show very high values in this specific region and generally tend towards smaller values in the later stretches of the shear layer. As mentioned above, the magnitude of the corrections of these terms is much higher than those of Model3-R. Furthermore, the distribution of these terms is also quite different, appearing much more uneven and dispersed, similar to the distribution of the terms in the b_{ij}^{Δ} models discussed in Section 7.1.3.

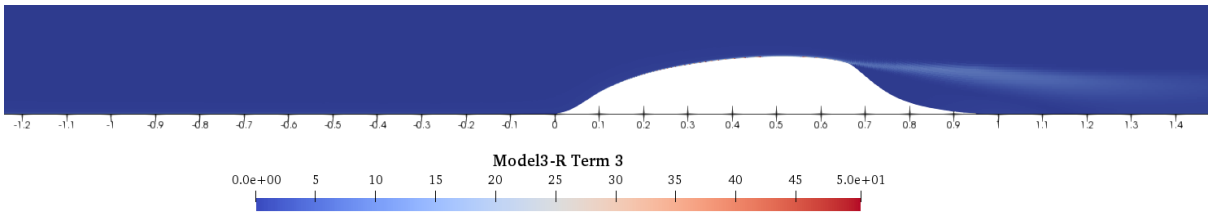
This is not unexpected; as seen in Chapter 6, the R corrections obtained from the Frozen simulations are also very dispersed and uneven. Therefore, during the regression, since these frozen corrections are used as the target data, the regression also tries to find model terms that can capture this unevenness in the R corrections, which is created by local uncertainties in the model equations of the $k-\omega$ SST model. While the uneven distributions of the b_{ij}^{Δ} model terms aimed to address missing turbulence anisotropy effects, the unevenness observed in the R model terms does not address the same fundamental missing physics. Instead, it is more likely that this unevenness results from the models attempting to correct for uncertainties arising from the calibration of the $k-\omega$ SST model's F_1 and F_2 blending functions, along with the underlying assumptions used to model the ω and k equations. Nevertheless, the R models demonstrate that the primary missing physics that the $k-\omega$ SST turbulence model fails to account for is the production of turbulent kinetic energy near the separation location and in the initial stretches of the shear layer.



(a) Term 1: 0.002ϵ

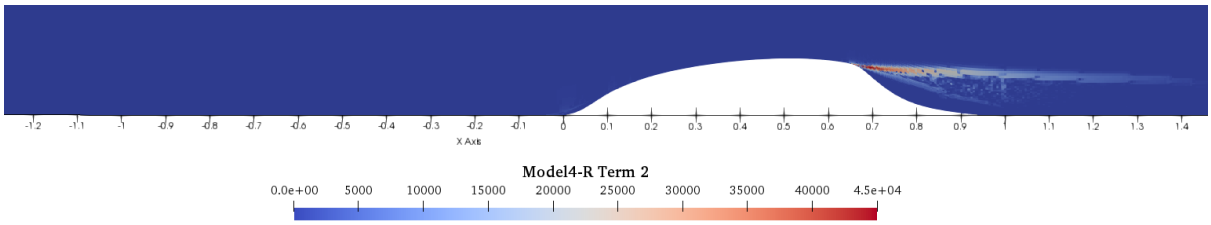


(b) Term 2: $0.807\sqrt{\tau_{ij,ratio}}\mathbf{G_1}$

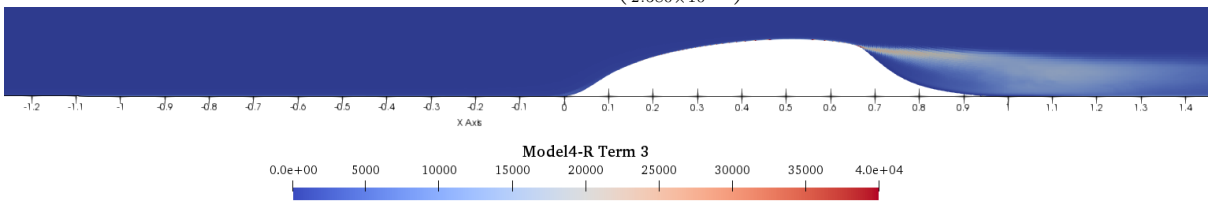


(c) Term 3: $0.038\tanh\left(\frac{PS}{0.0007}\right)\epsilon$

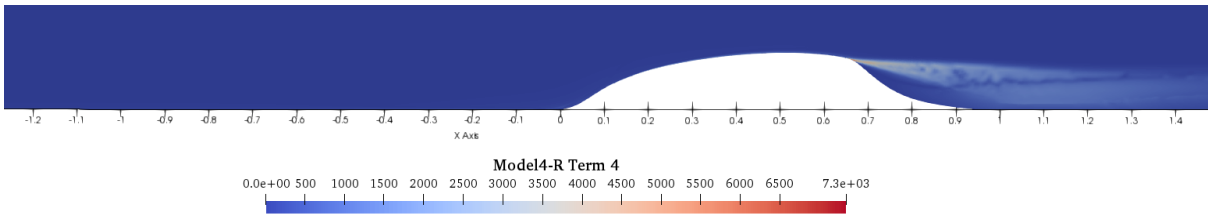
Figure 7.14: Distribution of the terms of Model3-R.



(a) Term 2: $32290\text{rdiv}\left(\frac{I_3}{2.580\times10^{-5}}\right)\mathbf{G_9}$



(b) Term 3: $0.089\text{rdiv}\left(\frac{I_2}{0.011}\right)\epsilon$



(c) Term 4: $0.006\text{RITA}^{0.5}_{C_k/D_{f,k}}\epsilon$

Figure 7.15: Distribution of the terms of Model4-R.

7.3. Comparison with Literature

Before combining and testing the b_{ij}^{Δ} and R models discussed above, it is beneficial to compare them with models from other studies that employed similar modeling approaches. Selecting a couple of these models to plot for comparison in Section 7.4 can help determine whether the models obtained in this study, specifically tailored to shear layer physics, offer any advantage in predicting the behavior of 2D-separated flows. Several studies have been chosen for this comparison.

First, it is essential to address the models listed in the original study by Schmelzer et al. [19], which established the SpaRTA methodology. This study identified three models, two of which exclusively modeled R while setting b_{ij}^{Δ} to zero. The R models are one-term models of the form $C_0 T_{ij}^{(1)}$, where C_0 is a constant coefficient. The single discovered model for b_{ij}^{Δ} has the following formulation: $C_0 T_{ij}^{(1)} + C_1 T_{ij}^{(2)}$, where C_0 and C_1 are constant coefficients.

Unlike the models discovered in this study, the models from Schmelzer et al. use only constant coefficient values in front of the tensor basis. Introducing non-constant coefficient values, as achieved in this study by incorporating functions of features and invariants, adds an extra layer of flexibility to the modeling approach, making it easier to capture changes in turbulence behavior. Additionally, the R models discovered in this study primarily depend on the ϵ scalar basis rather than the Pope tensor basis. This suggests that the physics the R models are accounting for is different from that of the b_{ij}^{Δ} models. In contrast, the models discovered by Schmelzer et al. rely exclusively on the Pope tensor basis, making this distinction in accounted physics less apparent.

In this study, discovering models for b_{ij}^{Δ} posed no significant challenges during the SpaRTA regression. However, in the study by Schmelzer et al., only one b_{ij}^{Δ} model was discovered. The key difference lies in the data used for regression: their study employed full-field data, whereas this study only uses data from the shear layer cluster. Restricting the regression to this subset of data significantly reduces the complexity of the regression problem, making it much easier to find models. This claim is further supported by findings in Kaj Hoefnagel's Master thesis [8], where similar difficulties were encountered in regressing models for b_{ij}^{Δ}, further supporting the effectiveness of restricting the regression to classifier region data for helping the model discovery process for b_{ij}^{Δ}.

When the models obtained from Schmelzer et al. were tested together with the models obtained in this study, the models that corrected solely for R successfully converged for all simulations. However, their model, which also included a correction for b_{ij}^{Δ}, did not converge for the NASA-Hump and APG cases. This aligns with the findings of Kaj Hoefnagel's study, who failed to identify any b_{ij}^{Δ} model that allowed the simulations to converge for the NASA-Hump case. The NASA-Hump and APG cases have much higher Reynolds numbers than the Periodic-Hill and CBFS cases and provide a long region for full turbulent boundary layer development. It is likely that applying corrections in the full field, as done in these two studies, affects the boundary layer development and causes the simulations to diverge. The fact that none of the models discovered in this study caused any convergence issues demonstrates the utility of applying corrections only in areas where the RANS model struggles, specifically the shear layer in 2D-separated flow cases.

The R model which showed the biggest promise in Kaj Hoefnagel's study is selected for comparison purposes in Section 7.4. This model (from here on referred to as Model-Kaj) has the following formulation:

$$R = 0.043\epsilon \tag{7.1}$$

The next important study to consider is that of Saidi et al. [64], which advances the SpaRTA methodology presented in Schmelzer et al. by developing a CFD-driven deterministic symbolic identification algorithm. This new approach involves solving a high-dimensional black-box optimization problem using sensitivity analysis and a Constrained Optimization using Response Surface (CORS) algorithm to reduce associated computational expenses. The models discovered in Saidi et al.'s study were tested alongside those from Schmelzer et al. for the same 2D-separated flow cases and demonstrated improved accuracy and flexibility. For this reason, one of the models from Saidi et al.'s study is chosen for comparison with the models in this study in Section 7.4. The best-performing model in Saidi et al.'s study appears to be Model 2 (referred to as Model-Saidi from now on) and has the following formulation:

$$b_{ij}^{\Delta} = -2.8356 \times 10^{-1} T_{ij}^{(1)} - 1.4738 \times 10^{-1} I_2^2 T_{ij}^{(2)} \tag{7.2}$$

$$R = \left(1.0375 \times 10^{-1} I_2 - 2.8833 \times 10^{-1} I_1 I_2\right) G_3 \tag{7.3}$$

These model equations are relatively similar to the ones discovered in this study, except that they do not include additional functions based on features. It is important to note that, similar to the models from Schmelzer et al., Model-Saidi also did not allow the APG and NASA-Hump simulations to converge initially. However, by restricting this model to the shear layer cluster, both simulations were able to converge. This further supports the claim that applying corrections only in regions where RANS turbulence models are known to struggle is highly beneficial. This is further discussed in Section 7.4.

Finally, it is important to mention the PhD thesis of Ali Amarloo [65], which combined CFD-driven optimization and progressive augmentation techniques to develop a Progressive data-augmented $k-\omega$ SST model. This model provides a correction for R, which, unlike the other studies mentioned above, applies the correction only in a region similar to the shear layer cluster used in this study. Ali Amarloo achieves this by using an activation function similar to the RITA/TI classifier, developed based on the ratio of production to dissipation of k. The models discovered for R take the following form:

$$R = 1 + F_{sep} = 1 + \alpha\Psi \tag{7.4}$$

$$\alpha = C_0 + C_1 B_1 + C_2 B_2 \tag{7.5}$$

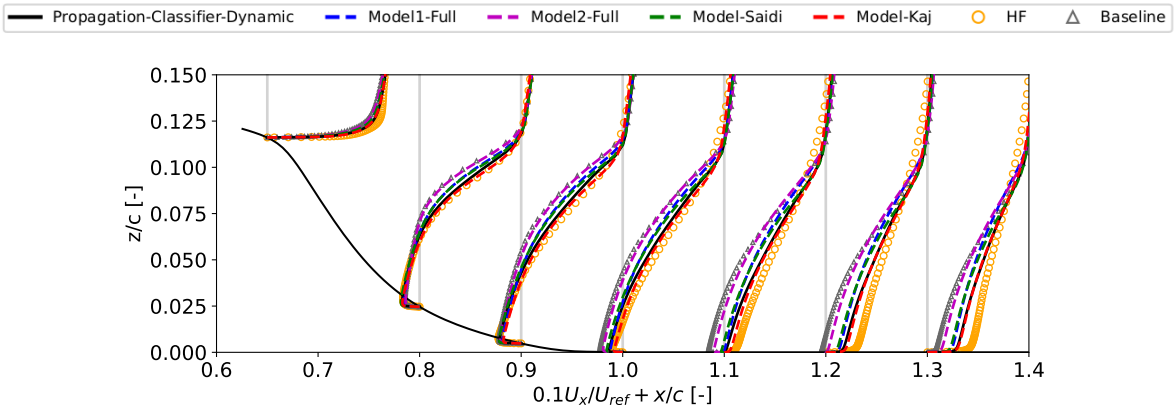
$$\Psi = \left(1 - \left(\nu_t \frac{\omega}{k}\right)^{\lambda_1}\right)^{\lambda_2} \tag{7.6}$$

Ψ is the activation function, which depends on the $\lambda_{1,2}$ optimization coefficients. α is the model correction, which depends on the normalized versions of the invariants I_1 and I_2, as well as the optimization coefficients C_0, C_1, and C_2. This model appears to perform very well for the same test cases used in this study (NASA-Hump, Periodic-Hill, and CBFS). Since this model was discovered in the final stages of this thesis project, there was not enough time to test it alongside the models obtained here. However, his study helps confirm the claims regarding the benefits of regressing and applying corrections only in the shear layer cluster, where RANS models fail to accurately predict the underlying physics. His approach of optimizing coefficients to create a shear layer activation function should be explored in future developments of the RITA/TI classifier, aiming to remove its dependence on the non-invariant TI feature and helping the classifier achieve better clustering results for the APG case.

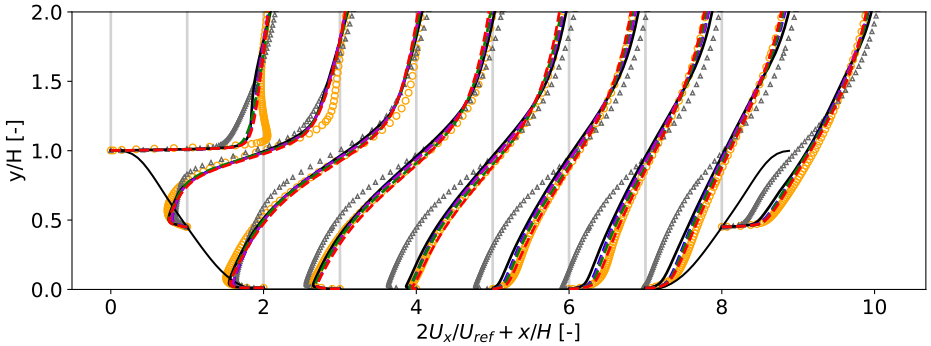
7.4. Combining b_{ij}^{Δ} and R Models

As discussed in the previous section, Model-Kaj obtained from the study of Kaj Hoefnagel [8] and Model-Saidi obtained from the study of Saidi et al. are tested alongside the combined b_{ij}^{Δ} and R Models discovered in this study. These combined models are based on Model3-Bij with Model4-R (referred to as Model1-Full) and Model3-Bij with Model3-R (referred to as Model2-Full). Model1-Full and Model2-Full are applied only in the shear layer cluster identified by the RITA/TI classifier, which is dynamically updated during the simulation. Model-Kaj is applied across the entire computational domain for all cases. Model-Saidi is applied in the entire domain for the Periodic-Hill and CBFS cases, while for the NASA-Hump and APG cases, it is restricted to the classifier region as the simulations failed to converge when it was applied across the entire domain for these cases.

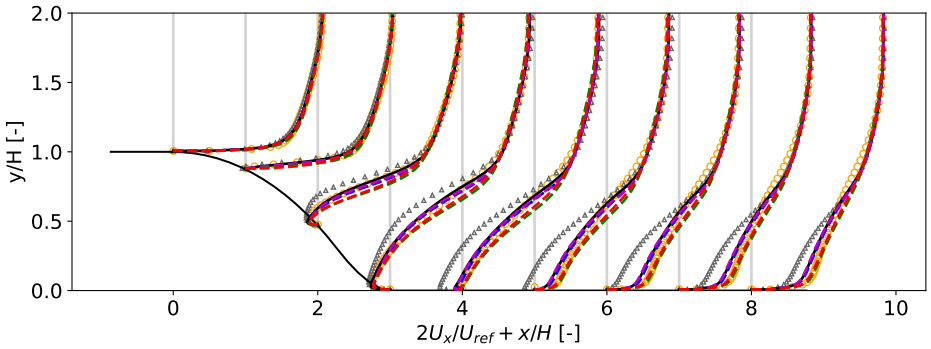
For comparison purposes, the results of the Baseline RANS, the Propagation-Classifier-Dynamic (PCD) simulation, and the high-fidelity data (HF) (LES or DNS depending on the case) are also displayed. Profiles of the axial velocity U_x, the turbulent kinetic energy k, and the shear stress component of the Reynolds-stress tensor $\overline{u'v'}$ are shown in Figures 7.16, 7.17, and 7.18 respectively. The separation and reattachment locations predicted by each Model Propagation simulation are listed in Table 7.7. Finally, the skin friction coefficient C_f plots obtained for each case are displayed in Figure 7.19.



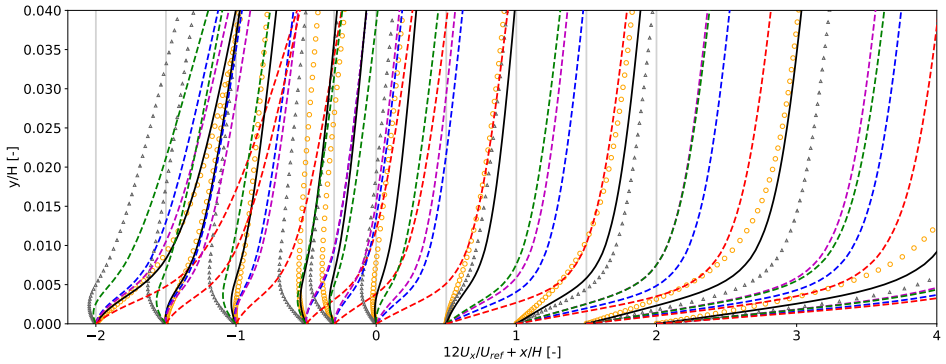
(a) U_x profiles NASA-Hump.



(b) U_x profiles Periodic-Hill.

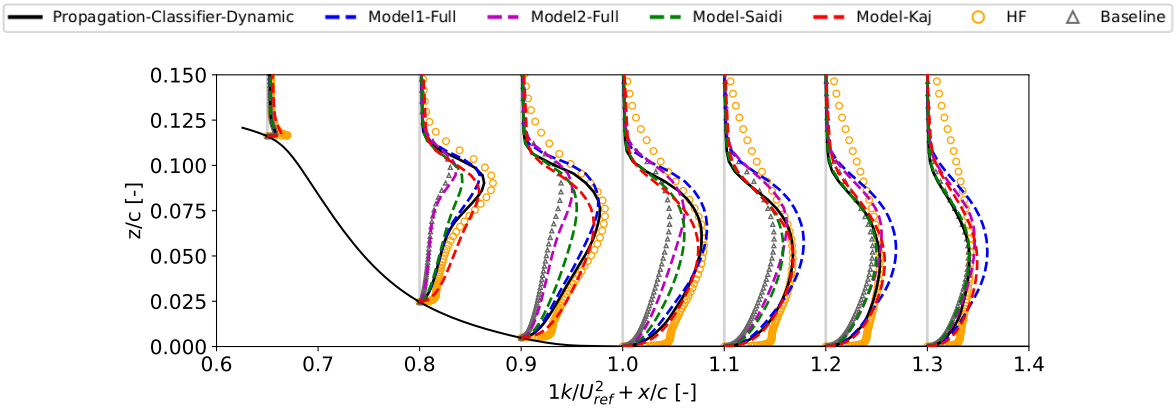


(c) U_x profiles CBFS.

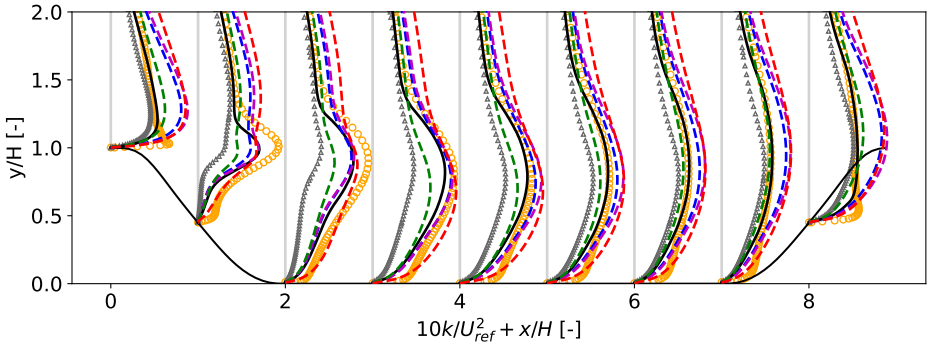


(d) U_x profiles APG.

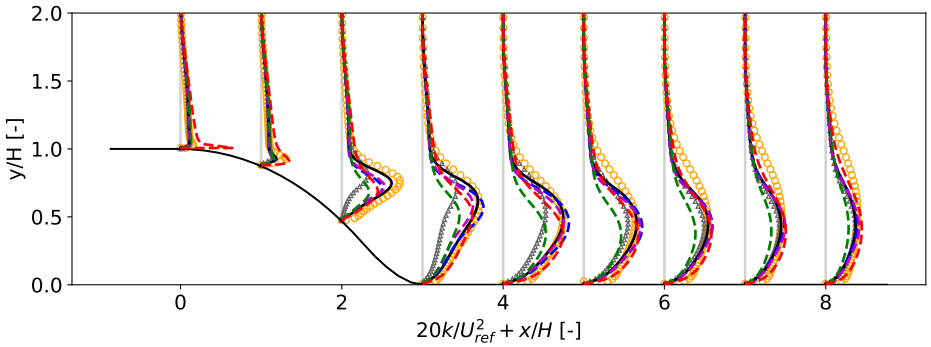
Figure 7.16: Comparison of U_x profiles obtained from the Model Propagation simulations.



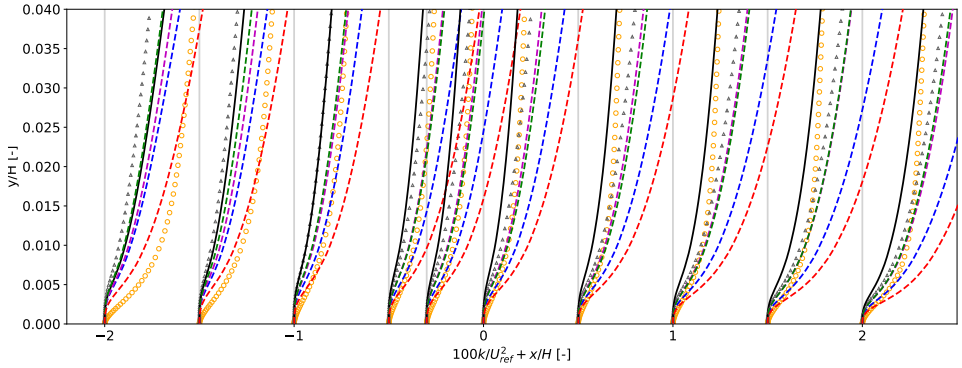
(a) k profiles NASA-Hump.



(b) k profiles Periodic-Hill.

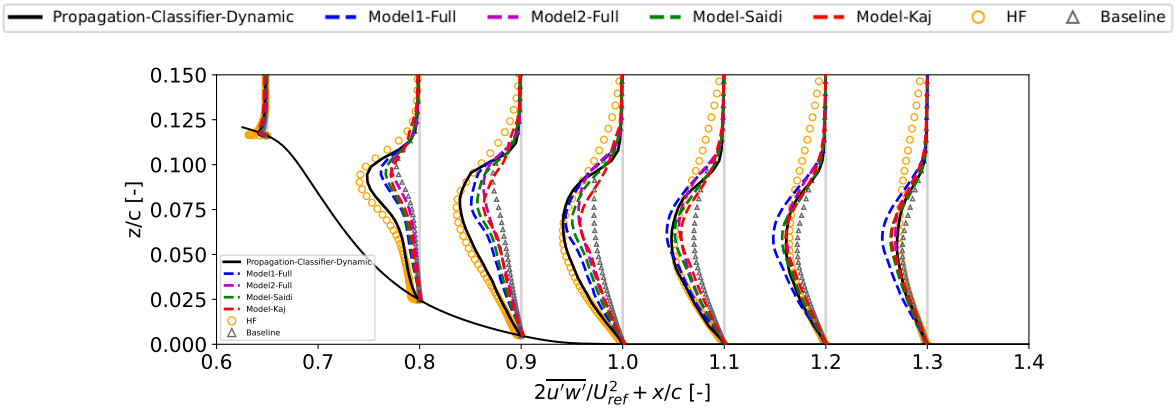


(c) k profiles CBFS.

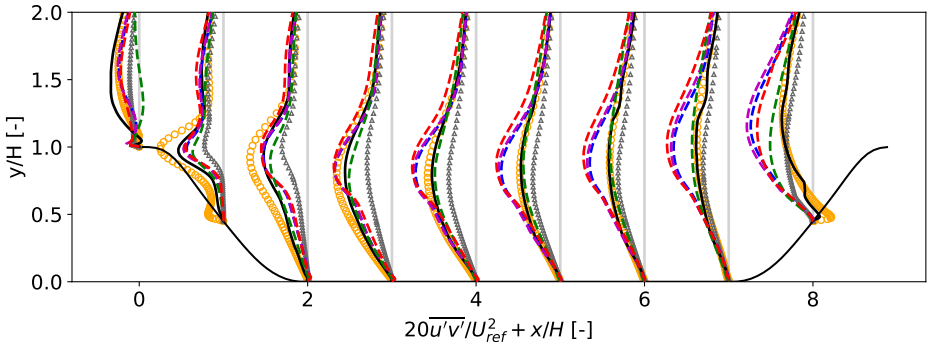


(d) k profiles APG.

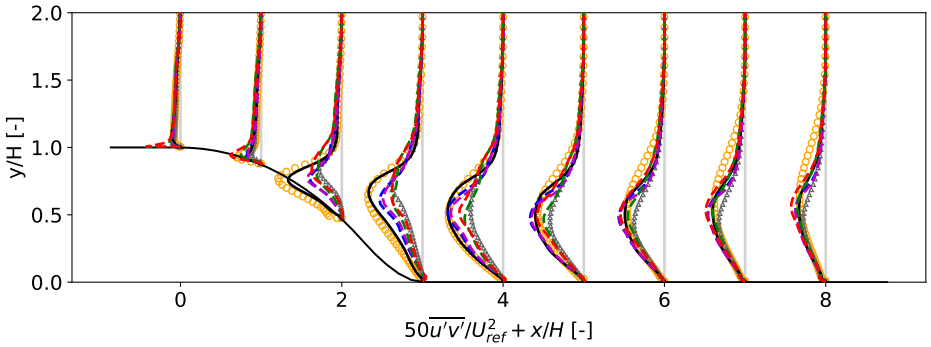
Figure 7.17: Comparison of k profiles obtained from the Model Propagation simulations.



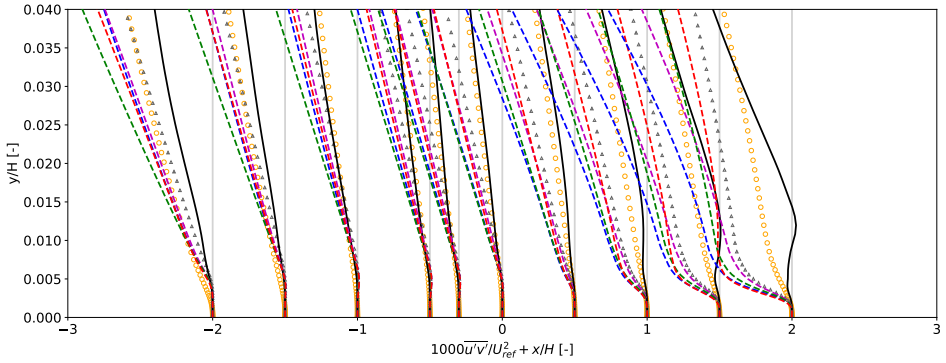
(a) $\overline{u'w'}$ profiles NASA-Hump.



(b) $\overline{u'v'}$ profiles Periodic-Hill.



(c) $\overline{u'v'}$ profiles CBFS.

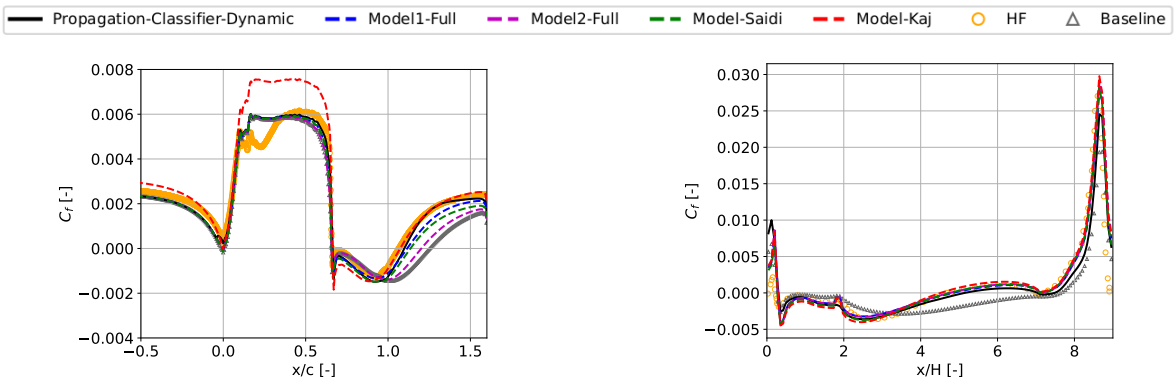


(d) $\overline{u'v'}$ profiles APG.

Figure 7.18: Comparison of $\overline{u'v'}$ ($\overline{u'w'}$ in the NASA-Hump case because the domain is designed in the x-z plane rather than the x-y plane) profiles obtained from the Model Propagation simulations.

Table 7.7: Comparison of the separation and reattachment location predictions obtained from the Model Propagation simulations and literature models.

Cases	NASA-Hump		Periodic-Hill		APG		CBFS	
	Separate x/c [-]	Reattach x/c [-]	Separate x/H [-]	Reattach x/H [-]	Separate x/H [-]	Reattach x/H [-]	Separate x/H [-]	Reattach x/H [-]
Model1-Full	0.66	1.12	0.28	4.83	-1.79	-0.47	0.95	4.46
Model2-Full	0.66	1.20	0.30	4.76	-1.66	-0.47	0.88	4.60
Model-Saidi	0.66	1.14	0.27	4.76	-2.35	0.03	1.08	4.19
Model-Kaj	0.66	1.07	0.28	4.52	-	-	1.27	4.23
PCD	0.66	1.09	0.30	5.15	-1.64	0.28	0.89	4.58
HF	0.66	1.06	0.22	4.69	-1.47	0.40	0.82	4.32
Baseline	0.65	1.25	0.26	7.64	-2.51	0.48	0.74	5.85



(a) C_f distribution across the bottom wall boundary of the NASA-Hump domain.

(b) C_f distribution across the bottom wall boundary of the Periodic-Hill domain.

(c) C_f distribution across the bottom wall boundary of the CBFS domain.

(d) C_f distribution across the bottom wall boundary of the APG domain.

Figure 7.19: Comparison of C_f distributions obtained from the Model Propagation simulations.

Both Model1-Full and Model2-Full show improvements over the Baseline RANS in terms of predicting the separation and reattachment locations and the axial velocity profiles U_x, for all cases apart from the APG case. They achieve this by reducing the underpredictions of k and $\overline{u'v'}$ in the initial stretches of the shear layer, thereby shrinking the recirculation bubble that forms as the flow separates from the surface.

For the NASA-Hump case, Model2-Full shows only a very marginal improvement over the Baseline RANS, while Model1-Full almost recovers the PCD simulation and high-fidelity predictions. This is expected because Model2-Full is composed of Model3-Bij and Model3-R, both of which, when tested in isolation, underpredict k and $\overline{u'v'}$, especially in the initial stretches of the shear layer. Combining these models amplifies their inability to accurately predict the production of turbulent kinetic energy in this location and hence the correct amounts of shear stress.

However, Model1-Full relies on Model3-Bij and Model4-R. While Model4-R significantly overpredicts k

and $\overline{u'v'}$ in the later stretches of the shear layer, it closely predicts the PCD and high-fidelity data in the initial stretches of the shear layer, which helps the model to achieve accurate flow reattachment predictions. When coupled with Model3-Bij, Model4-R effectively compensates for the underprediction of Model3-Bij in the initial stretches of the shear layer. Furthermore, it improves the predictions of k and $\overline{u'v'}$ in the later stretches of the shear layer.

Interestingly, Model2-Full appears to outperform Model1-Full for the APG and Periodic-Hill cases. For the APG case, Model1-Full overpredicts k and $\overline{u'v'}$ in the majority of the flow profile. The corrections applied by Model2-Full are less strong as evidenced by the underprediction observed in k and $\overline{u'v'}$ for the other cases. This gives Model2-Full an advantage in the APG case, as the corrections are applied in a cluster region slightly above the shear layer, which should not necessitate very strong corrections.

For the Periodic-Hill case, Model2-Full ends up overpredicting k and $\overline{u'v'}$ even more strongly than Model1-Full towards the end of the shear layer. Since this case has periodic boundary conditions, it means that the inlet to the domain where the shear layer starts forming receives a boost in the production of k, resulting in a higher shear stress. This effect leads to a more significant reduction in the recirculation region compared to Model1-Full, which is why Model2-Full achieves better reattachment predictions for this case.

Overall, Model1-Full appears to have a relatively constant prediction for changes in the Reynolds number of the flow. However, it becomes sensitive to steep geometrical changes, as evidenced by the Periodic-Hill case. Model2-Full, on the other hand, appears to be sensitive to both changes in Reynolds number and geometry, as evidenced by the NASA-Hump and CBFS cases. This outcome is not unexpected, as the SpaRTA training to discover these models relied on regressing using a very limited amount of data from only three cases (as the APG case data was not included in the regression). A less sensitive model could likely be discovered if new test cases with new geometries and Reynolds numbers are added to the training dataset. Nevertheless, even with this limited training data, the discovered models show very good performance.

It is therefore important to quantify this performance against the models obtained from the literature. As a reminder, Model-Saidi, which was designed to be applied in the full field, did not converge for the NASA-Hump case or the APG case and has therefore been restricted to the shear layer cluster. This reveals the advantage of only correcting the RANS simulations in regions where RANS struggles, rather than overcorrecting in regions where the RANS turbulence models have been calibrated to work well. This advantage is further evidenced by the performance of Model-Kaj, which was also applied in the full field for all cases. While this model achieves very good reattachment predictions for the majority of cases, it predicts no separation for the APG case and significantly affects the skin friction coefficient distribution for all cases, as can be observed in Figure 7.19.

Compared to Model-Saidi, the models discovered in this study show similar performance, which is not unexpected given the relatively similar formulation of the model equations. Comparing the results of the different cases, the models identified in this study slightly outperform Model-Saidi in predicting separation and reattachment locations. This indicates that by restricting the regression to the shear layer cluster, it is possible to discover models that specifically address the missing physics that the Baseline RANS fails to account for.

7.5. Final Conclusions

In this chapter, the symbolic model expressions discovered for the b_{ij}^{Δ} and R models, using the SpaRTA regression methodology on data from the shear layer cluster identified by the RITA/TI classifier, were discussed and compared. Each expression was tested in isolation to accurately evaluate its performance. Additionally, the best-performing expressions were combined in a Model Propagation simulations and compared to other similar models from the literature, the Baseline RANS, the PCD simulation results, and the high-fidelity data.

The best b_{ij}^{Δ} model was found to be Model3-Bij. This model was able to account for the missing anisotropy effects that Baseline RANS cannot reproduce, thereby reducing the underprediction of the turbulent kinetic energy and shear stress in the shear layer. This model, although not able to fully reproduce the high-fidelity data, still showed a good performance, also providing generalizability for different geometries and Reynolds numbers of the flows.

The two best R models were identified as Model3-R and Model4-R. Model4-R excelled at capturing the significant increase in the production of turbulent kinetic energy near the point of flow separation and in the initial stretches of the shear layer, a critical aspect that the Baseline RANS fails to address. However, Model4-R overpredicted the turbulent kinetic energy and shear stresses in the later stretches of the shear layer. Conversely, Model3-R, while less adept at predicting this initial burst in turbulent kinetic energy production, better reproduced the profiles of turbulent kinetic energy and shear stress in the later stretches of the shear layer.

While the underperformance of the models in the APG case can be attributed to classification errors, it is also important to note that the APG case's geometry is substantially different from the other test cases. The recirculation bubble that forms is much smaller, and since the Baseline RANS already provides a good prediction of the flow, applying these models in the correct region might still lead to overpredictions. This suggests that finding universal corrections for turbulence models is very difficult, if not impossible. However, this remains open for debate as the results of this study are not conclusive enough to support or disprove this claim. This is an important consideration for future studies.

When the b_{ij}^{Δ} and R models were combined in simulations, they demonstrated promising results, sometimes outperforming those obtained in other studies. It appears that limiting the regression to the classifier region enables models to better account for the missing shear layer physics that Baseline RANS fails to capture. Furthermore, by restricting corrections to this region, the portions of the flow domain, particularly those regions where Baseline RANS is well-calibrated such as the boundary layer, remain unaffected. This ensures that simulations converge more reliably and yield improved results across the entire computational domain.

These findings address the final two research sub-questions set for this study. To a significant extent, the SpaRTA framework can derive symbolic model expressions that accurately represent the Reynolds-stress tensor and the $k-\omega$ SST model-form error. Although these models do not fully recover the high-fidelity data, they provide a substantial improvement over the Baseline RANS.

Regarding changes in geometry and Reynolds number of the flow, the models showed varying performance, especially for the APG case as previously discussed. Overall, Model1-Full (comprising Model3-Bij and Model4-R) demonstrates relatively consistent predictions for changes in Reynolds number but becomes sensitive to steep geometrical changes, as evidenced by the Periodic-Hill case. Conversely, Model2-Full (comprising Model3-Bij and Model3-R) is sensitive to both changes in Reynolds number and geometry, as seen in the NASA-Hump and CBFS cases.

While both models show some sensitivity to changing conditions, they remain generalizable to a certain extent, meaning they provide a notable improvement over the Baseline RANS. This suggests that even with their limitations, these models represent a meaningful step forward in turbulence modeling, enhancing predictive capabilities across various 2D-separated flow scenarios.

8

Conclusion and Future Recommendations

This study focused on addressing the ongoing challenges of classical RANS simulations in predicting flow separation for changing geometries and Reynolds numbers of the flow, to support F1 aerodynamic design. Flow separation directly impacts the pressure distribution around an F1 car, consequently influencing both drag and downforce. Thus, accurately determining the location of flow separation and subsequent reattachment is crucial in CFD simulations of F1 race cars. The $k-\omega$ SST model, commonly used by F1 aerodynamics teams, is known to underpredict turbulent shear stress in the separated shear layer that forms when the boundary layer detaches from the car's surface. This underprediction results in an overestimation of the recirculation region and a delayed reattachment location. Consequently, it leads to inaccurate predictions of pressure distribution and skin friction, making it challenging to compute drag and downforce accurately.

Correcting these RANS flow separation simulations presents a significant challenge, as corrections must be made without affecting regions of the flow where the $k-\omega$ SST model has been calibrated as this can lead to faulty predictions and convergence issues during a simulation. In recent years, a new field of research known as Data-Driven turbulence modeling has developed as a response to the efforts made in discovering new and improved RANS models or calibrating existing ones to enhance their accuracy in predicting complex flows. This area of study focuses on applying machine learning tools, which are effective at handling extensive datasets and revealing complex underlying patterns, to help improve RANS predictions of turbulent flows. While various approaches have been developed in this field, this study focuses on SpaRTA, which uses a sparsity-promoting regression technique to derive algebraic models capable of addressing uncertainties in Reynolds-stress anisotropy and model-form errors in the $k-\omega$ SST model equation. The challenges encountered thus far with SpaRTA, which are also prevalent in the majority of other approaches used in the data-driven turbulence modeling field, have been to obtain models that are generalizable to different geometries and Reynolds numbers and that apply corrections only in areas of the flow where the RANS simulation shows the worst performance - the shear layer for separated flow cases.

This study aimed to address these challenges by developing a classifier capable of distinguishing the shear layer from the rest of the flow domain and applying the SpaRTA methodology to infer symbolic model equations for the Reynolds-stress tensor and model-form errors in the $k-\omega$ SST model, specifically targeting this region. The primary objective is to ensure the generalizability of these classifications and model corrective equations across various domain geometries and Reynolds numbers of the flow. By focusing the SpaRTA training exclusively on data obtained from the shear layer, the complexity of the regression problem can be simplified, reducing the volume of data to be fitted.

While each of the results chapters in this study has its own conclusion sections where the answers to the research questions defined for this study are presented, these conclusions will be restated here for clarity.

Thesis Research Questions

Sub-Questions:

1. *Can a classifier be constructed for the shear layer and effectively used to activate model corrections only where necessary?*

In this study, various methods have been investigated to obtain a generalizable classifier capable of identifying the shear layer region from the rest of the flow domain. The first set of methods relied on unsupervised clustering methods such as K-Means and GMM, which yielded poor results. Not only were the trained classifiers not generalizable, but they also failed to identify the shear layer. These methods are highly sensitive to the input feature dataset, and noise can significantly bias the clustering output. Feature datasets constructed from flow variables are notoriously difficult to preprocess: outlier removal often either negatively impacts the distribution of the features or fails to identify outliers.

The second method was based on designing a physics-inspired classifier from observations regarding the levels of turbulence production in the shear layer. This led to the development of the RITA/TI classifier, which demonstrated the best overall performance in identifying the shear layer from the rest of the domain. Because of the simple form of this classifier, which is based on the threshold of the production to destruction of turbulent kinetic energy and the turbulence intensity, the classifier assigns a value of 1 to domain cells situated inside the shear layer cluster and values of 0 to the rest of the domain, efficiently activating model corrections only where necessary.

2. *Is the classification consistent across different domain geometries and flow Reynolds numbers?*

The shear layer classification achieved by the RITA/TI classifier is relatively consistent across different domain geometries and flow Reynolds numbers. Furthermore, this classifier can be computed dynamically during a simulation and its predictions adjust according to the changing physics imposed by the corrections for example during Propagation and Model Propagation simulations. Nevertheless, the classification does appear to be sensitive to the predictions of the F_1 and F_2 blending functions of the $k-\omega$ SST model. This dependency reduces predictive power in some cases, such as in the APG case, where it predicts the shear layer cluster slightly too high above the recirculation region. Furthermore, since this classifier is based on turbulence intensity, which is not Galilean invariant, it is likely to fail for flows with several different moving reference frames.

3. *Does applying corrections exclusively in the shear layer region result in improved flow predictions in separated-flow scenarios?*

Indeed, applying corrections exclusively in the shear layer region results in improved flow predictions in separated-flow scenarios. This was proven by implementing the true b_{ij}^{Δ} and R corrections obtained from the Frozen simulation only within the shear layer cluster identified by the RITA/TI classifier. While this approach does not fully replicate the high-fidelity results, as flow predictions in other regions of the domain also influence the outcome, applying corrections in the shear layer removes a significant portion of uncertainty in Baseline RANS.

4. *Can the SpaRTA framework derive symbolic model expressions that accurately represent the Reynolds-stress tensor and $k-\omega$ SST model-form error in the shear layer?*

The SpaRTA framework can indeed derive symbolic model expressions that accurately represent the Reynolds-stress tensor and the $k-\omega$ SST model form error. Although these models do not fully recover the high-fidelity data, they provide a substantial improvement over the Baseline RANS.

The best b_{ij}^{Δ} model was found to be Model3-Bij. This model was able to account for the missing anisotropy effects that Baseline RANS cannot reproduce, thereby reducing the underprediction of the turbulent kinetic energy and shear stress in the shear layer. This model, although not able to fully reproduce the high-fidelity data, still showed a good performance, also providing generalizability for different geometries and Reynolds numbers of the flows.

The two best R models were identified as Model3-R and Model4-R. Model4-R excelled at cap-

turing the significant increase in the production of turbulent kinetic energy near the point of flow separation and in the initial stretches of the shear layer, a critical aspect that the Baseline RANS fails to address. However, Model4-R overpredicted turbulent kinetic energy and shear stresses in the later stretches of the shear layer. Conversely, Model3-R, while less adept at predicting this initial burst in turbulent kinetic energy production, better reproduced the profiles of turbulent kinetic energy and shear stress in the later stretches of the shear layer.

These models showed the worst performance for the APG case. While this can be attributed to classification errors, it is also important to note that the APG case's geometry is substantially different from the other flow cases. The recirculation bubble that forms is much smaller, and since the Baseline RANS already provides a good prediction of the flow, applying these models in the correct region might still lead to overpredictions. This suggests that finding universal corrections for turbulence models is very difficult, if not impossible. However, this remains open for debate as the results of this study are not conclusive enough to support or disprove this claim. This is an important consideration for future studies.

When the b_{ij}^{Δ} and R models were combined in simulations, they demonstrated promising results, sometimes outperforming those obtained in other studies. It appears that limiting the regression to the classifier region enables models to better account for the missing shear layer physics that Baseline RANS fails to capture. Furthermore, the key distinction between these models and those found in the literature is their application exclusively within the shear layer cluster. By restricting corrections to this region, the portions of the flow domain, particularly those regions where Baseline RANS is well-calibrated, remain unaffected. This ensures that simulations converge more reliably and yield improved results across the entire computational domain.

5. *Do the derived model equations yield consistent results for changes in domain geometries or Reynolds numbers of the flow?*

Regarding changes in geometry and Reynolds number of the flow, the models showed varying performance, especially for the APG case. While the models show some sensitivity to changing conditions, they remain generalizable to a certain extent, meaning they provide a notable improvement over the Baseline RANS. This suggests that even with their limitations, these models represent a meaningful step forward in turbulence modeling, enhancing predictive capabilities across various 2D-separated flow scenarios.

Main Question: Can the understanding and prediction of shear layer dynamics in 2D-separated flows be improved by incorporating targeted model corrections based on isolated shear layer data?

This study indeed demonstrates that incorporating targeted model corrections based on isolated shear layer data improves the understanding and prediction of shear layer dynamics in 2D-separated flows. Baseline RANS primarily fails to account for the initial rise in the production of turbulent kinetic energy in the initial stretches of the shear layer, which increases the levels of shear stress and further fails to accurately predict the anisotropy effects of turbulence in this layer. By limiting the regression to isolated shear layer data, models accounting for the physics that Baseline RANS is not able to account for were discovered. Despite slight performance variations across different geometries and Reynolds numbers, the derived models contribute to enhancing the predictive capabilities of the shear layer dynamics in 2D-separated flows, which, as stated at the beginning of this chapter, is essential for F1 race car design.

Based on the results and conclusions obtained in this study, several future recommendations have been formulated:

1. **Classifier Development:**
   (a) The RITA/TI classifier developed in this study remains sensitive to changes in geometry and does not assure Galilean invariance. Therefore, future efforts should focus on addressing these two primary concerns. One approach to this would be to investigate the methodology developed by Ali Amarloo in his PhD thesis [65]. Instead of designing a separate classifier, he created an activation function for the shear layer, which he regressed together with the model corrective equation for R to obtain a model that activates only in the shear layer. This approach of regressing a function that activates in the shear layer region should be further

explored in the context of designing a shear layer classifier.

(b) If it is possible to improve upon the RITA/TI classifier, it would be interesting to re-analyze the performance of the b_{ij}^{Δ} and R models discovered in this study for the APG case. This could provide a definitive conclusion on whether the classifier is completely at fault for the poorer performance that these models show for this case, or whether it is very difficult or impossible to obtain corrective equations that are universally applicable.

(c) It would also be interesting to explore how the RITA/TI classifier would perform in 3D-separated flow cases, which can help demonstrate whether it can indeed be useful in F1 aerodynamic design.

2. **Regression Methods:** The SpaRTA methodology used to derive the b_{ij}^{Δ} and R models in this study is based on linear regression. It is likely that employing non-linear regression techniques, such as the CuRTA regression methods developed by Kaj Hoejnagel in his Master's thesis [8], or multi-ojective regression could lead to the discovery of models capable of capturing a greater portion of the underlying shear layer physics. Thus, it is recommended that future studies prioritize the use of more advanced regression methods to investigate the potential for obtaining improved models that better represent the underlying shear layer physics.

3. **Case Specifics:**

(a) This study focused on only four flow cases. Better corrective models can probably be regressed if the initial dataset includes numerous other cases with varying geometries and Reynolds numbers. Therefore, to enhance generalizability, it is recommended to expand the portfolio of 2D-separated flow cases. Furthermore, the levels of grid refinement varied among the separated flow cases. Future studies should aim to refine the mesh uniformly across all cases and to a very fine level, ensuring full resolution of the boundary layer and eliminating any secondary effects that may influence simulation results.

(b) To obtain the R and b_{ij}^{Δ} corrections for the APG case, the R correction had to be removed from the ω equation during the Frozen simulation run to allow for convergence of the correction fields in the Propagation simulations. This was discovered to be very problematic for the Propagation simulations, which were run only with R corrections active or only with b_{ij}^{Δ} corrections active. The results showed that, in this case, the correction fields do not represent the true model-form error. This is likely the reason why the best-performing models identified with SpaRTA were all found once the APG case was removed from the regression input. Therefore, future studies should either select cases where the same model-form error extraction can be used or perform a deeper analysis to understand why some cases show this particular convergence issue.

(c) The cases used in this study had various levels of refinement, which can influence the flow separation predictions, as the boundary layer is better resolved in some cases than in others. For consistency, future studies should strive to achieve the same level of refinement among all cases and aim for a y^+ value below 1, thereby ensuring that all mesh resolution effects are completely eliminated.

4. **Features:** The features used throughout this study serve two primary purposes: they constitute the input dataset for the clustering algorithms and form the dataset used for model discovery within the SpaRTA framework. The performance of the R models was found to be particularly sensitive to the types of features used during regression. Thus, it is crucial to construct features that accurately represent the local turbulence characteristics, thereby offering greater flexibility to the regression problem. However, several features employed in this study lacked invariance and independence from Reynolds number variations. This limitation diminished their usefulness for both the model regression procedure and shear layer classification. Therefore, it is highly recommended to explore the development of new features that satisfy the criteria of being both Reynolds number-independent and invariant. Such features should be capable of effectively distinguishing between the shear layer and the remainder of the flow domain.

5. **Modelling Ansatz:** The modeling approach used in this study to represent the model-form error in the Reynolds-Averaged Navier-Stokes (RANS) prediction of turbulence anisotropy is based on Pope's eddy viscosity hypothesis [17]. Additionally, the modeling technique employed to represent R in this study assimilates this term with the production term of the k equation of the $k-\omega$ SST

model. Both of these modeling approaches rely solely on simplifications and assumptions inherent to their respective derivations. Therefore, future studies should explore alternative modeling strategies that could potentially offer a more comprehensive and accurate representation of the uncertainties in RANS simulations. This may involve incorporating more sophisticated turbulence closure models.

Bibliography

[1] J. Ferziger, M. Perić, and R. Street, *Computational Methods for Fluid Dynamics*. Springer, 2020, ISBN: 978-90-6562-165-8.

[2] K. Hanjalić, S. Kenjeres, M. J. Tummers, and H. J. J. Jonker, *Analysis and Modelling of Physical Transport Phenomena*. Delft Academic Press/VSSD, 2007.

[3] A. Bouferrouk, *On the Applicability of Trapped Vortices to Ground Vehicles*. The International Vehicle Aerodynamics Conference, 2020, ISBN: 9780081001998.

[4] J. Katz, "Aerodynamics of Race Cars," *Annual Review of Fluid Mechanics*, vol. 38, pp. 27–63, 2006. DOI: 10.1146/annurev.fluid.38.050304.092016.

[5] FIA, *2023 Formula One Sporting Regulations*, 2022. [Online]. Available: https://www.fia.com/sites/default/files/fia_2023_formula_1_sporting_regulations_-_issue_2_-_2022-09-30.pdf.

[6] Reddit. "Small Amount of Flowviz on the Williams Car." (2024), [Online]. Available: https://www.reddit.com/r/formula1/comments/szgcja/small_amount_of_flowviz_on_the_williams/.

[7] Williams Racing. "Williams in 60 Seconds: Wind Tunnel." (2012), [Online]. Available: https://www.youtube.com/watch?v=cCUQoi_RzxI.

[8] K. Hoefnagel, "Multi–Flow Generalization in Data–Driven Turbulence Modeling: An Exploratory Study," Master's Thesis, Delft University of Technology, Delft, Netherlands, 2023.

[9] K. Duraisamy, G. Iaccarino, and H. Xiao, "Turbulence Modeling in the Age of Data," *Annual Review of Fluid Mechanics*, vol. 51, pp. 357–377, 2019. DOI: 10.1146/annurev-fluid-010518-040547.

[10] P. Cinnella, "Data-driven turbulence modeling," *arXiv preprint arXiv:2404.09074*, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2404.09074.

[11] R. D. Sandberg and Y. Zhao, "Machine-learning for turbulence and heat-flux model development: A review of challenges associated with distinct physical phenomena and progress to date," *International Journal of Heat and Fluid Flow*, vol. 95, p. 108 983, 2022. DOI: https://doi.org/10.1016/j.ijheatfluidflow.2022.108983.

[12] M. Emory, J. Larsson, and G. Iaccarino, "Modeling of Structural Uncertainties in Reynolds-averaged Navier-Stokes Closures," *Physics of Fluids*, vol. 25, p. 110 822, 2013. DOI: 10.1063/1.4824659.

[13] P. Platteeuw, G. Loeven, and H. Bijl, "Uncertainty Quantification Applied to the $k-\epsilon$ Model of Turbulence Using the Probabilistic Collocation Method," Jun. 2012. DOI: 10.2514/6.2008-2150.

[14] S. H. Cheung, T. A. Oliver, E. E. Prudencio, S. Prudhomme, and R. D. Moser, "Bayesian Uncertainty Analysis with Applications to Turbulence Modeling," *Reliability Engineering & System Safety*, vol. 96, no. 9, pp. 1137–1149, 2011. DOI: 10.1016/j.ress.2010.09.013.

[15] W. N. Edeling, P. Cinnella, R. P. Dwight, and H. Bijl, "Bayesian Estimates of Parameter Variability in the $k-\epsilon$ Turbulence Model," *Journal of Computational Physics*, vol. 258, pp. 73–94, 2014.

[16] J. Ling and J. Templeton, "Evaluation of machine learning algorithms for prediction of regions of High Reynolds averaged Navier Stokes uncertainty," *Physics of Fluids*, vol. 27, no. 8, Aug. 2015. DOI: 10.1063/1.4927765.

[17] S. Pope, "A more general effective-viscosity hypothesis," *Journal of Fluid Mechanics*, vol. 72, pp. 331–340, 1975. DOI: 10.1017/S002211207500056X.

[18] J. Weatheritt and R. Sandberg, "A novel evolutionary algorithm applied to algebraic modifications of the RANS stress–strain relationship," *Journal of Computational Physics*, vol. 325, pp. 22–37, 2016. DOI: 10.1016/j.jcp.2016.08.009.

[19] M. Schmelzer, R. P. Dwight, and P. Cinnella, "Discovery of Algebraic Reynolds-stress Models Using Sparse Symbolic Regression," *Flow, Turbulence and Combustion*, vol. 104, pp. 579–603, 2019. DOI: 10.1007/s10494-019-00089-x.

[20] D. C. Wilcox, *Turbulence Modelling for CFD*. DCW Industries Inc., 1994.

[21] O. Reynolds, "On the dynamical theory of incompressible viscous fluids and the determination of the criterion," *Proceedings of the Royal Society of London*, vol. 56, pp. 40–45, 1894. DOI: 10.1098/rspl.1894.0075.

[22] J. Boussinesq, *Essai sur la théorie des eaux courantes*, French. Imprimerie Nationale, 1877, vol. 1, pp. XXII-680–61. [Online]. Available: http://catalogue.bnf.fr/ark:/12148/cb301489808.

[23] F. R. Menter, "Two–Equation Eddy–Viscosity Turbulence Models for Engineering Applications," *AIAA Journal*, vol. 32, pp. 1598–1605, 1994. DOI: 10.2514/3.12149.

[24] D. Wilcox, *A half century historical review of the $k-\omega$ Model*. 29th Aerospace Sciences Meeting, 1991. DOI: 10.2514/6.1991-615.

[25] W. P. Jones and B. E. Launder, "The Prediction of Laminarization with a Two-Equation Model of Turbulence," *International Journal of Heat and Mass Transfer*, vol. 15, pp. 301–314, 1972. DOI: 10.1016/0017-9310(72)90076-2.

[26] B. Launder, "Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc," *International Communications in Heat and Mass Transfer*, vol. 1, no. 2, pp. 131–137, 1974. DOI: 10.1016/0735-1933(74)90024-4.

[27] D. C. Wilcox, "Reassessment of the Scale Determining Equation for Advanced Turbulence Models," *AIAA Journal*, vol. 26, pp. 1299–1310, 1988. DOI: 10.2514/3.10041.

[28] W. Rodi and G. Scheuerer, "Scrutinizing the $k-\epsilon$ Turbulence Model Under Adverse Pressure Gradient Conditions," *Transactions of the ASME*, vol. 108, pp. 174–179, 1986. DOI: 10.1115/1.3242559.

[29] F. R. Menter, "Influence of freestream values on k-omega turbulence model predictions," *AIAA Journal*, vol. 30, pp. 1657–1659, 1992. DOI: 10.2514/3.11115.

[30] F. Menter, M. Kuntz, and R. Langtry, "Ten years of industrial experience with the SST turbulence model," *Heat and Mass Transfer*, vol. 4, 2003.

[31] P. G. Huang, P. Bradshaw, and T. J. Coakley, "Assessment of closure coefficients for compressible-flow turbulence models," NASA, Technical Memorandum TM-103882, 1992.

[32] V. Togiti, B. Eisfeld, and O. Brodersen, "Turbulence model study for the flow around the NASA common research model," *Journal of Aircraft*, vol. 51, pp. 1331–1343, 2014. DOI: 10.2514/1.c032609.

[33] J. Steiner, A. Viré, and R. P. Dwight, "Classifying regions of high model error within a data-driven RANS closure: Application to wind turbine wakes," *Flow, Turbulence and Combustion*, vol. 109, pp. 545–570, 2022. DOI: 10.1007/s10494-022-00346-6.

[34] OpenFOAM. "OpenFOAM – The Open Source CFD Toolbox." (2024), [Online]. Available: https://www.openfoam.com/.

[35] Skill-Lync. "CFD: All About the Convergence Criteria." (2022), [Online]. Available: https://skill-lync.com/blogs/technical-blogs/cfd-all-about-the-convergence-criteria.

[36] C. Rumsey. "2022 Symposium on Turbulence Modeling: Roadblocks, and the Potential for Machine Learning." (2022), [Online]. Available: https://turbmodels.larc.nasa.gov/turb-prs2022.html.

[37] D. Greenblatt, K. B. Paschal, C.-S. Yao, J. Harris, N. W. Schaeffler, and A. E. Washburn, "Experimental Investigation of Separation Control Part 1: Baseline and Steady Suction," *AIAA Journal*, vol. 44, pp. 2820–2830, 2006. DOI: 10.2514/1.13817.

[38] A. Uzun and M. R. Malik, "Wall–Resolved Large–Eddy Simulation of Flow Separation over NASA Wall–Mounted Hump," in *55th AIAA Aerospace Sciences Meeting*, American Institute of Aeronautics and Astronautics, 2017. DOI: 10.2514/6.2017-0538.

[39] M. Breuer, N. Peller, C. Rapp, and M. Manhart, "Flow over Periodic Hills – Numerical and Experimental Study in a Wide Range of Reynolds Numbers," *Computers & Fluids*, vol. 38, pp. 433–457, 2009. DOI: 10.1016/j.compfluid.2008.05.002.

[40] OpenFOAM–Documentation. “Guide to boundary conditions: Wall functions - nutlowrewallfunction.” (2024), [Online]. Available: https://www.openfoam.com/documentation/guides/latest/doc/guide-bcs-wall-turbulence-nutLowReWallFunction.html.

[41] Y. Bentaleb, S. Lardeau, and M. A. Leschziner, “Large-eddy simulation of turbulent boundary layer separation from a rounded step,” *Journal of Turbulence*, vol. 13, 2012. DOI: 10.1080/14685248.2011.637923.

[42] G. N. Coleman, C. L. Rumsey, and P. R. Spalart, “Numerical study of turbulent separation bubbles with varying pressure gradient and Reynolds number,” *Journal of Fluid Mechanics*, vol. 847, pp. 28–70, 2018. DOI: 10.1017/jfm.2018.257.

[43] Towards Data Science. “K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks.” (2018), [Online]. Available: https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a.

[44] Scikit-learn. “scikit-learn: Machine Learning in Python.” (2022), [Online]. Available: https://scikit-learn.org/stable/modules/clustering.html#k-means.

[45] D. Arthur and S. Vassilvitskii, “K-means++: The Advantages of Careful Seeding,” *Proc. of the Annu. ACM-SIAM Symp. on Discrete Algorithms*, vol. 8, pp. 1027–1035, Jan. 2007. DOI: 10.1145/1283383.1283494.

[46] Scikit-learn. “Sklearn.metrics.silhouette_score.” (2024), [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html.

[47] Wikipedia. “Pearson correlation coefficient.” (2024), [Online]. Available: https://en.wikipedia.org/wiki/Pearson_correlation_coefficient.

[48] H. Akoglu, “User’s Guide to Correlation Coefficients,” *Turkish Journal of Emergency Medicine*, vol. 18, no. 3, pp. 91–93, Sep. 2018. DOI: 10.1016/j.tjem.2018.08.001.

[49] Scikit-learn. “Sklearn.feature_selection.SequentialFeatureSelector.” (2024), [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SequentialFeatureSelector.html#sklearn.feature_selection.SequentialFeatureSelector.

[50] J. Hui. “Machine learning: Singular Value Decomposition (SVD) & Principal Component Analysis (PCA).” (2020), [Online]. Available: https://jonathan-hui.medium.com/machine-learning-singular-value-decomposition-svd-principal-component-analysis-pca-1d45e885e491.

[51] M. Patacchiola. “An overview of Gaussian Mixture Models.” (2020), [Online]. Available: https://mpatacchiola.github.io/blog/2020/07/31/gaussian-mixture-models.html.

[52] Scikit-learn. “Gaussian mixture models.” (2024), [Online]. Available: https://scikit-learn.org/stable/modules/mixture.html.

[53] J. L. Callaham, J. V. Koch, B. W. Brunton, J. N. Kutz, and S. L. Brunton, “Learning Dominant Physical Processes with Data-Driven Balance Models,” *Nature Communications*, vol. 12, p. 1016, 2021. DOI: 10.1038/s41467-021-21253-3.

[54] J. Wu, H. Xiao, and E. Paterson, “Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework,” *Physical Review Fluids*, vol. 3, Jul. 2018. DOI: 10.1103/physrevfluids.3.074602.

[55] J. Wang, J. Wu, and H. Xiao, “Physics-informed machine learning approach for reconstructing Reynolds Stress Modeling discrepancies based on DNS data,” *Physical Review Fluids*, vol. 2, Mar. 2017. DOI: 10.1103/physrevfluids.2.034603.

[56] P. R. Spalart, S. Deck, M. L. Shur, K. D. Squires, M. K. Strelets, and A. Travin, “A new version of detached-eddy simulation, resistant to ambiguous grid densities,” *Theoretical and Computational Fluid Dynamics*, vol. 20, pp. 181–195, May 2006. DOI: 10.1007/s00162-006-0015-0.

[57] Y. Yin, P. Yang, Y. Zhang, H. Chen, and S. Fu, “Feature selection and processing of turbulence modeling based on an artificial neural network,” *Physics of Fluids*, vol. 32, Oct. 2020. DOI: 10.1063/5.0022561.

[58] C. Wu and Y. Zhang, “Development of a Generalizable Data-driven Turbulence Model: Conditioned Field Inversion and Symbolic Regression,” *arXiv preprint*, 2024. DOI: 10.48550/arXiv.2402.16355.

[59] V. Srivastava, C. L. Rumsey, G. N. Coleman, and L. Wang, “On Generalizably Improving RANS Predictions of Flow Separation and Reattachment,” in *2024 AIAA SciTech Forum*, American Institute of Aeronautics and Astronautics, 2024.

[60] Y. Fang, Y. Zhao, F. Waschkowski, A. S. Ooi, and R. D. Sandberg, “Toward more general turbulence models via Multicase computational-fluid-dynamics-driven training,” *AIAA Journal*, vol. 61, pp. 2100–2115, May 2023. DOI: 10.2514/1.j062572.

[61] D. C. Wilcox, “Turbulence Modeling: An Overview,” in *Proceedings of the 39th Aerospace Sciences Meeting and Exhibit*, American Institute of Aeronautics and Astronautics, 2001, pp. 8–11. DOI: 10.2514/6.2001-724.

[62] C. Y. MacDougall, U. Piomelli, and F. Ambrogi, “Evaluation of Turbulence Models in Unsteady Separation,” *Fluids*, vol. 8, p. 273, 2023. DOI: 10.3390/fluids8100273.

[63] U. Piomelli, W. H. Cabot, P. Moin, and S. Lee, “Subgrid scale backscatter in turbulent and transitional flows,” *Physics of Fluids*, vol. 3, pp. 1766–1771, 1991. DOI: 10.1063/1.857956.

[64] I. B. H. Saïdi, M. Schmelzer, P. Cinnella, and F. Grasso, “CFD-Driven Symbolic Identification of Algebraic Reynolds-Stress Models,” *Journal of Computational Physics*, vol. 456, p. 111 037, 2022. DOI: 10.1016/j.jcp.2022.111037.

[65] A. Amarloo, “Data-Driven RANS Modelling and Prediction of Turbulent Flows,” Ph.D. dissertation, Department of Mechanical & Production Engineering, Aarhus University, 2023.

A

Feature Plots for NASA-Hump Case

A.1. Raw Feature Distributions

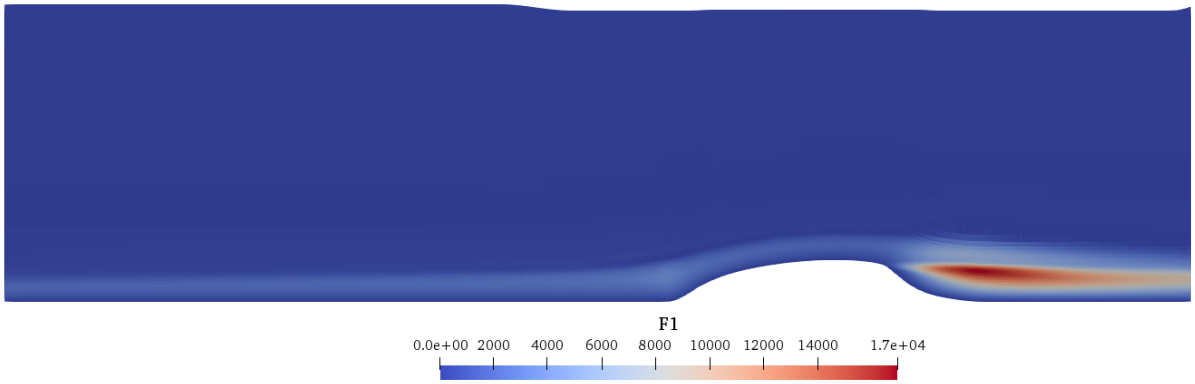


Figure A.1: The distribution of feature F1 (Re_d) for the NASA-hump test case.

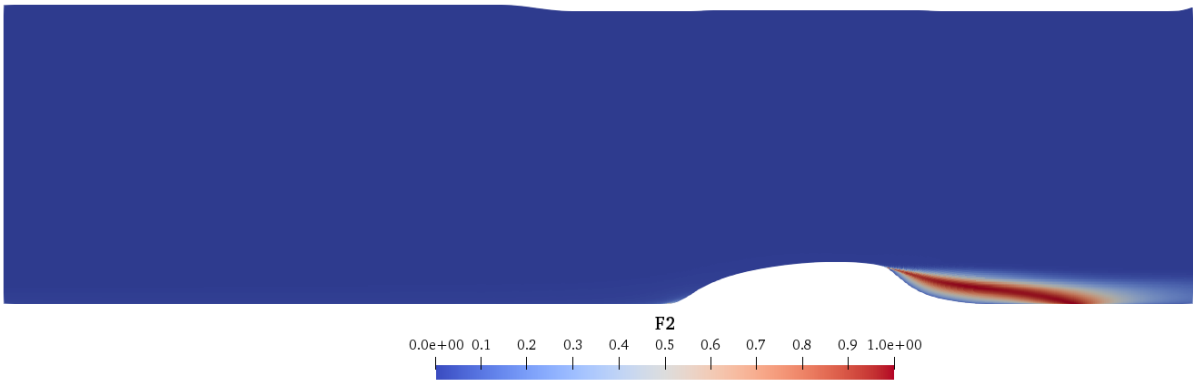


Figure A.2: The distribution of feature F2 (TI) for the NASA-hump test case.

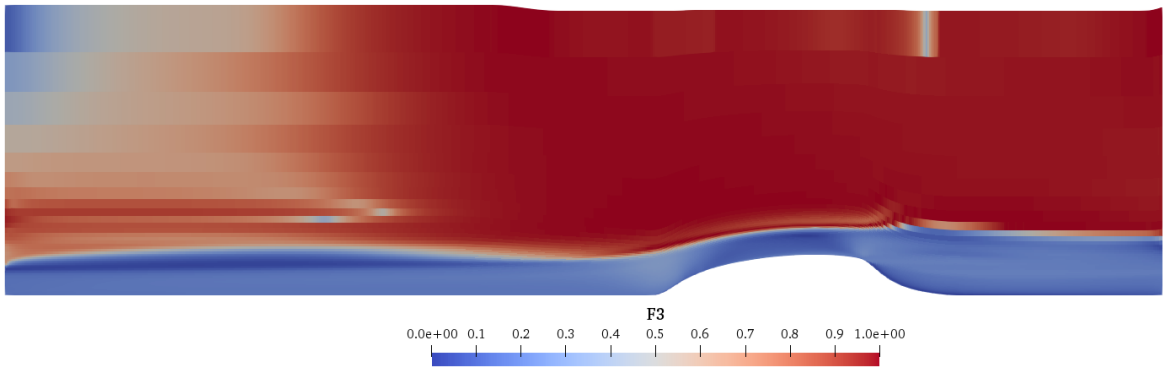


Figure A.3: The distribution of feature F3 (TS) for the NASA-hump test case.

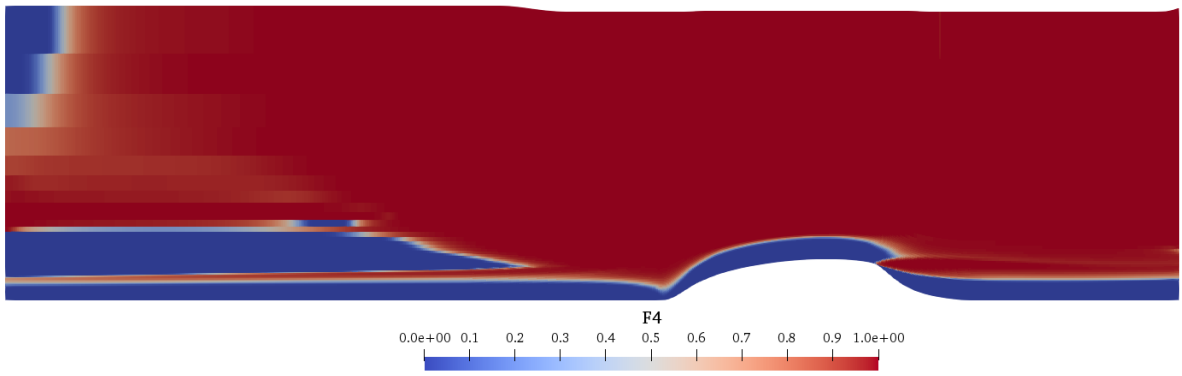


Figure A.4: The distribution of feature F4 (f_d) for the NASA-hump test case.

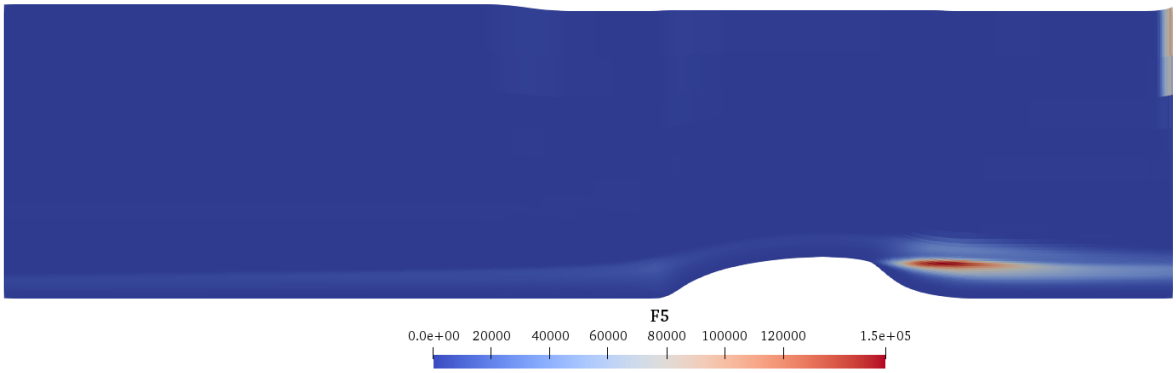


Figure A.5: The distribution of feature F5 (Re_Ω) for the NASA-hump test case.

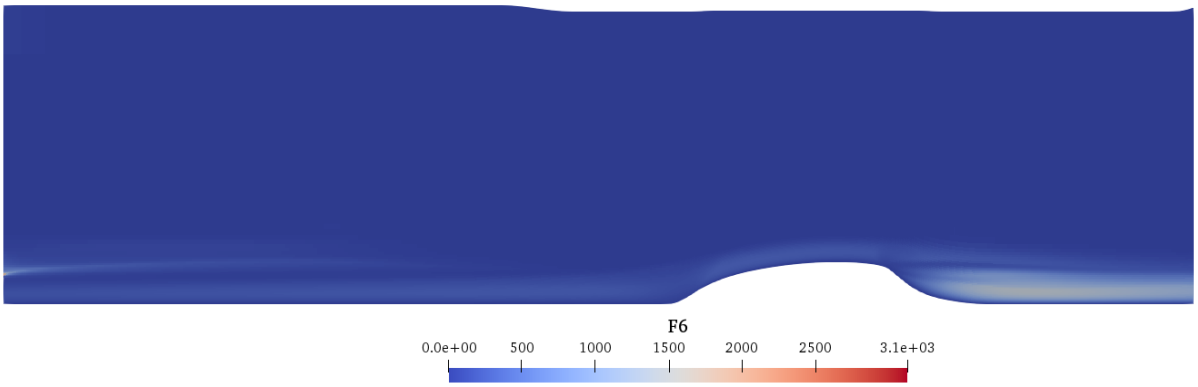


Figure A.6: The distribution of feature F6 (Re_k) for the NASA-hump test case.

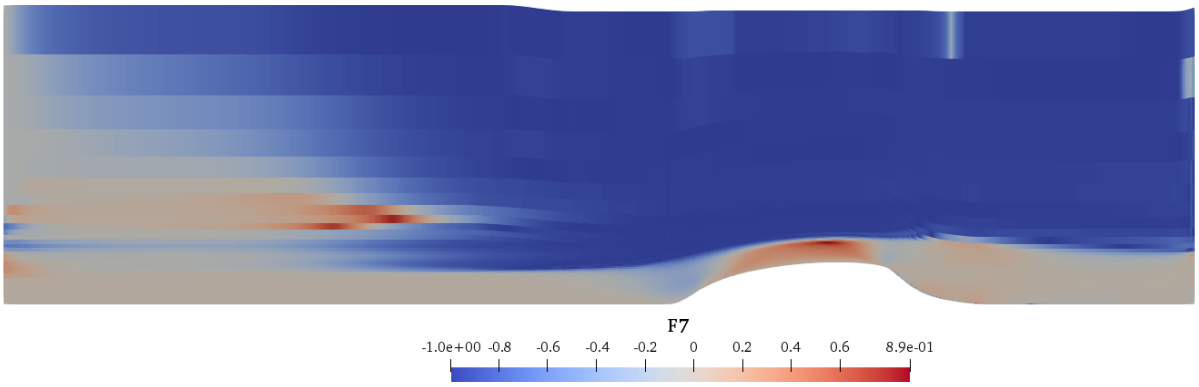


Figure A.7: The distribution of feature F7 ($Q_{criterion}$) for the NASA-hump test case.

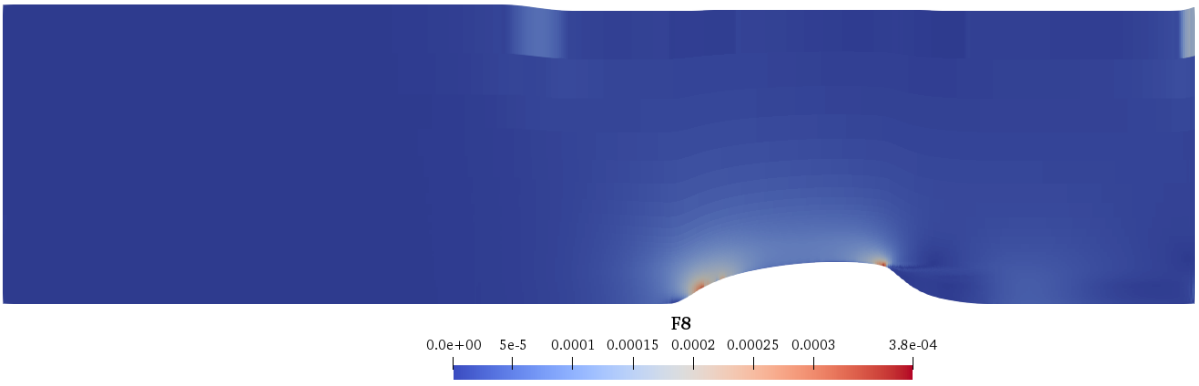


Figure A.8: The distribution of feature F8 (PS) for the NASA-hump test case.

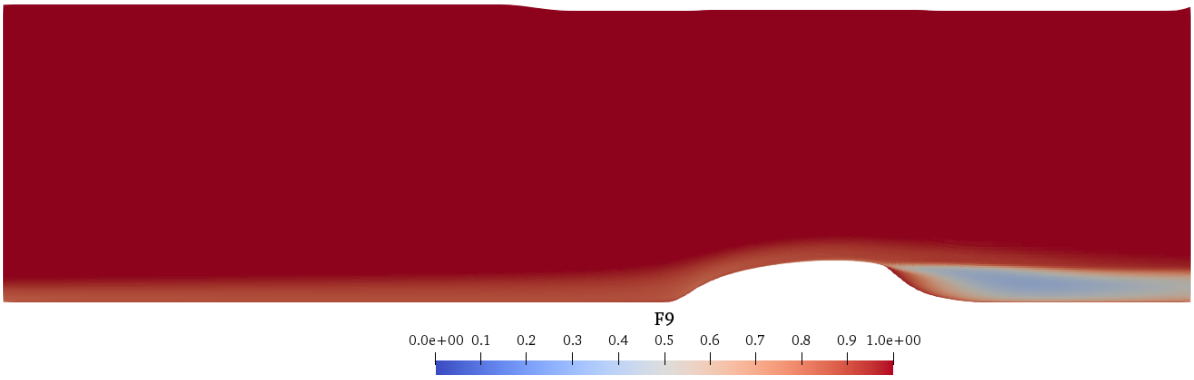


Figure A.9: The distribution of F9 ($\tau_{ij,ratio}$) for the NASA-hump test case.

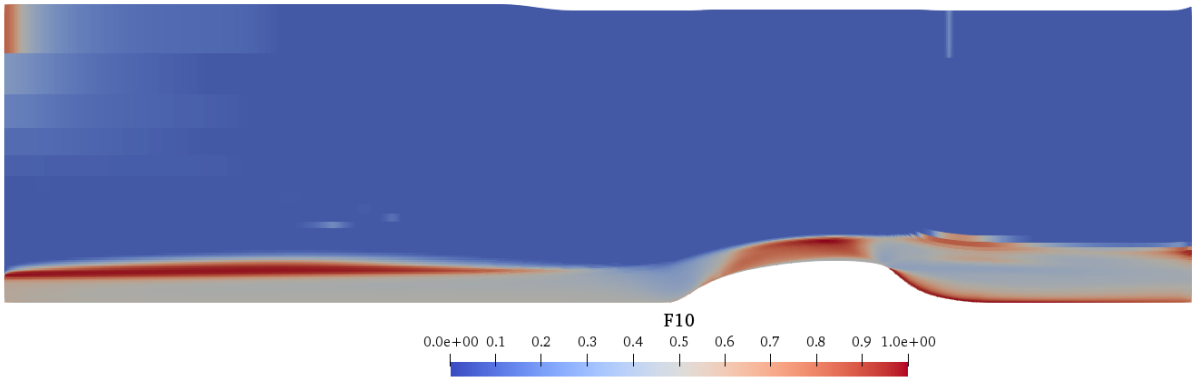


Figure A.10: The distribution of F10 (RITA$_{P_k/D_k}$) for the NASA-hump test case.

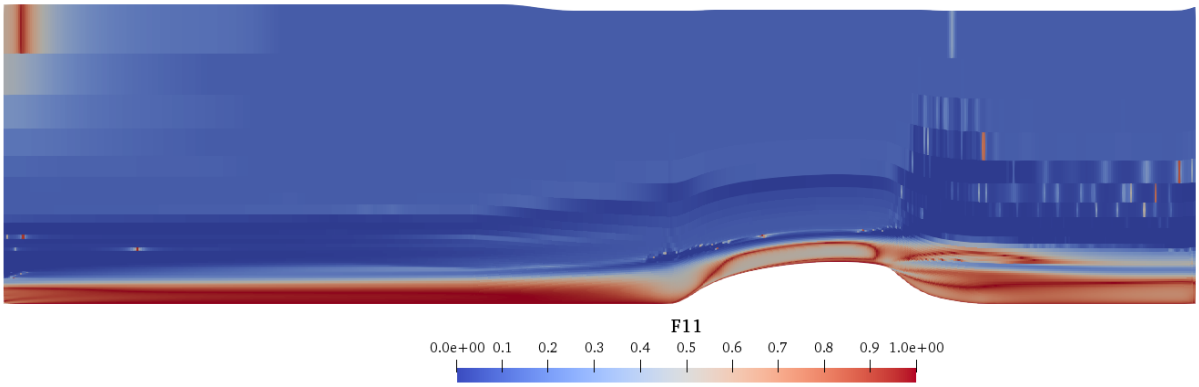


Figure A.11: The distribution of F11 (RITA$_{C_k/D_k}$) for the NASA-hump test case.

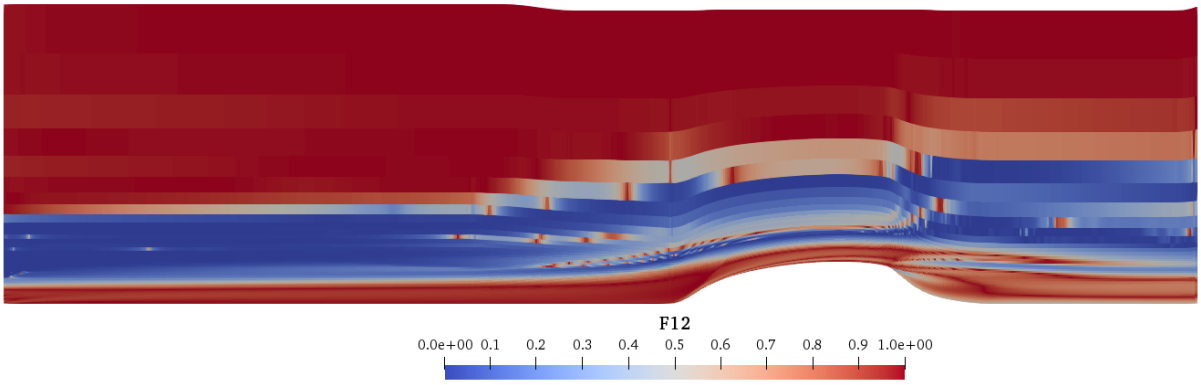


Figure A.12: The distribution of F12 (RITA$_{D_{f,k}/D_k}$) for the NASA-hump test case.

A.2. Feature Distributions Post Outlier Removal

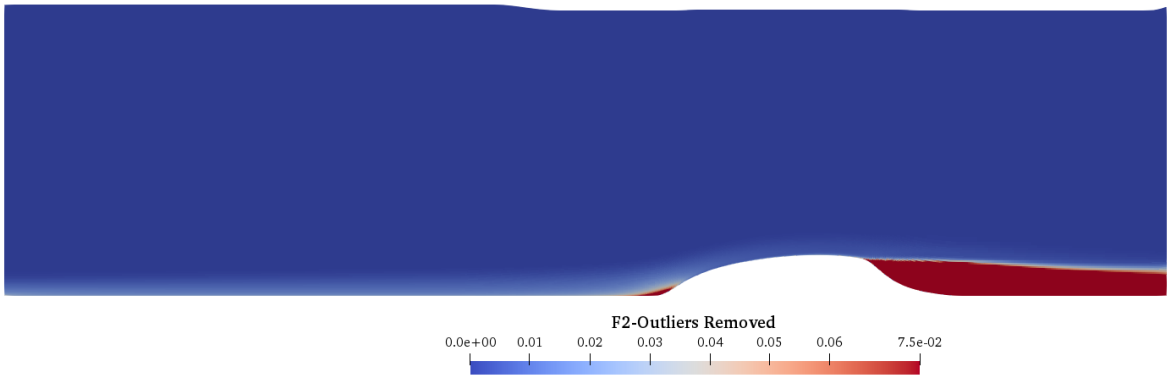


Figure A.13: Distribution of feature F2 (TI) for the NASA-Hump test case after outliers have been removed from this feature.

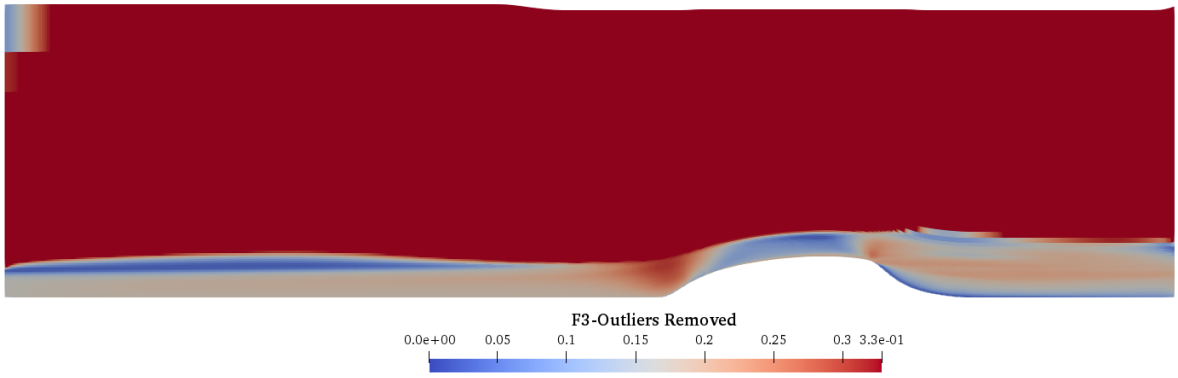


Figure A.14: Distribution of feature F3 (TS) for the NASA-Hump test case after outliers have been removed from this feature.

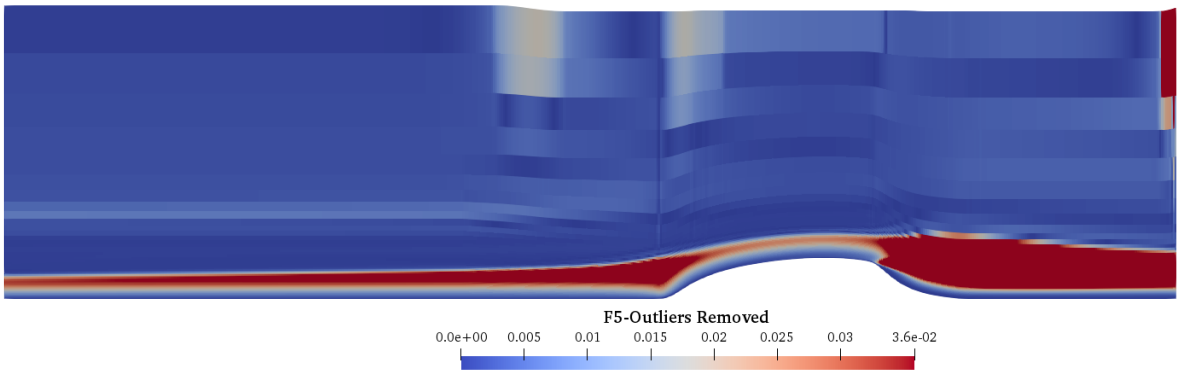


Figure A.15: Distribution of feature F5 (Re_{Ω}) for the NASA-Hump test case after outliers have been removed from this feature.

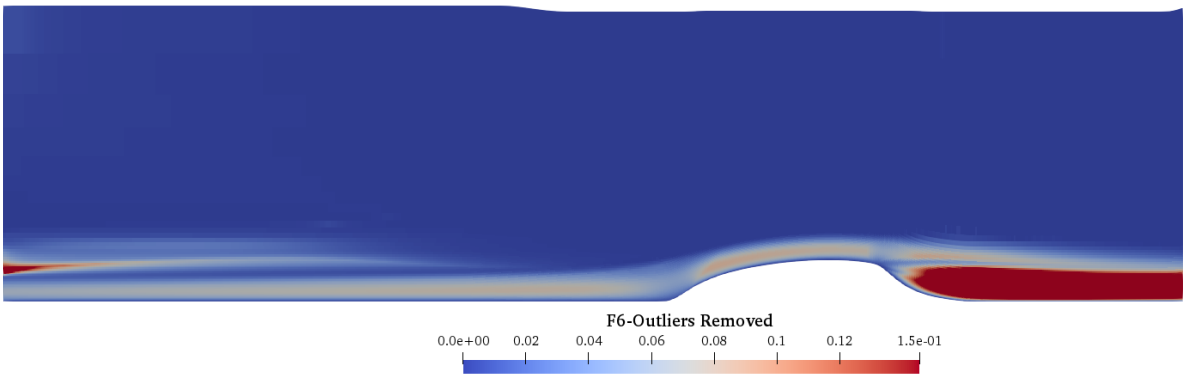


Figure A.16: Distribution of feature F6 (Re_k) for the NASA-Hump test case after outliers have been removed from this feature.

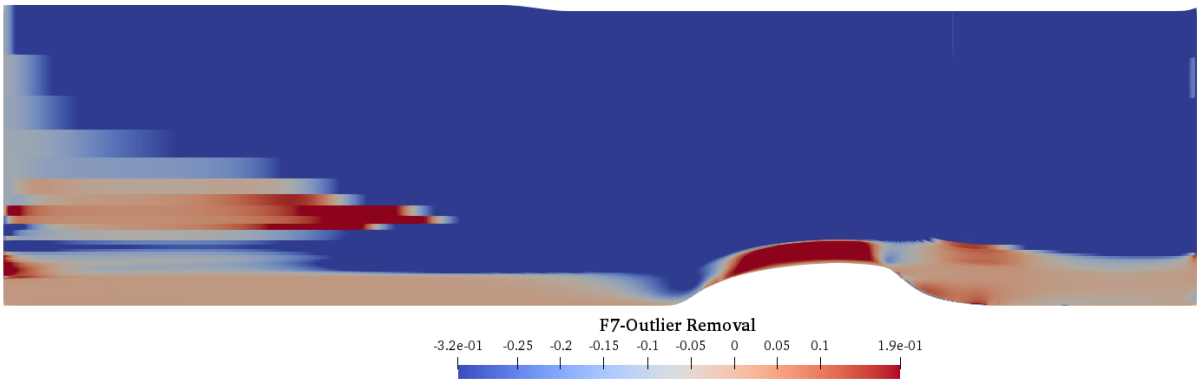


Figure A.17: Distribution of feature F7 ($Q_{criterion}$) for the NASA-Hump test case after outliers have been removed from this feature.

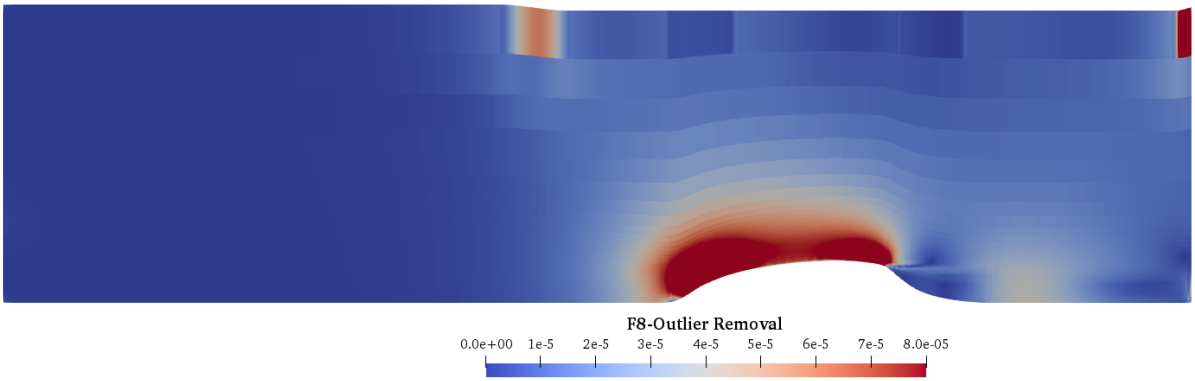


Figure A.18: Distribution of feature F8 (PS) for the NASA-Hump test case after outliers have been removed from this feature.

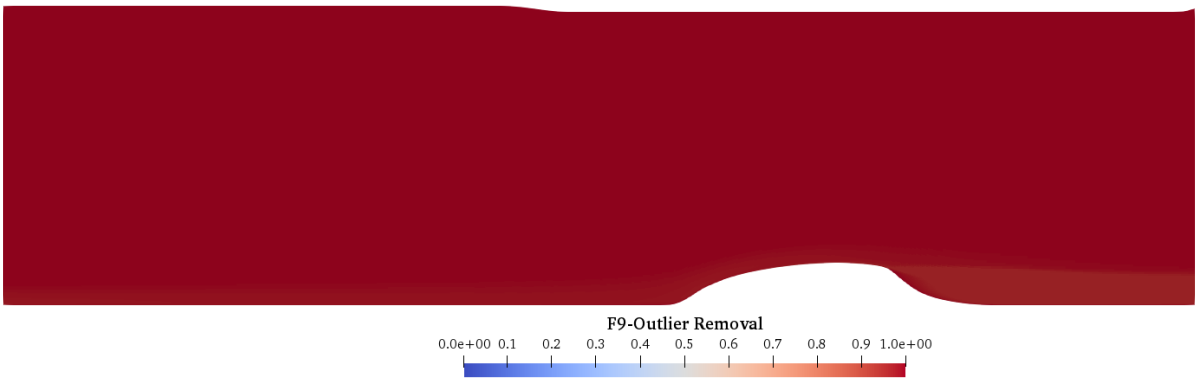


Figure A.19: Distribution of feature F9 ($\tau_{ij,ratio}$) for the NASA-Hump test case after outliers have been removed from this feature.

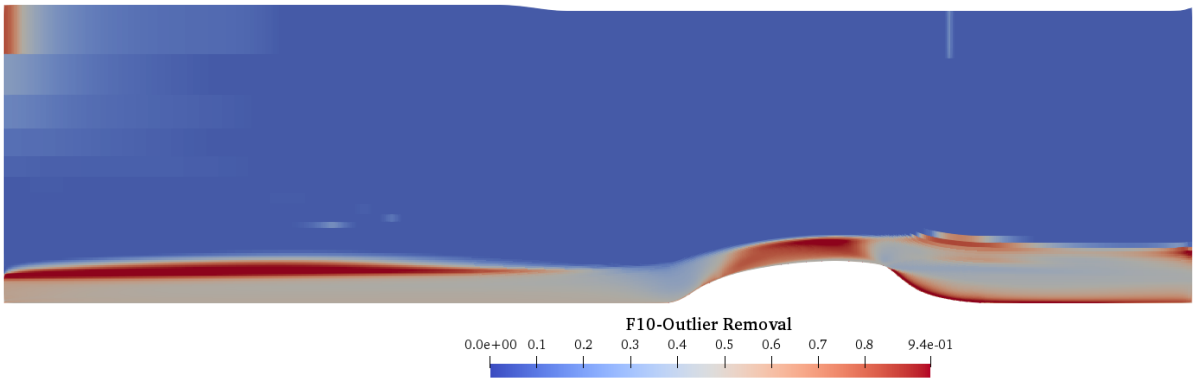


Figure A.20: Distribution of feature F10 (RITA$_{P_k/D_k}$) for the NASA-Hump test case after outliers have been removed from this feature.

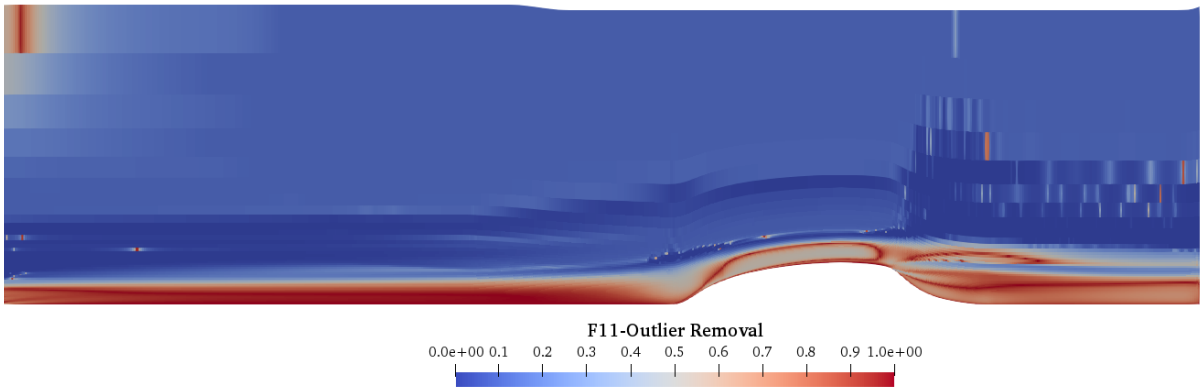


Figure A.21: Distribution of feature F11 (RITA$_{C_k/D_k}$) for the NASA-Hump test case after outliers have been removed from this feature.

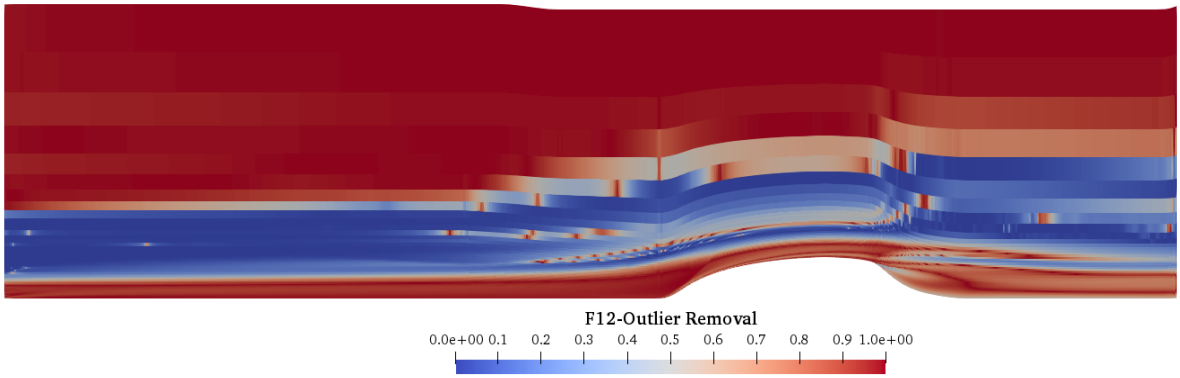


Figure A.22: Distribution of feature F12 (RITA$_{D_{f,k}/D_k}$) for the NASA-Hump test case after outliers have been removed from this feature.

B

Invariant RITA Classifier Development

The efforts towards developing an invariant version of the RITA/TI classifier described in this section focus on feature F9 ($\tau_{ij,ratio}$) in section 4.4. This feature is based on the ratio of total to normal Reynolds stresses. Unlike the turbulence intensity used in the RITA/TI classifier, this feature is Galilean invariant. As it is not normalized in its raw form, it is here constrained to a [0,1] range using Min-Max scaling, before it is used to construct the classifier. The normalized version of this feature ($\tau_{ij,ratio,norm}$) is shown in Figure B.1 for the NASA-Hump case. Similar to the turbulence intensity, the distribution of this feature shows a clear distinction between the downstream area of the hump, where the shear layer is located, and the rest of the domain.

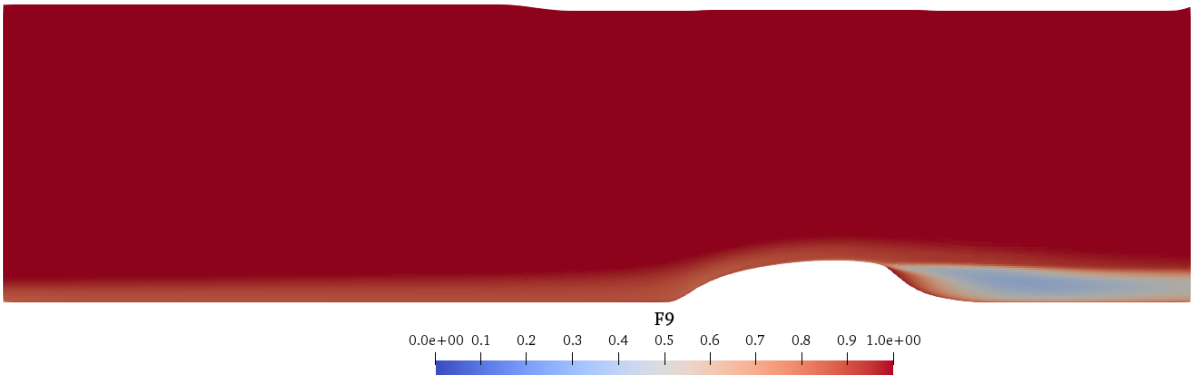


Figure B.1: The distribution of $\tau_{ij,ratio,norm}$ for the NASA-Hump case.

Setting a threshold on this ratio so that $\tau_{ij,ratio,norm} \leq 0.70$ and lowering the threshold on the RITA ratio to RITA$_{P_k/D_k} < 0.456$, ensures that only the shear layer region of the domain meets these conditions, while all other regions are excluded. The RITA ratio had to be lowered to ensure that the outer boundary layer region is not included in the classifier, as the $\tau_{ij,ratio,norm}$ is less accurate in terms of differentiating the shear layer region from the outer boundary layer compared to the turbulence intensity. The way the classifier is constructed based on these thresholds is further explained in the modeling note below.

Modelling Note - RITA/$\tau_{ij,ratio,norm}$ Classifier

The classifier, here denoted as σ, is based on the thresholds set on the RITA$_{P_k/D_k}$ ratio and the normalized ratio of total to normal Reynolds stresses ($\tau_{ij,ratio,norm}$). It assigns a value of 0 or 1 to every mesh cell in the domain, according to:

If: RITA$_{P_k/D_k} < 0.456$ and $\tau_{ij,ratio,norm} \leq 0.70 : \sigma = 1$ **Else**: $\sigma = 0$.

The mesh cells where $\sigma = 1$ indicate the location of the shear layer. The mesh cells where $\sigma = 0$ represent the rest of the domain.

Applying this classifier to all the separated flow cases leads to the clustering assignments displayed in Figures B.2, B.3, B.4 and B.5 for the NASA-Hump, CBFS, Periodic Hill and APG case respectively.

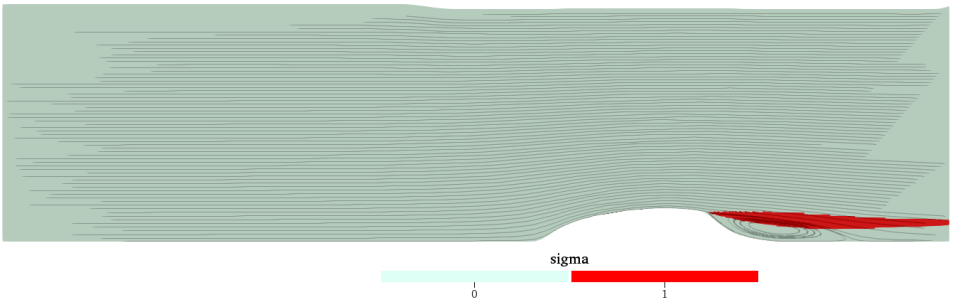


Figure B.2: σ classifier based on RITA/$\tau_{ij,ratio,norm}$ classifer for the NASA-Hump case.

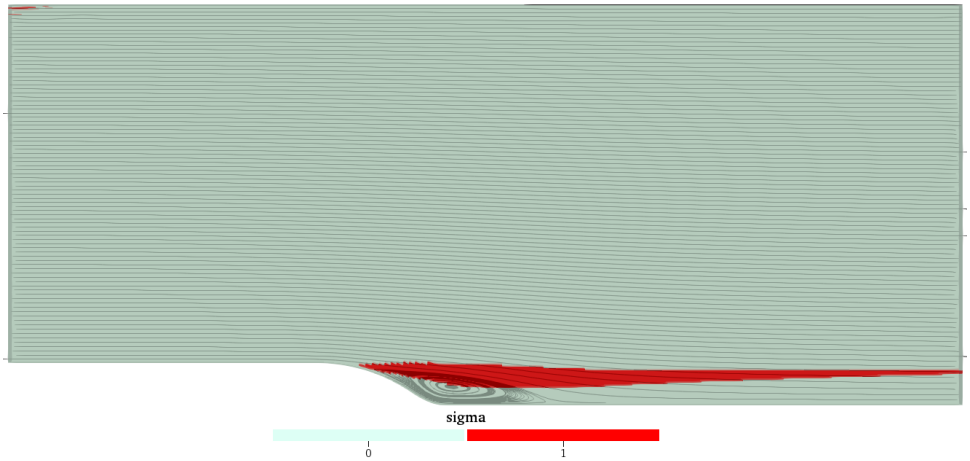


Figure B.3: σ classifier based on RITA/$\tau_{ij,ratio,norm}$ thresholding for the CBFS case.

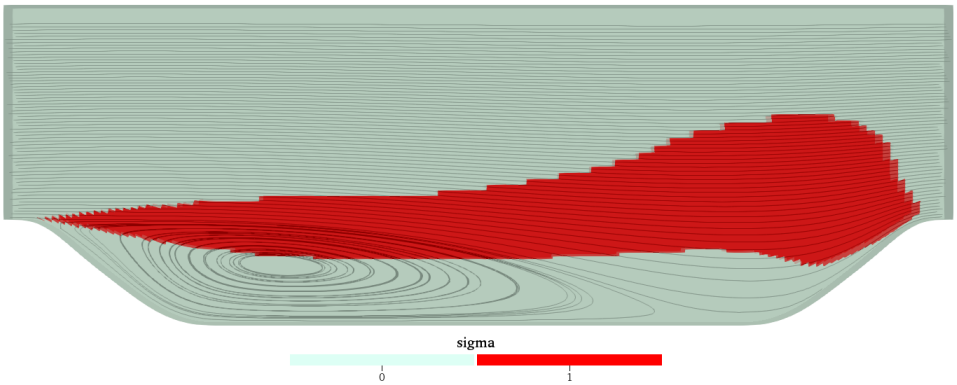


Figure B.4: σ classifier based on RITA/$\tau_{ij,ratio,norm}$ thresholding for the Periodic-Hill case.

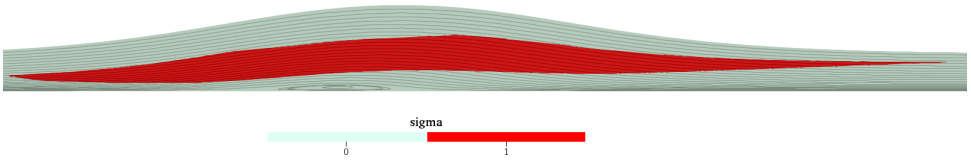


Figure B.5: σ classifier based on RITA/$\tau_{ij,ratio,norm}$ thresholding for the APG case.

As can be seen from comparing the above figures with the ones obtained using the RITA/TI classifier in Section 4.3, the shear layer cluster has maintained the same overall shape however it is now slightly wider, encompassing a larger area downstream of the recirculation region. Like the RITA/TI classifier, this alternative remains to a large extent generalizable across different geometries and flow conditions while also being fully invariant. Nevertheless, it still fails to accurately predict the location of the shear layer region for the APG case. This is likely due to the differences in the F_1 and F_2 blending functions of this case, as also discussed for the RITA/TI classifier

C

Simulation Infrastructure

C.1. `AugmentedkOmegaSST` Turbulence Model in OpenFOAM

The essence of this study lies in research; thus, the code infrastructure for conducting the various simulations necessary for testing and analyzing different correction fields and model equations should be flexible, efficient, and automated. This will enable it to be easily adapted for different types of simulations.

The OpenFOAM CFD solver used throughout this study is been written in C++, a compiled language. Therefore, if the models discovered through SpaRTA are implemented directly in this source code, the code must be recompiled each time a new model is investigated. This makes the overall process difficult to automate and time-consuming. Additionally, one must consider that C++ is a challenging language to program in, making the task of building all the necessary functions to compute the tensors, invariants, and features that constitute the different models (see Section 2.6) both complex and time-consuming. Furthermore, the SpaRTA code base, which will simply be referred to as `SpaRTA` from now on, has already been developed in Python, an interpreted language that does not need to be pre-compiled and is much more user-friendly. `SpaRTA` also already has all the functions defined for evaluating the tensors, invariants, and features that make up the models. Therefore, evaluating the models through `SpaRTA` is much more efficient and quick to implement. Testing different models requires no changes to the `SpaRTA` source code.

Taking into account the above considerations, Kaj Hoefnagel built a code infrastructure that couples the OpenFOAM source code to `SpaRTA` via a Python interpreter [8]. With his code, it is possible to perform both Propagation and Model Propagation simulations. In this study, several additional functionalities have been added to this code infrastructure to allow for more user input flexibility regarding the simulations being performed and to provide a more efficient approach to transferring data between `SpaRTA` and OpenFOAM. The new infrastructure consists of a turbulence model called `AugmentedkOmegaSST`, which is defined via a header file (`AugmentedkOmegaSST.H`) and a source file (`AugmentedkOmegaSST.C`), as well as a Python script called `python_model.py` that acts as an intermediary platform connecting `AugmentedkOmegaSST` and `SpaRTA`. The source code for all these files can be found in Section C.2 below.

C.1.1. Inputs to `AugmentedkOmegaSST`

The `AugmentedkOmegaSST` turbulence model enables users to carry out Baseline, Propagation, and Model Propagation simulations, with or without classifiers for the latter two, based on a series of inputs. An overview of these inputs is provided in Table C.1. The first input, when set to True, performs a standard Baseline simulation. The next three inputs, starting with the word 'use', specify whether to use the R correction field, the b_{ij}^{Δ} correction field, and the σ classifier.

If the following three inputs, which all start with the word 'model', are set to False and the previous three (with the word 'use') are set to True, a Propagation simulation is performed. In this case, the R and b_{ij}^{Δ} corrections are read from their respective fields obtained from the Frozen simulation and applied only in the shear layer cluster identified using the σ field. This field is derived by applying the RITA/TI classifier

to the converged Baseline simulation. If the user wants to perform a full Propagation simulation without the classifier, the useSigma input can be set to False. If the inputs that start with the word 'model' are set to True, alongside the 'use' inputs, a Model Propagation simulation is performed. If modelSigma input is set to True, then the σ classifier is updated at every solver iteration via the python_model.py file. The inputs offer full flexibility, allowing for scenarios such as performing a simulation where the R field is read from a file and the b_{ij}^{Δ} field is modeled using its respective model equation.

The inputs starting with the word 'ramp' specify the times to start applying corrections to the Baseline model using a ramp function and the end time at which full corrections are applied. This function, therefore, has a value of 0 at rampStartTime which then grows linearly to a value of 1 by rampEndTime. The R and b_{ij}^{Δ} correction terms are multiplied by the ramp function, ensuring that corrections are applied gradually rather than all at once, which can lead to stability issues in the simulation. The final two inputs specify whether additional factors should be used to reduce the strength of the R and b_{ij}^{Δ} corrections being applied. This can often help alleviate convergence and stability issues. For example, if this factor is set to 0.5, the corrections will be halved.

Based on these inputs, several if else statements have been used throughout the AugmentedkOmegaSST to allow for the correct simulation type to be performed.

Table C.1: Input specifications for theAugmentedkOmegaSST turbulence model.

Input	Value (Default)	Functionality
baseline	False	Switch to activate Baseline simulation.
usekDeficit	False	Switch to activate R corrections.
usebijDelta	False	Switch to activate b_{ij}^{Δ} corrections.
useSigma	False	Switch to activate σ classifier.
modelkDeficit	False	Switch to activate R model corrections.
modelbijDelta	False	Switch to activate b_{ij}^{Δ} model corrections.
modelSigma	False	Switch to update σ classifier at each solver iteration.
rampStartTime	10	Time at which corrections start being applied.
rampEndTime	100	Time at which full corrections are applied.
bijDeltastabilizer	1	Controls the intensity of b_{ij}^{Δ} corrections.
kDeficitstabilizer	1	Controls the intensity of R corrections.

C.1.2. Implementation Details of python_model.py Python Script

The python_model.py script serves two main purposes:

1. It computes the b_{ij}^{Δ} and R model corrections via the equations discovered using the SpaRTA methodology.
2. It dynamically updates the RITA/TI classifier during the simulation. The reason behind implementing this option of dynamic classification is further elaborated upon in Chapter 6.

An overview of the Python script algorithm is available in Algorithm 4 below. The initialization of the script is achieved using the Application Programmers Interface to Python, which allows a C++ program to access the Python interpreter (Python/C API). This interpreter is initialized in the AugmentedkOmegaSST.C source file at the start of the simulation and remains active throughout. During this initialization phase, the python_model.py script is loaded and executed. This ensures that all its global variables are defined and its libraries, including the SpaRTA code base, are loaded. Consequently, there is no need to reload these at every iteration, reducing the overhead of the simulation. Baseline and Propagation simulations do not require to be connected to the Python interpreter as they do not require any information from SpaRTA. Therefore, only for Model Propagation simulations, the python_model.py and its interpreter is employed.

The global variables of python_model.py include the paths to the files where the model equations are defined (bijDeltaEq for the b_{ij}^{Δ} model and kDeficitEq for the R model). Additionally, a dictionary named funcDict is created which defines the functions used by the model equation files. Using separate model equation files instead of hardcoding them in the Python script offers more flexibility. This approach allows

for the same Python script to be used for all Model Propagation simulations while enabling adjustments to the structure of the models via the model equation files.

Algorithm 4 Python script `python_model.py` algorithm.

Initialization:
1. Load libraries: `SpaRTA, numpy, copy, os` ,etc.
2. Load global variables: paths to `bijDeltaEq` and `kDeficitEq` files, `funcDict`.

Iterative Procedure (repeats at every solver iteration):

Input to `model` function:
Import `inputDict` from `AugmentedkOmegaSST`.

`model` **Function:**
Make a deep copy of `inputDict`.
Send deep copy of `inputDict` to `SpaRTA`.
Extract `featureDict` from `SpaRTA` output.
Evaluate `bijDeltaEq` and `kDeficitEq` equation files given `featureDict`.
Evaluate dynamic σ classifier output.
Build `ReturnDict` and send to `AugmentedkOmegaSST`.

Finalize:
Delete copy of `inputDict` from memory along with other locally defined variables.

To compute the tensors, invariants, and features that constitute the model equations, all necessary flow variables are sent from `AugmentedkOmegaSST` to the Python script via a dictionary at each solver iteration. This input dictionary, called `inputDict`, contains essential fields such as U, k, ω, etc., and is passed to the main function in the Python script, named `model()`. Within the `model()` function, `inputDict` is first deep-copied before being sent to the `SpaRTA` library, which computes all the input features, tensors, and invariants. This deep copy ensures that no information is lost during data manipulation in `SpaRTA`. Deleting the deep copy from memory at the end of the Python script protects against memory leaks.

Once all necessary parameters are computed using `SpaRTA`, the model equations are evaluated, and the outputs are stored in a new dictionary called `featureDict`. Additionally, if the 'modelSigma' input is set to True for `AugmentedkOmegaSST`, the σ classifier is updated based on the input fields provided by `AugmentedkOmegaSST`. Finally, a dictionary containing the outputs of the model equations (i.e. b_{ij}^{Δ} and R correction values for each point in the domain) and the classification (σ values of 0 or 1 for each point in the domain) is created. This dictionary, named `ReturnDict`, is then sent back to `AugmentedkOmegaSST`.

C.1.2.1 Main Code Structure of `AugmentedkOmegaSST`

As discussed in the section above, for Model Propagation simulations, the Python interpreter needs to be initialized and the `AugmentedkOmegaSST` turbulence model needs to be connected to the `python_model.py` script. The Python interpreter is started by calling `Py_Initialize()` from the Python/C API at the beginning of the `AugmentedkOmegaSST.C` source file. Next, the `python_model.py` script is loaded using the `PyImport_Import()` function. Additionally, the `model` function from `python_model.py` is loaded, making it available for subsequent calls from `AugmentedkOmegaSST.C` file. This initialization procedure only needs to be performed once at the beginning of the simulation, which saves a lot of computational time by avoiding the need to reload these parameters at every solver iteration.

Algorithm 5 provides an overview of the algorithm used by `AugmentedkOmegaSST`, specifically for Model Propagation simulations that require the initialization procedure described above. As previously discussed, Baseline and Propagation simulations do not use the `python_model.py` script and therefore do not need to connect to the Python interpreter.

The flow field data that needs to be sent from `AugmentedkOmegaSST` to `python_model.py` to evaluate the model equations for b_{ij}^{Δ} and R is transferred via a dictionary. Dictionaries offer a very efficient, clear, and organized approach to transferring large datasets between two programming languages, especially since they can handle storing different data types. The dictionary created is called `fieldDict`, and it is updated at every solver iteration using a void function called `update_fieldDict()`, defined at the top of the `AugmentedkOmegaSST.C` source file. The `model()` function, defined in the `python_model.py` script, takes this dictionary as an input argument and stores the output in a `pReturn` object, which contains all the model corrections. These corrections, once extracted from this return object, are used to correct the $k-\omega$ SST model equations and the Reynolds-stress tensor at every solver iteration.

Algorithm 5 `AugmentedkOmegaSST` turbulence model algorithm (specifically for Model Propagation simulations)

Input:
Input parameters specified in Table C.1.
Initial conditions of flow fields: U, k, ω, ν_t, p.
Initialization:
1. Start Python interpreter.
2. Load the `python_model.py` file.
3. Load the `model()` function inside `python_model.py`.
Itterative Procedure:
Build `fieldDict` dictionary to pass field data to Python : U, k, ω, τ_{ij}, ∇U, etc (see code in Section C.2).
Update `fieldDict` with `update_fieldDict()` function.
Call Python `model()` function with `fieldDict` as input.
Extract Python `model()` outputs: R, b_{ij}^{Δ}, σ.
Solve k and ω equations given R, b_{ij}^{Δ}, σ.

C.2. Source Code

```
/*---------------------------------------------------------------------------
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Copyright (C) 2016-2018 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------------*/
/*---------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
    ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
    FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
    for more details.

    You should have received a copy of the GNU General Public License
    along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.

-------------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------
 This turbulence model file provides explanations and settings for
    different
 simulations using the Augumented k-Omega SST turbulence model.

 Simulation Types:

 1. Baseline k-Omega SST:

      - Simulation without corrections from high-fidelity data, equivalent
        to
        the OpenFOAM kOmegaSST turbulence model.
```

```
37
38      - To run simulations with this setting, modify the following lines in
39        your turbulenceProperties file in the constant case directory:
40          RASModel        AugmentedkOmegaSST;
41          baseline        true;
42
43    2. Propagation k-Omega SST:
44
45      - k-Omega SST simulation with correction fields obtained from high-
           fidelity
46        data.
47
48      - Corrective fields (bijDelta and kDeficit) should be obtained from a
49        frozen simulation and added to the 0 folder of the case directory.
50
51      - To run a propagation simulation, modify/add the following lines:
52          RASModel        AugmentedkOmegaSST;
53          usebijDelta     true;    // Enable bijDelta corrections.
54          usekDeficit     true;    // Enable kDeficit corrections.
55
56      ---- Optional Settings ----
57          rampStartTime   10;      // Time to start applying corrections.
58          rampEndTime     100;     // Time at which full corrections are
               applied.
59          useSigma        true;    // Use corrections only in classifier
               region.
60          modelSigma      true;    // Compute the 'sigma' classifier field
               during simulation.
61          bijDeltastabilizer 1;    // Factor to stabilize bijDelta
               corrections.
62          kDeficitstabilizer 1;    // Factor to stabilize kDeficit
               corrections.
63
64    3. Model k-Omega SST:
65
66      - k-Omega SST simulation with correction fields obtained from model
           equation
67        files.
68
69      - Model equation fields ('kDeficit' and 'bijDeltaEq') should be text
           files
70        present in your case directory.
71
72      - To run a model simulation, modify/add the following lines:
73          RASModel        AugmentedkOmegaSST;
74          usebijDelta     true;    // Enable bijDelta corrections.
75          usekDeficit     true;    // Enable kDeficit corrections.
76          modelbijDelta   true;    // Enable modeling of bijDelta equation
               from model equation file.
77          modelkDeficit   true;    // Enable modeling of kDeficit equation
               from model equation file.
78
79      ---- Optional Settings ----
80          rampStartTime   10;      // Time to start applying corrections.
81          rampEndTime     100;     // Time at which full corrections are
               applied.
82          useSigma        true;    // Use corrections only in classifier
               region.
```

```
83          modelSigma      true;    // Compute the 'sigma' classifier field
               during simulation.
84          bijDeltastabilizer 1;    // Factor to stabilize bijDelta
               corrections.
85          kDeficitstabilizer 1;    // Factor to stabilize kDeficit
               corrections.
86
87
88    Author : Monica Lacatus.
89    Adapated from: Kaj Hoefnagel, Richard Dwight.
90
91  ----------------------------------------------------------------------*/
92
93 #include "error.H"
94
95 // Check for a Python exception - use after every PyAPI call that could
96 // possibly raise an exception.
97 #define PyErr()  if(PyErr_Occurred()) { \
98                     PyErr_PrintEx(0); \
99                     FatalError << "Python exception" << exit(FatalError); \
100                 }
101
102 /*---------------------------------------------------------------------*/
103
104 namespace Foam
105 {
106 namespace RASModels
107 {
108
109
110 template<class BasicMomentumTransportModel>
111 void AugmentedkOmegaSST<BasicMomentumTransportModel>::update_fieldDict()
112 {
113   // Create a numpy array wrapper around data pointed to by the given
       pointer. The array flags
114   // will have a default that the data area is well-behaved and C-style
       contiguous.
115
116     tmp<volScalarField> p_ = this->U_.db().objectRegistry::lookupObject<
         volScalarField>("p");
117     tmp<volTensorField> tgradU = fvc::grad(this->U_);
118     tmp<volVectorField> tgradp = fvc::grad(p_);
119     tmp<volVectorField> tgradk = fvc::grad(this->k_);
120     tmp<volVectorField> tcurlU = fvc::curl(this->U_);
121
122   {
123     npy_intp dim[] = {num_cells, 3};
124     PyObject *array = PyArray_SimpleNewFromData(2, dim, NPY_DOUBLE, (void
         *)&(this->U_[0]));
125     PyDict_SetItemString(fieldDict, "U", array);
126   }{
127     npy_intp dim[] = {num_cells, 3, 3};
128     PyObject *array = PyArray_SimpleNewFromData(3, dim, NPY_DOUBLE, (void
         *)&(tgradU()[0]));
129     PyDict_SetItemString(fieldDict, "gradU", array);
130   }{
131     npy_intp dim[] = {num_cells, 3};
```

```
132     PyObject *array = PyArray_SimpleNewFromData(2, dim, NPY_DOUBLE, (void
            *)&(tgradk()[0]));
133     PyDict_SetItemString(fieldDict, "gradk", array);
134   }{
135     npy_intp dim[] = {num_cells, 3};
136     PyObject *array = PyArray_SimpleNewFromData(2, dim, NPY_DOUBLE, (void
            *)&(tgradp()[0]));
137     PyDict_SetItemString(fieldDict, "gradp", array);
138   }{
139     npy_intp dim[] = {num_cells};
140     PyObject *array = PyArray_SimpleNewFromData(1, dim, NPY_DOUBLE, (void
            *)&(this->k_[0]));
141     PyDict_SetItemString(fieldDict, "k", array);
142   }{
143     npy_intp dim[] = {num_cells};
144     PyObject *array = PyArray_SimpleNewFromData(1, dim, NPY_DOUBLE, (void
            *)&(this->omega_[0]));
145     PyDict_SetItemString(fieldDict, "omega", array);
146   }{
147     npy_intp dim[] = {num_cells};
148     PyObject *array = PyArray_SimpleNewFromData(1, dim, NPY_DOUBLE, (void
            *)&(this->nut_()[0]));
149     PyDict_SetItemString(fieldDict, "nut", array);
150   }{
151     npy_intp dim[] = {num_cells};
152     PyObject *array = PyArray_SimpleNewFromData(1, dim, NPY_DOUBLE, (void
            *)&(this->y_[0]));
153     PyDict_SetItemString(fieldDict, "walldist", array);
154   }{
155     npy_intp dim[] = {num_cells};
156     PyObject *array = PyArray_SimpleNewFromData(1, dim, NPY_DOUBLE,(void *)
            &(this->nu()().internalField()[0]));
157     PyDict_SetItemString(fieldDict, "nu", array);
158   }{
159     npy_intp dim[] = {num_cells, 3};
160     PyObject *array = PyArray_SimpleNewFromData(2, dim, NPY_DOUBLE, (void
            *)&(tcurlU()[0]));
161     PyDict_SetItemString(fieldDict, "curlU", array);
162   }{
163     npy_intp dim[] = {num_cells, 3};
164     PyObject *array = PyArray_SimpleNewFromData(2, dim, NPY_DOUBLE, (void
            *)&(this->mesh_.C()[0]));
165     PyDict_SetItemString(fieldDict, "C", array);
166   }{
167     npy_intp dim[] = {num_cells};
168     PyObject *array = PyArray_SimpleNewFromData(1, dim, NPY_DOUBLE, (void
            *)&(this->Pk_()[0]));
169     PyDict_SetItemString(fieldDict, "Pk", array);
170   }{
171     npy_intp dim[] = {num_cells};
172     PyObject *array = PyArray_SimpleNewFromData(1, dim, NPY_DOUBLE, (void
            *)&(this->Pk_prop_()[0]));
173     PyDict_SetItemString(fieldDict, "Pk_prop", array);
174   }{
175     npy_intp dim[] = {num_cells};
176     PyObject *array = PyArray_SimpleNewFromData(1, dim, NPY_DOUBLE, (void
            *)&(this->Dk_()[0]));
177     PyDict_SetItemString(fieldDict, "Dk", array);
```

```
178   }{
179     npy_intp dim[] = {num_cells};
180     PyObject *array = PyArray_SimpleNewFromData(1, dim, NPY_DOUBLE, (void
          *)&(this->k_conv_()[0]));
181     PyDict_SetItemString(fieldDict, "k_conv", array);
182   }{
183     npy_intp dim[] = {num_cells};
184     PyObject *array = PyArray_SimpleNewFromData(1, dim, NPY_DOUBLE, (void
          *)&(this->k_diff_()[0]));
185     PyDict_SetItemString(fieldDict, "k_diff", array);
186   }{
187     npy_intp dim[] = {num_cells, 3, 3};
188     PyObject *array = PyArray_SimpleNewFromData(3, dim, NPY_DOUBLE, (void
          *)&(this->tauij_B_()[0]));
189     PyDict_SetItemString(fieldDict, "tauij_B", array);
190   };
191
192     p_.clear();
193     tgradU.clear();
194     tgradp.clear();
195     tgradk.clear();
196     tcurlU.clear();
197 }
198
199 /*---------- Protected Member Functions ---------*/
200
201 template<class BasicTurbulenceModel>
202 tmp<volScalarField>
203 AugmentedkOmegaSST<BasicTurbulenceModel>::F1
204 (
205     const volScalarField& CDkOmega
206 ) const
207 {
208     tmp<volScalarField> CDkOmegaPlus = max
209     (
210         CDkOmega,
211         dimensionedScalar(dimless/sqr(dimTime), 1.0e-10)
212     );
213
214     tmp<volScalarField> arg1 = min
215     (
216         min
217         (
218             max
219             (
220                 (scalar(1)/betaStar_)*sqrt(k_)/(omega_*y_),
221                 scalar(500)*(this->mu()/this->rho_)/(sqr(y_)*omega_)
222             ),
223             (4*alphaOmega2_)*k_/(CDkOmegaPlus*sqr(y_))
224         ),
225         scalar(10)
226     );
227
228     return tanh(pow4(arg1));
229 }
230
231 template<class BasicTurbulenceModel>
232 tmp<volScalarField>
```

```
AugmentedkOmegaSST<BasicTurbulenceModel>::F2() const
{
    tmp<volScalarField> arg2 = min
    (
        max
        (
            (scalar(2)/betaStar_)*sqrt(k_)/(omega_*y_),
            scalar(500)*(this->mu()/this->rho_)/(sqr(y_)*omega_)
        ),
        scalar(100)
    );

    return tanh(sqr(arg2));
}

template<class BasicTurbulenceModel>
tmp<volScalarField>
AugmentedkOmegaSST<BasicTurbulenceModel>::F3() const
{
    tmp<volScalarField> arg3 = min
    (
        150*(this->mu()/this->rho_)/(omega_*sqr(y_)),
        scalar(10)
    );

    return 1 - tanh(pow4(arg3));
}

template<class BasicTurbulenceModel>
tmp<volScalarField>
AugmentedkOmegaSST<BasicTurbulenceModel>::F23() const
{
    tmp<volScalarField> f23(F2());

    if (F3_)
    {
        f23.ref() *= F3();
    }

    return f23;
}


template<class BasicTurbulenceModel>
void AugmentedkOmegaSST<BasicTurbulenceModel>::correctNut
(
    const volScalarField& S2,
    const volScalarField& F2
)
{
    this->nut_ = a1_*k_/max(a1_*omega_, b1_*F2*sqrt(S2));
    this->nut_.correctBoundaryConditions();
    fv::options::New(this->mesh_).correct(this->nut_);

    BasicTurbulenceModel::correctNut();
}

template<class BasicTurbulenceModel>
```

```
291 void AugmentedkOmegaSST<BasicTurbulenceModel>::correctNut()
292 {
293     correctNut(2*magSqr(symm(fvc::grad(this->U_))), F23());
294 }
295
296
297 template<class BasicTurbulenceModel>
298 tmp<volScalarField::Internal>
299 AugmentedkOmegaSST<BasicTurbulenceModel>::Pk
300 (
301     const volScalarField::Internal& G
302 ) const
303 {
304     return min(G, (c1_*betaStar_)*this->k_()*this->omega_());
305 }
306
307
308 template<class BasicTurbulenceModel>
309 tmp<volScalarField::Internal>
310 AugmentedkOmegaSST<BasicTurbulenceModel>::epsilonByk
311 (
312     const volScalarField::Internal& F1,
313     const volScalarField::Internal& F2
314 ) const
315 {
316     return betaStar_*omega_();
317 }
318
319
320 template<class BasicTurbulenceModel>
321 tmp<fvScalarMatrix>
322 AugmentedkOmegaSST<BasicTurbulenceModel>::kSource() const
323 {
324     return tmp<fvScalarMatrix>
325     (
326         new fvScalarMatrix
327         (
328             k_,
329             dimVolume*this->rho_.dimensions()*k_.dimensions()/dimTime
330         )
331     );
332 }
333
334
335 template<class BasicTurbulenceModel>
336 tmp<fvScalarMatrix>
337 AugmentedkOmegaSST<BasicTurbulenceModel>::omegaSource() const
338 {
339     return tmp<fvScalarMatrix>
340     (
341         new fvScalarMatrix
342         (
343             omega_,
344             dimVolume*this->rho_.dimensions()*omega_.dimensions()/dimTime
345         )
346     );
347 }
348
```

```
template<class BasicTurbulenceModel>
tmp<fvScalarMatrix> AugmentedkOmegaSST<BasicTurbulenceModel>::Qsas
(
    const volScalarField::Internal& S2,
    const volScalarField::Internal& gamma,
    const volScalarField::Internal& beta
) const
{
    return tmp<fvScalarMatrix>
    (
        new fvScalarMatrix
        (
            omega_,
            dimVolume*this->rho_.dimensions()*omega_.dimensions()/dimTime
        )
    );
}


/*----Updating Reynolds-Stress Source Term For Momentum Equation----*/

// divDevReff function
template<class BasicTurbulenceModel>
tmp<fvVectorMatrix> AugmentedkOmegaSST<BasicTurbulenceModel>::divDevReff
(
    volVectorField& U
) const
{
    Info << "In: AugmentedkOmegaSST::divDevReff()" << endl;
    return
    (
      // Boussinesq part
      - fvc::div((this->alpha_*this->rho_*this->nuEff())*dev2(T(fvc::grad(U
          ))))
      - fvm::laplacian(this->alpha_*this->rho_*this->nuEff(), U)

      // Nonlinear correction part
      + this->sigma_ * fvc::div(dev(2.*this->k_*this->bijDelta_) *
      bijDeltastabilizer_ * xi_)
    );
}


// devRhoReff function
template<class BasicTurbulenceModel>
Foam::tmp<Foam::volSymmTensorField>
AugmentedkOmegaSST<BasicTurbulenceModel>::devRhoReff() const
{
    Info << "In: AugmentedkOmegaSST::devRhoReff()" << endl;
    return volSymmTensorField::New
    (
        // Boussinesq part
        IOobject::groupName("devRhoReff", this->alphaRhoPhi_.group()),
        (-(this->alpha_*this->rho_*this->nuEff()))
        *dev(twoSymm(fvc::grad(this->U_)))

        // Nonlinear correction part
```

```
405         + this->sigma_ * dev(2.*this->k_*this->bijDelta_) *
                bijDeltastabilizer_ * xi_
406     );
407 }
408
409 // divDevRhoReff function
410 template<class BasicTurbulenceModel>
411 Foam::tmp<Foam::fvVectorMatrix>
412 AugmentedkOmegaSST<BasicTurbulenceModel>::divDevRhoReff
413 (
414     volVectorField& U
415 ) const
416 {
417     Info << "In: AugmentedkOmegaSST::divDevRhoReff()" << endl;
418     return
419     (
420       // Boussinesq part
421       - fvc::div((this->alpha_*this->rho_*this->nuEff())*dev2(T(fvc::grad(U
              ))))
422       - fvm::laplacian(this->alpha_*this->rho_*this->nuEff(), U)
423
424       // Nonlinear correction part
425       + this->sigma_ * fvc::div(dev(2.*this->k_*this->bijDelta_) *
426       bijDeltastabilizer_ * xi_)
427     );
428 }
429
430 // divDevRhoReff function (different input template)
431 template<class BasicTurbulenceModel>
432 Foam::tmp<Foam::fvVectorMatrix>
433 AugmentedkOmegaSST<BasicTurbulenceModel>::divDevRhoReff
434 (
435     const volScalarField& rho,
436     volVectorField& U
437 ) const
438 {
439     Info << "In: AugmentedkOmegaSST::divDevRhoReff()" << endl;
440     return
441     (
442       // Boussinesq part
443       - fvc::div((this->alpha_*rho*this->nuEff())*dev2(T(fvc::grad(U))))
444       - fvm::laplacian(this->alpha_*rho*this->nuEff(), U)
445
446       // Nonlinear correction part
447       + this->sigma_ * fvc::div(dev(2.*this->k_*this->bijDelta_) *
448       bijDeltastabilizer_ * xi_)
449     );
450 }
451
452 /*-----------------Constructors-----------------*/
453
454 template<class BasicTurbulenceModel>
455 AugmentedkOmegaSST<BasicTurbulenceModel>::AugmentedkOmegaSST
456 (
457     const alphaField& alpha,
458     const rhoField& rho,
459     const volVectorField& U,
460     const surfaceScalarField& alphaRhoPhi,
```

```
461     const surfaceScalarField& phi,
462     const transportModel& transport,
463     const word& propertiesName,
464     const word& type
465 )
466 :
467     eddyViscosity<RASModel<BasicTurbulenceModel>>
468     (
469         type,
470         alpha,
471         rho,
472         U,
473         alphaRhoPhi,
474         phi,
475         transport,
476         propertiesName
477     ),
478 
479 
480     /*------k-Omega SST Model Coefficients------*/
481 
482     alphaK1_
483     (
484         dimensioned<scalar>::lookupOrAddToDict
485         (
486             "alphaK1",
487             this->coeffDict_,
488             0.85
489         )
490     ),
491     alphaK2_
492     (
493         dimensioned<scalar>::lookupOrAddToDict
494         (
495             "alphaK2",
496             this->coeffDict_,
497             1.0
498         )
499     ),
500     alphaOmega1_
501     (
502         dimensioned<scalar>::lookupOrAddToDict
503         (
504             "alphaOmega1",
505             this->coeffDict_,
506             0.5
507         )
508     ),
509     alphaOmega2_
510     (
511         dimensioned<scalar>::lookupOrAddToDict
512         (
513             "alphaOmega2",
514             this->coeffDict_,
515             0.856
516         )
517     ),
518     gamma1_
```

```
519     (
520         dimensioned<scalar>::lookupOrAddToDict
521         (
522             "gamma1",
523             this->coeffDict_,
524             5.0/9.0
525         )
526     ),
527     gamma2_
528     (
529         dimensioned<scalar>::lookupOrAddToDict
530         (
531             "gamma2",
532             this->coeffDict_,
533             0.44
534         )
535     ),
536     beta1_
537     (
538         dimensioned<scalar>::lookupOrAddToDict
539         (
540             "beta1",
541             this->coeffDict_,
542             0.075
543         )
544     ),
545     beta2_
546     (
547         dimensioned<scalar>::lookupOrAddToDict
548         (
549             "beta2",
550             this->coeffDict_,
551             0.0828
552         )
553     ),
554     betaStar_
555     (
556         dimensioned<scalar>::lookupOrAddToDict
557         (
558             "betaStar",
559             this->coeffDict_,
560             0.09
561         )
562     ),
563     a1_
564     (
565         dimensioned<scalar>::lookupOrAddToDict
566         (
567             "a1",
568             this->coeffDict_,
569             0.31
570         )
571     ),
572     b1_
573     (
574         dimensioned<scalar>::lookupOrAddToDict
575         (
576             "b1",
```

```
            this->coeffDict_,
            1.0
        )
    ),
    c1_
    (
        dimensioned<scalar>::lookupOrAddToDict
        (
            "c1",
            this->coeffDict_,
            10.0
        )
    ),
    F3_
    (
        Switch::lookupOrAddToDict
        (
            "F3",
            this->coeffDict_,
            false
        )
    ),


    /*********************** General Flow Features *********************/

    y_(
       IOobject
       (
            "walldist",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
       ),
       wallDist::New(this->mesh_).y()
    ),


    k_
    (
        IOobject
        (
            IOobject::groupName("k", alphaRhoPhi.group()),
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        this->mesh_
    ),

    omega_
    (
        IOobject
        (
            IOobject::groupName("omega", alphaRhoPhi.group()),
            this->runTime_.timeName(),
```

```
635             this->mesh_,
636             IOobject::MUST_READ,
637             IOobject::AUTO_WRITE
638         ),
639         this->mesh_
640     ),
641 
642     gradU_
643     (
644         IOobject
645         (
646             "gradU",
647             this->runTime_.timeName(),
648             this->mesh_,
649             IOobject::NO_READ,
650             IOobject::AUTO_WRITE
651         ),
652         fvc::grad(this->U_)
653     ),
654 
655     tauij_B_
656     (
657         IOobject
658         (
659             "tauij_B",
660             this->runTime_.timeName(),
661             this->mesh_,
662             IOobject::NO_READ,
663             IOobject::AUTO_WRITE
664         ),
665         -this->nut_ * twoSymm(fvc::grad(this->U_)) + ((2.0/3.0)*I)*this->k_
666     ),
667 
668     Pk_
669     (
670         IOobject
671         (
672             "Pk",
673             this->runTime_.timeName(),
674             this->mesh_,
675             IOobject::NO_READ,
676             IOobject::AUTO_WRITE
677         ),
678         this->mesh_,
679         dimensionedScalar("Pk", dimensionSet(0,2,-3,0,0,0,0), 0.0)
680     ),
681 
682 
683     Pk_prop_
684     (
685         IOobject
686         (
687             "Pk_prop",
688             this->runTime_.timeName(),
689             this->mesh_,
690             IOobject::NO_READ,
691             IOobject::AUTO_WRITE
692         ),
```

```
693         this->mesh_,
694         dimensionedScalar("Pk_prop", dimensionSet(0,2,-3,0,0,0,0), 0.0)
695     ),
696
697     Dk_
698     (
699         IOobject
700         (
701             "Dk",
702             this->runTime_.timeName(),
703             this->mesh_,
704             IOobject::NO_READ,
705             IOobject::AUTO_WRITE
706         ),
707         this->mesh_,
708         dimensionedScalar("Dk", dimensionSet(0,2,-3,0,0,0,0), 0.0)
709     ),
710
711     k_conv_
712     (
713         IOobject
714         (
715             "k_conv",
716             this->runTime_.timeName(),
717             this->mesh_,
718             IOobject::NO_READ,
719             IOobject::AUTO_WRITE
720         ),
721         this->mesh_,
722         dimensionedScalar("k_conv", dimensionSet(0,2,-3,0,0,0,0), 0.0)
723     ),
724
725     k_diff_
726     (
727         IOobject
728         (
729             "k_diff",
730             this->runTime_.timeName(),
731             this->mesh_,
732             IOobject::NO_READ,
733             IOobject::AUTO_WRITE
734         ),
735         this->mesh_,
736         dimensionedScalar("k_diff", dimensionSet(0,2,-3,0,0,0,0), 0.0)
737     ),
738
739
740    /*---------------Simulation Options---------------*/
741
742     // Boolean to decide whether to perform a baseline kOmegaSST
        simualation.
743     // Deafault is false.
744
745     baseline_
746     (
747         Switch::lookupOrAddToDict
748         (
749             "baseline",
```

```
750             this->coeffDict_,
751             false
752         )
753     ),
754
755     // Boolean to decide whether to use kDeifict correction or not.
756     // Default is false such that no corrections are used.
757
758     usekDeficit_
759     (
760         Switch::lookupOrAddToDict
761         (
762             "usekDeficit",
763             this->coeffDict_,
764             false
765         )
766     ),
767
768     // Boolean to decide whether to use bijDelta correction or not.
769     // Default is false such that no corrections are used.
770
771     usebijDelta_
772     (
773         Switch::lookupOrAddToDict
774         (
775             "usebijDelta",
776             this->coeffDict_,
777             false
778         )
779     ),
780
781     // Boolean to decide whether to use a classifier or not.
782     // Default is false such that no classifier is used.
783     useSigma_
784     (
785         Switch::lookupOrAddToDict
786         (
787             "useSigma",
788             this->coeffDict_,
789             false
790         )
791     ),
792
793     // Switches to decide whether to use a model for a correction or frozen
794     // fields in the 0 directory. If the model switch is true, a model is
        used
795     // and the {var}Eq file (e.g. bijDeltaEq) should be present in the case
796     // directory with the model equation to use. If the model switch is
        false,
797     // the frozen field should be present in the zero directory. Default is
         to
798     // use fields for all corrections/classifier.
799
800     modelbijDelta_
801     (
802         Switch::lookupOrAddToDict
803         (
804             "modelbijDelta",
```

```
805             this->coeffDict_,
806             false
807         )
808     ),
809
810     modelkDeficit_
811     (
812         Switch::lookupOrAddToDict
813         (
814             "modelkDeficit",
815             this->coeffDict_,
816             false
817         )
818     ),
819
820
821     modelSigma_
822     (
823         Switch::lookupOrAddToDict
824         (
825             "modelSigma",
826             this->coeffDict_,
827             false
828         )
829     ),
830
831
832     // Scalar by which the exact/modeled bijDelta correction is multiplied,
833     // typically in the [0,1] range, mostly used for stabilization.
834
835     bijDeltastabilizer_
836     (
837         dimensioned<scalar>::lookupOrAddToDict
838         (
839             "bijDeltastabilizer",
840             this->coeffDict_,
841             1.0
842         )
843     ),
844
845     // Scalar by which the exact/modeled kDeficit correction is multiplied,
846     // typically in the [0,1] range, mostly used for stabilization.
847
848     kDeficitstabilizer_
849     (
850         dimensioned<scalar>::lookupOrAddToDict
851         (
852             "kDeficitstabilizer",
853             this->coeffDict_,
854             1.0
855         )
856     ),
857
858     // Ramping gradually introduce corrections (both kDeficit and bijDelta)
859     // to aid solver stability.  Before `rampStartTime` corrections are
        zero,
860     // after `rampEndTime` they are 1.0.  Linear in between.
861
```

```
862     rampStartTime_
863     (
864         dimensioned<scalar>::lookupOrAddToDict
865         (
866             "rampStartTime",
867             this->coeffDict_,
868                 dimTime,
869             0
870         )
871     ),
872     rampEndTime_
873     (
874         dimensioned<scalar>::lookupOrAddToDict
875         (
876             "rampEndTime",
877             this->coeffDict_,
878                 dimTime,
879             100
880         )
881     ),
882 
883     // Corrections are multiplied by xi_ to apply ramping; xi_ is 0 before
884     // rampStartTime, it linearly goes to 1 between rampStartTime and
885     // rampEndTime and it stays 1 after rampEndTime. At each iteration,
886     // xi_ is calculated based on the specified rampStartTime and
        rampEndTime.
887     xi_
888     (
889         dimensioned<scalar>::lookupOrAddToDict
890         (
891             "xi_ramp",
892             this->coeffDict_,
893                 dimless,
894             1
895         )
896     ),
897 
898     /*---Correction/Classifier Fields to be Read/Modelled---*/
899 
900     // The correction/classifier fields are initialized at a bit-specific
        small
901     // value. The READ_IF_PRESENT directive is used to overwrite this value
902     // with whatever is read in from the zero directory. If model{var}_ is
903     // true, it is later checked whether the field was successfully read in
         by
904     // checking whether the field still has the specific initialized value.
905     // If model{var} is false, the initialized value is overwritten by
906     // whatever is calculated based on the model equation.
907 
908     kDeficit_
909     (
910         IOobject(
911             "kDeficit",
912             this->runTime_.timeName(),
913             this->mesh_,
914             IOobject::READ_IF_PRESENT,
915             IOobject::AUTO_WRITE
916         ),
```

```
        this->mesh_,
        dimensionedScalar
        (
            "kDeficit",
            dimensionSet(0,2,-3,0,0,0,0),
            1.20813608515e-37
        )
    ),

    bijDelta_
    (
        IOobject
        (
            "bijDelta",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::READ_IF_PRESENT,
            IOobject::AUTO_WRITE
        ),
        this->mesh_,
        dimensionedSymmTensor
        (
            "bijDelta",
            dimensionSet(0,0,0,0,0,0,0),
            symmTensor(1.20813608515e-37,0,0,0,0,0)
        )
    ),

    sigma_
    (
        IOobject(
            "sigma",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::READ_IF_PRESENT,
            IOobject::AUTO_WRITE
        ),
        this->mesh_,
        dimensionedScalar
        (
            "sigma",
            dimensionSet(0,0,0,0,0,0,0),
            1.20813608515e-37
        )
    )


{
    if (type == typeName)
    {
        this->printCoeffs(type);
    }
    bound(k_, this->kMin_);
    bound(omega_, this->omegaMin_);

    // Only when Propagation or Model Propagation simulations with modeled
       sigma is enabled, the python interpreter is initialized.
    if ((modelSigma_ || modelbijDelta_ || modelkDeficit_) && !baseline_){

```

```
974         // Start up the Python interpreter
975         Py_Initialize();
976         PyErr();
977 
978         // Add the run directory to the python path
979         PyRun_SimpleString("import sys");
980         PyErr();
981         PyRun_SimpleString("sys.path.append(\".\")");
982         PyErr();
983 
984         // Initializes numpy array library, allowing the C++ code to
                interact with NumPy arrays.
985         import_array1();
986         PyErr();
987 
988         // Load the file python_model.py which should be in the case
                directory
989         PyObject *pName = PyUnicode_DecodeFSDefault("python_model");
990         PyErr();
991         PyObject *pModule = PyImport_Import(pName);
992         PyErr();
993 
994         // Load the model function inside python_model.py
995         model = PyObject_GetAttrString(pModule, "model");
996         PyErr();}
997 
998 }
999 
1000 
1001 /*----------------Member Functions----------------*/
1002 
1003 
1004 template<class BasicTurbulenceModel>
1005 bool AugmentedkOmegaSST<BasicTurbulenceModel>::read()
1006 {
1007     if (eddyViscosity<RASModel<BasicTurbulenceModel>>::read())
1008     {
1009         alphaK1_.readIfPresent(this->coeffDict());
1010         alphaK2_.readIfPresent(this->coeffDict());
1011         alphaOmega1_.readIfPresent(this->coeffDict());
1012         alphaOmega2_.readIfPresent(this->coeffDict());
1013         gamma1_.readIfPresent(this->coeffDict());
1014         gamma2_.readIfPresent(this->coeffDict());
1015         beta1_.readIfPresent(this->coeffDict());
1016         beta2_.readIfPresent(this->coeffDict());
1017         betaStar_.readIfPresent(this->coeffDict());
1018         a1_.readIfPresent(this->coeffDict());
1019         b1_.readIfPresent(this->coeffDict());
1020         c1_.readIfPresent(this->coeffDict());
1021         F3_.readIfPresent("F3", this->coeffDict());
1022 
1023         return true;
1024     }
1025     else
1026     {
1027         return false;
1028     }
1029 }
```

```
1030
1031
1032 template<class BasicTurbulenceModel>
1033 void AugmentedkOmegaSST<BasicTurbulenceModel>::correct()
1034 {
1035     if (!this->turbulence_)
1036     {
1037         return;
1038     }
1039
1040     // Local references
1041     const alphaField& alpha = this->alpha_;
1042     const rhoField& rho = this->rho_;
1043     const surfaceScalarField& alphaRhoPhi = this->alphaRhoPhi_;
1044     const volVectorField& U = this->U_;
1045     const volScalarField& nut = this->nut_;
1046     fv::options& fvOptions(fv::options::New(this->mesh_));
1047
1048     BasicTurbulenceModel::correct();
1049
1050     volScalarField::Internal divU
1051     (
1052         fvc::div(fvc::absolute(this->phi(), U))()()
1053     );
1054
1055     tmp<volTensorField> tgradU = fvc::grad(U);
1056     volScalarField S2(2*magSqr(symm(tgradU())));
1057     volScalarField GbyNu(dev(twoSymm(tgradU())) && tgradU());
1058     volScalarField::Internal G(this->GName(), nut()*GbyNu);
1059
1060     volScalarField CDkOmega
1061     (
1062         (2*alphaOmega2_)*(fvc::grad(k_) & fvc::grad(omega_))/omega_
1063     );
1064
1065     volScalarField F1(this->F1(CDkOmega));
1066     volScalarField F23(this->F23());
1067
1068     volScalarField G2_new
1069     (
1070          "G2_new",
1071            nut*GbyNu - xi_ * bijDeltastabilizer_ * sigma_ * (2*(this->k_)*
                   bijDelta_ && tgradU())
1072     );
1073
1074     // Reynolds stress tensor (computed under Boussinesq)
1075     tauij_B_.ref() =  -this->nut_ * twoSymm(tgradU()) + ((2.0/3.0)*I)*this
        ->k_ + 2*this->k_* xi_*bijDeltastabilizer_ *sigma_*bijDelta_;
1076
1077     // Production term for k equation (bijDelta correction)
1078     Pk_.ref() = alpha()*rho()*this->Pk(G2_new)-(2.0/3.0)*alpha()*rho()*divU
        *k_();
1079
1080     // Production term for k equation (kdeficit corection)
1081     Pk_prop_.ref()= alpha()*rho()*sigma_*kDeficit_()*(xi_ *
        kDeficitstabilizer_);
1082
1083     // Destruction term of k equation
```

```
    Dk_.ref() = (alpha()*rho()*this->epsilonByk(F1, F23)*k_())/rho();

    // Convection term of k equation
    k_conv_.ref() = (fvc::div(alphaRhoPhi, k_)())/rho();

    // Diffusion term of k equation
    k_diff_.ref() = (fvc::laplacian(alpha*rho*this->DkEff(F1), k_)())/rho
        ();


    // Turning corrections off if baseline model is used, by setting all
        correction fields to 0.
    if (baseline_){

        Info << " Running kOmegaSST Baseline Simulation! " << endl;

        forAll(sigma_.internalField(), id)
        {
            sigma_[id] = 0.0;
        }

        forAll(bijDelta_.internalField(), id)
        {
            bijDelta_[id][0] = 0.0;
            bijDelta_[id][1] = 0.0;
            bijDelta_[id][2] = 0.0;
            bijDelta_[id][3] = 0.0;
            bijDelta_[id][4] = 0.0;
            bijDelta_[id][5] = 0.0;
        }

        forAll(kDeficit_.internalField(), id)
        {
            kDeficit_[id] = 0.0;
        }

    } else {

        // Computing correction factor from rampTime option.
        const dimensionedScalar time = this->runTime_;
        xi_ = (time < rampStartTime_)? 0.0:
            (time > rampEndTime_)? 1.0:
            (time - rampStartTime_) / (rampEndTime_ - rampStartTime_);
        Info << "Corrections: xi = " << xi_.value() <<
            ", kDeficit factor = " << (xi_*kDeficitstabilizer_).value() <<
            ", bijDelta factor = " << (xi_*bijDeltastabilizer_).value() <<
                endl;

        // Setting up fields for propagation simulations.
        if (!modelkDeficit_ && !modelbijDelta_){

            Info << "Running Propagation Simulation: " << endl;

            if (usekDeficit_){

                if (kDeficit_[0] == 1.20813608515e-37){
```

```
1138                    FatalError << "modelkDeficit set to false, but no
                            kDeficit file found in 0 folder."
1139                    << nl << exit(FatalError);
1140
1141                } else {
1142
1143                    Info << "kDeficit corrections are being read from file.
                             " << endl;
1144                }
1145
1146            } else {
1147
1148                Info << "kDeifict corrections are not being used. " << endl
                        ;
1149
1150                forAll(kDeficit_.internalField(), id){
1151
1152                kDeficit_[id] = 0.0;
1153
1154                }
1155            }
1156
1157            if (usebijDelta_){
1158
1159                if (bijDelta_[0][0] == 1.20813608515e-37){
1160
1161                    FatalError << "modelbijDelta set to false, but no
                            bijDelta file found in the 0 folder."
1162                    << nl << exit(FatalError);
1163
1164                } else {
1165
1166                    Info << "bijDelta corrections are being read from file.
                            " << endl;
1167                }
1168
1169            } else {
1170
1171                Info << "bijDelta corrections are not being used. " << endl
                        ;
1172
1173                forAll(bijDelta_.internalField(), id){
1174
1175                    bijDelta_[id][0] = 0.0;
1176                    bijDelta_[id][1] = 0.0;
1177                    bijDelta_[id][2] = 0.0;
1178                    bijDelta_[id][3] = 0.0;
1179                    bijDelta_[id][4] = 0.0;
1180                    bijDelta_[id][5] = 0.0;
1181                }
1182
1183            }
1184
1185             if (useSigma_){
1186
1187                if (modelSigma_){
1188
```

```
1189                    Info << "Sigma Classifier is being updated dynamically.
                            " << endl;
1190
1191                    // Set num_cells equal to the number of cells in the
                            mesh
1192                    num_cells = this->mesh_.cells().size();
1193
1194                    // Build dictionary to pass field-data to Python
1195                    fieldDict = PyDict_New();
1196                    PyErr();
1197
1198                    // Update dictionary with field data.
1199                    update_fieldDict();
1200
1201                    // Call the user-defined Python function - giving
                            fields and arguments,
1202                    // which should return model corrections.
1203                    PyObject* pReturn = PyObject_CallObject(model,
                            PyTuple_Pack(1, fieldDict));
1204                    PyErr();
1205
1206                    // Extract the sigma result from the python model
                            return object.
1207                    PyArrayObject* sigma_return = (PyArrayObject *)
                            PyDict_GetItemString(pReturn, "sigma");
1208                    PyErr();
1209
1210                    // Check if the sigma result is actually returned by
                            python and is not NULL
1211                    if(sigma_return == NULL) FatalError << "'sigma' not in
                            return value of python model()"<< exit(FatalError);
1212
1213                    // Check if the returned array has the expected
                            dimensions.
1214                    if (PyArray_DIM(sigma_return, 0) != num_cells)
1215                    FatalError << "Dimension of python model sigma
                            corrections are wrong != " << num_cells << exit(
                            FatalError);
1216
1217                    // Extract sigma return values from python model return
                             object.
1218                    forAll(sigma_.internalField(), id){
1219
1220                        sigma_[id] = *((double*)PyArray_GETPTR2(
                                sigma_return, id,0));
1221                        PyErr();
1222                    }
1223
1224                    // Delete references to created objects to free up
                            memory.
1225                    Py_XDECREF(fieldDict);
1226                    PyErr();
1227                    Py_DECREF(pReturn);
1228                    PyErr();
1229
1230                } else if (sigma_[0] == 1.20813608515e-37){
1231
```

```
1232                    FatalError << "modelSigma set to false, but no sigma
                            file found in the 0 folder."
1233                    << nl << exit(FatalError);
1234
1235                } else {
1236
1237                    Info << "Sigma classifier is being read from file." <<
                            endl;
1238                }
1239            } else {
1240
1241                Info << "Sigma classifier is not being used. " << endl;
1242
1243                forAll(sigma_.internalField(), id){
1244
1245                    sigma_[id] = 1.;
1246                }
1247            }
1248
1249        } else {
1250
1251            Info << "Running Model Propagation Simulation: " << endl;
1252
1253            // Set num_cells equal to the number of cells in the mesh.
1254            num_cells = this->mesh_.cells().size();
1255
1256            // Build dictionary to pass field-data to Python.
1257            fieldDict = PyDict_New();
1258            PyErr();
1259
1260            // Update dictionary with field data.
1261            update_fieldDict();
1262
1263            // Call the user-defined Python function - giving fields and
                    arguments,
1264            // which should return model corrections.
1265            PyObject* pReturn = PyObject_CallObject(model, PyTuple_Pack(1,
                    fieldDict));
1266            PyErr();
1267
1268            if (usekDeficit_) {
1269
1270                if (modelkDeficit_){
1271
1272                    Info << "kDeficit corrrections ar being modeled from
                            kDeficitEq. " << endl;
1273
1274                    // Extract the kDeificit result from the python model
                            return object.
1275                    PyArrayObject* kDeficit_return = (PyArrayObject *)
                            PyDict_GetItemString(pReturn, "kDeficit");
1276                    PyErr();
1277
1278                    // Check if the kDeificit result is actually returned
                            by python and is not NULL.
1279                    if(kDeficit_return == NULL) FatalError << "'kDeficit'
                            not in return value of python model()"<< exit(
                            FatalError);
```

```
1280
1281                    // Check if the returned array has the expected
                            dimensions.
1282                    if (PyArray_DIM(kDeficit_return, 0) != num_cells)
1283                    FatalError << "Dimension of python model kDeifict
                            corrections are wrong != " << num_cells << exit(
                            FatalError);
1284
1285                    forAll(kDeficit_.internalField(), id){
1286
1287                        // Extract kdeficit return values from python model
                                return object.
1288                        kDeficit_[id] = *((double*)PyArray_GETPTR2(
                                kDeficit_return, id, 0));
1289                        PyErr();
1290
1291                    }
1292
1293                } else if (kDeficit_[0] == 1.20813608515e-37) {
1294
1295                    FatalError << "modelkDeficit set to false, but no
                            kDeficit file found in 0 folder."
1296                    << nl << exit(FatalError);
1297
1298                } else {
1299
1300                    Info << "kDeficit corrections are being read from file.
                            " << endl;
1301                }
1302
1303            } else {
1304
1305                Info << "kDeficit corrections are not being used." << endl;
1306
1307                forAll(kDeficit_.internalField(), id){
1308
1309                    kDeficit_[id] = 0.0;
1310                }
1311            }
1312
1313            if (usebijDelta_) {
1314
1315                if (modelbijDelta_){
1316
1317                    Info << "bijDelta corrections are being modeled from
                            bijDeltaEq. " << endl;
1318
1319                    // Extract the bijDelta result from the python model
                            return object.
1320                    PyArrayObject* bijDelta_return = (PyArrayObject *)
                            PyDict_GetItemString(pReturn, "bijDelta");
1321                    PyErr();
1322
1323                    // Check if the bijDelta result is actually returned by
                             python and is not NULL.
1324                    if(bijDelta_return == NULL) FatalError << "'bijDelta'
                            not in return value of python model()"<< exit(
                            FatalError);
```

```
1325
1326                    // Check if the returned array has the expected
                            dimensions.
1327                    if (PyArray_DIM(bijDelta_return, 0) != num_cells)
1328                    FatalError << "Dimension of python model bijDelta
                            corrections are wrong != " << num_cells << exit(
                            FatalError);
1329
1330                    // Extract bijDelta return values from python model
                            return object.
1331                    forAll(bijDelta_.internalField(), id){
1332
1333                        bijDelta_[id][0] = *((double*)PyArray_GETPTR2(
                                bijDelta_return, id, 0));
1334                        bijDelta_[id][1] = *((double*)PyArray_GETPTR2(
                                bijDelta_return, id, 1));
1335                        bijDelta_[id][2] = *((double*)PyArray_GETPTR2(
                                bijDelta_return, id, 2));
1336                        bijDelta_[id][3] = *((double*)PyArray_GETPTR2(
                                bijDelta_return, id, 3));
1337                        bijDelta_[id][4] = *((double*)PyArray_GETPTR2(
                                bijDelta_return, id, 4));
1338                        bijDelta_[id][5] = *((double*)PyArray_GETPTR2(
                                bijDelta_return, id, 5));
1339                        PyErr();
1340
1341                    }
1342
1343                } else if (bijDelta_[0][0] == 1.20813608515e-37){
1344
1345                    FatalError << "modelbijDelta set to false, but no
                            bijDelta file found in the 0 folder."
1346                    << nl << exit(FatalError);
1347
1348                } else {
1349
1350                    Info << "bijDelta corrections are being read from file.
                            " << endl;
1351                }
1352
1353            } else {
1354
1355                Info << "bijDelta corrections are not being used." << endl;
1356
1357                forAll(bijDelta_.internalField(), id){
1358
1359                    bijDelta_[id][0] = 0.;
1360                    bijDelta_[id][1] = 0.;
1361                    bijDelta_[id][2] = 0.;
1362                    bijDelta_[id][3] = 0.;
1363                    bijDelta_[id][4] = 0.;
1364                    bijDelta_[id][5] = 0.;
1365                }
1366            }
1367
1368            if (useSigma_){
1369
1370                if (modelSigma_){
```

```
1371
1372                    Info << "Sigma classifier is being dynamically updated.
                            " << endl;
1373
1374                     // Extract the sigma result from the python model
                            return object.
1375                    PyArrayObject* sigma_return = (PyArrayObject *)
                            PyDict_GetItemString(pReturn, "sigma");
1376                    PyErr();
1377
1378                    // Check if the sigma result is actually returned by
                            python and is not NULL.
1379                    if(sigma_return == NULL) FatalError << "'sigma' not in
                            return value of python model()"<< exit(FatalError);
1380
1381                    // Check if the returned array has the expected
                            dimensions.
1382                    if (PyArray_DIM(sigma_return, 0) != num_cells)
1383                    FatalError << "Dimension of python model sigma
                            corrections are wrong != " << num_cells << exit(
                            FatalError);
1384
1385                    // Extract sigma return values from python model return
                             object.
1386                    forAll(sigma_.internalField(), id){
1387
1388                        sigma_[id] = *((double*)PyArray_GETPTR2(
                                sigma_return, id,0));
1389                        PyErr();
1390                    }
1391
1392                } else if (sigma_[0] == 1.20813608515e-37){
1393
1394                    FatalError << "modelSigma set to false, but no sigma
                            file found in the 0 folder!"
1395                    << nl << exit(FatalError);
1396
1397                } else {
1398
1399                    Info << "Sigma classifier is being read from file. " <<
                             endl;
1400                }
1401            } else {
1402
1403                Info << "Sigma classifier is not being used! " << endl;
1404
1405                forAll(sigma_.internalField(), id){
1406
1407                    sigma_[id] = 1.;
1408                }
1409            }
1410
1411            // Delete references to created objects to free up memory.
1412            Py_XDECREF(fieldDict);
1413            PyErr();
1414            Py_DECREF(pReturn);
1415            PyErr();
1416
```

```
1417        }
1418    }
1419
1420
1421    volScalarField G2
1422    (
1423         "G2",
1424           nut*GbyNu - xi_ * bijDeltastabilizer_ * sigma_ * (2*(this->k_)*
                  bijDelta_ && tgradU())
1425    );
1426
1427
1428    tgradU.clear();
1429
1430
1431    /*--------------------Omega Equation--------------------*/
1432
1433    {
1434        volScalarField::Internal gamma(this->gamma(F1));
1435        volScalarField::Internal beta(this->beta(F1));
1436
1437        tmp<fvScalarMatrix> omegaEqn
1438        (
1439            fvm::ddt(alpha, rho, omega_)
1440          + fvm::div(alphaRhoPhi, omega_)
1441          - fvm::laplacian(alpha*rho*this->DomegaEff(F1), omega_)
1442         ==
1443            alpha()*rho()*gamma
1444           *min
1445            (
1446                    // Production modified due to bijDelta correction.
1447                    G2 / nut(),
1448                (this->c1_/this->a1_)*this->betaStar_*omega_()
1449               *max(this->a1_*omega_(), this->b1_*F23()*sqrt(S2()))
1450            )
1451                // Residual term due to the kDeficit correction.
1452              + alpha()*rho()*gamma*sigma_*kDeficit_/nut()*(xi_ *
                    kDeficitstabilizer_)
1453          - fvm::SuSp((2.0/3.0)*alpha()*rho()*gamma*divU, omega_)
1454          - fvm::Sp(alpha()*rho()*beta*omega_(), omega_)
1455          - fvm::SuSp
1456            (
1457                alpha()*rho()*(F1() - scalar(1))*CDkOmega()/omega_(),
1458                omega_
1459            )
1460          + this->Qsas(S2(), gamma, beta)
1461          + this->omegaSource()
1462          + fvOptions(alpha, rho, omega_)
1463        );
1464
1465            // Update omega and G at the wall
1466            omega_.boundaryFieldRef().updateCoeffs();
1467        omegaEqn.ref().relax();
1468        fvOptions.constrain(omegaEqn.ref());
1469        omegaEqn.ref().boundaryManipulate(omega_.boundaryFieldRef());
1470        solve(omegaEqn);
1471        fvOptions.correct(omega_);
1472        bound(omega_, this->omegaMin_);
```

```
    }

     /*----------------------k Equation ----------------------*/

    tmp<fvScalarMatrix> kEqn
    (
        fvm::ddt(alpha, rho, k_)
      + fvm::div(alphaRhoPhi, k_)
      - fvm::laplacian(alpha*rho*this->DkEff(F1), k_)
     ==
            // Production modified due to bijDelta correction.
        alpha()*rho()*this->Pk(G2)
             // Residual term due to the kDeficit correction.
          + alpha()*rho()*sigma_*kDeficit_()*(xi_ * kDeficitstabilizer_)
      - fvm::SuSp((2.0/3.0)*alpha()*rho()*divU, k_)
      - fvm::Sp(alpha()*rho()*this->epsilonByk(F1, F23), k_)
      + this->kSource()
      + fvOptions(alpha, rho, k_)
    );

    kEqn.ref().relax();
    fvOptions.constrain(kEqn.ref());
    solve(kEqn);
    fvOptions.correct(k_);
    bound(k_, this->kMin_);

    this->correctNut(S2, F23);


}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

} // End namespace RASModels
} // End namespace Foam

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

Listing C.1: `AugmentedkOmegaSST.C` source code

```

/*---------------------------------------------------------------------------
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Copyright (C) 2016-2018 OpenFOAM Foundation
     \\/     M anipulation  |
-------------------------------------------------------------------------*/

/*---------------------------------------------------------------------------
License
    This file is part of OpenFOAM.

    OpenFOAM is free software: you can redistribute it and/or modify it
    under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    OpenFOAM is distributed in the hope that it will be useful, but WITHOUT
```

```
20      ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
21      FITNESS FOR A PARTICULAR PURPOSE.  See the GNU General Public License
22      for more details.
23
24      You should have received a copy of the GNU General Public License
25      along with OpenFOAM.  If not, see <http://www.gnu.org/licenses/>.
26
27  SourceFiles
28      AugmentedkOmegaSST.C
29
30  \*---------------------------------------------------------------------------*/
31
32  #ifndef AugmentedkOmegaSST_H
33  #define AugmentedkOmegaSST_H
34
35  #include "RASModel.H"
36  #include "eddyViscosity.H"
37
38  #include "eddyViscosity.H"
39
40  // The following is for Python interoperability.
41  #include <Python.h>
42  #define NPY_NO_DEPRECATED_API NPY_1_7_API_VERSION
43  #include <numpy/arrayobject.h>
44
45  /*---------------------------------------------------------------------------*/
46
47  namespace Foam
48  {
49  namespace RASModels
50  {
51
52  template<class BasicTurbulenceModel>
53  class AugmentedkOmegaSST
54  :
55      public eddyViscosity<RASModel<BasicTurbulenceModel>>
56  {
57  protected:
58
59          // Model coefficients:
60              dimensionedScalar alphaK1_;
61              dimensionedScalar alphaK2_;
62              dimensionedScalar alphaOmega1_;
63              dimensionedScalar alphaOmega2_;
64              dimensionedScalar gamma1_;
65              dimensionedScalar gamma2_;
66              dimensionedScalar beta1_;
67              dimensionedScalar beta2_;
68              dimensionedScalar betaStar_;
69              dimensionedScalar a1_;
70              dimensionedScalar b1_;
71              dimensionedScalar c1_;
72              Switch F3_;
73
74          // Outputting fields of interest:
75              volScalarField y_;
76              volScalarField k_;
77              volScalarField omega_;
```

```
78              volTensorField gradU_;
79              volSymmTensorField tauij_B_;
80              volSymmTensorField tauij_no_bij;
81
82          // k Equation Terms:
83              volScalarField Pk_;
84              volScalarField Pk_prop_;
85              volScalarField Dk_;
86              volScalarField k_conv_;
87              volScalarField k_diff_;
88
89          // Simulation options:
90              Switch baseline_;
91              Switch usekDeficit_;
92              Switch usebijDelta_;
93              Switch useSigma_;
94              Switch modelbijDelta_;
95              Switch modelkDeficit_;
96              Switch modelSigma_;
97
98          // Custom model variables:
99
100             // Scalars
101             dimensionedScalar   bijDeltastabilizer_;
102             dimensionedScalar   kDeficitstabilizer_;
103             dimensionedScalar   rampStartTime_;
104             dimensionedScalar   rampEndTime_;
105             dimensionedScalar   xi_;
106
107             // Modeled Fields
108             volScalarField      kDeficit_;
109             volSymmTensorField  bijDelta_;
110             volScalarField      sigma_;
111
112             // Python interaction variables
113             PyObject *pName;        // name of python file
114             PyObject *pModule;      // name of module inside python file
115             PyObject *model;        // Model function
116             PyObject *fieldDict;    // Dict-of-arrays contain field data
117
118
119             // Define number of mesh cells variable (set in constructor).
120             int num_cells;
121
122
123     // Protected Member Functions
124
125
126         // Setup field pointers
127         void update_fieldDict();
128
129         virtual tmp<volScalarField> F1(const volScalarField& CDkOmega)
                const;
130         virtual tmp<volScalarField> F2() const;
131         virtual tmp<volScalarField> F3() const;
132         virtual tmp<volScalarField> F23() const;
133
134         // Boundary cells included
```

```
        tmp<volScalarField> blend
        (
            const volScalarField& F1,
            const dimensionedScalar& psi1,
            const dimensionedScalar& psi2
        ) const
        {
            return F1*(psi1 - psi2) + psi2;
        }

        // Only internal mesh cells
        tmp<volScalarField::Internal> blend
        (
            const volScalarField::Internal& F1,
            const dimensionedScalar& psi1,
            const dimensionedScalar& psi2
        ) const
        {
            return F1*(psi1 - psi2) + psi2;
        }

        tmp<volScalarField> alphaK(const volScalarField& F1) const
        {
            return blend(F1, alphaK1_, alphaK2_);
        }

        tmp<volScalarField> alphaOmega(const volScalarField& F1) const
        {
            return blend(F1, alphaOmega1_, alphaOmega2_);
        }

        tmp<volScalarField::Internal> beta
        (
            const volScalarField::Internal& F1
        ) const
        {
            return blend(F1, beta1_, beta2_);
        }

        tmp<volScalarField::Internal> gamma
        (
            const volScalarField::Internal& F1
        ) const
        {
            return blend(F1, gamma1_, gamma2_);
        }

        virtual void correctNut
        (
            const volScalarField& S2,
            const volScalarField& F2
        );

        virtual void correctNut();

        //- Return k production rate
        virtual tmp<volScalarField::Internal> Pk
        (
```

```
193             const volScalarField::Internal& G
194         ) const;
195 
196         //- Return epsilon/k which for standard RAS is betaStar*omega
197         virtual tmp<volScalarField::Internal> epsilonByk
198         (
199             const volScalarField::Internal& F1,
200             const volScalarField::Internal& F2
201         ) const;
202 
203         virtual tmp<fvScalarMatrix> kSource() const;
204 
205         virtual tmp<fvScalarMatrix> omegaSource() const;
206 
207 
208         virtual tmp<fvScalarMatrix> Qsas
209         (
210             const volScalarField::Internal& S2,
211             const volScalarField::Internal& gamma,
212             const volScalarField::Internal& beta
213         ) const;
214 
215 
216 public:
217 
218     typedef typename BasicTurbulenceModel::alphaField alphaField;
219     typedef typename BasicTurbulenceModel::rhoField rhoField;
220     typedef typename BasicTurbulenceModel::transportModel transportModel;
221 
222     //- Runtime type information
223     TypeName("AugmentedkOmegaSST");
224 
225     // Constructors
226 
227         //- Construct from components
228         AugmentedkOmegaSST
229         (
230             const alphaField& alpha,
231             const rhoField& rho,
232             const volVectorField& U,
233             const surfaceScalarField& alphaRhoPhi,
234             const surfaceScalarField& phi,
235             const transportModel& transport,
236             const word& propertiesName = turbulenceModel::propertiesName,
237             const word& type = typeName
238         );
239 
240         //- Disallow default bitwise copy construction
241         AugmentedkOmegaSST(const AugmentedkOmegaSST&) = delete;
242 
243 
244     //- Destructor
245     virtual ~AugmentedkOmegaSST()
246     {}
247 
248 
249     // Member Functions
250 
```

```
251         //- Re-read model coefficients if they have changed
252         virtual bool read();
253 
254         //- Return the effective diffusivity for k
255         tmp<volScalarField> DkEff(const volScalarField& F1) const
256         {
257             return volScalarField::New
258             (
259                 "DkEff",
260                 alphaK(F1)*this->nut_ + this->nu()
261             );
262         }
263 
264         //- Return the effective diffusivity for omega
265         tmp<volScalarField> DomegaEff(const volScalarField& F1) const
266         {
267             return volScalarField::New
268             (
269                 "DomegaEff",
270                 alphaOmega(F1)*this->nut_ + this->nu()
271             );
272         }
273 
274         //- Return the turbulence kinetic energy
275         virtual tmp<volScalarField> k() const
276         {
277             return k_;
278         }
279 
280         //- Return the turbulence kinetic energy dissipation rate
281         virtual tmp<volScalarField> epsilon() const
282         {
283             return volScalarField::New
284             (
285                 "epsilon",
286                 betaStar_*k_*omega_,
287                 omega_.boundaryField().types()
288             );
289         }
290 
291         //- Return the turbulence kinetic energy dissipation rate
292         virtual tmp<volScalarField> omega() const
293         {
294             return omega_;
295         }
296 
297         //- Solve the turbulence equations and correct the turbulence
               viscosity
298         virtual void correct();
299 
300 
301     // Member Operators
302 
303         //- Disallow default bitwise assignment
304         void operator=(const AugmentedkOmegaSST&) = delete;
305 
306         //Probably not used, but left in just in case -Kaj
307         tmp<Foam::fvVectorMatrix> divDevReff(volVectorField& U) const;
```

```
308
309         //- Return the modified effective stress tensor
310         virtual tmp<volSymmTensorField> devRhoReff() const;
311
312         //- Return the modified source term for the momentum equation
313         virtual tmp<fvVectorMatrix> divDevRhoReff(volVectorField& U) const;
314
315          //- Return the modified source term for the momentum equation
316         virtual tmp<fvVectorMatrix> divDevRhoReff
317         (
318             const volScalarField& rho,
319             volVectorField& U
320         ) const;
321
322 };
323
324 /*---------------------------------------------------------------------------*/
325
326 } // End namespace RASModels
327 } // End namespace Foam
328
329 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
330 #ifdef NoRepository
331     #include "AugmentedkOmegaSST.C"
332 #endif
333
334 #endif
335
336 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
```

Listing C.2: `AugmentedkOmegaSST.H` source code

```
1
2 """
3
4 Template for the Python file used by AugumentedkOmegaSST.
5
6 This file should be in the main case directory. With the current setup,
7 this file looks for two other files: kDeficitEq and bijDeltaEq, defining
8 the equation for kDeficit, bijDelta. The equation files for kDeficit and
9 bijDelta only need to be present if modelkDeficit and modelbijDelta
10 are set to true.
11
12 If modelSigma is set to true, sigma will be modeled based on the
13 Pk_Dk_ratio and turbulence intensity.
14
15 Author: Monica Lacatus
16 Adapated from: Kaj Hoefnagel
17
18 """
19
20 # ********************** Importing Libraries ********************** #
21
22 import numpy as np
23 import os
24 from sparta.readOFInternalField import readOFInternalField
25 from sparta.features import FlowFeatures
26 from sparta.util import rdiv, rlog, sqrt_abs
27 import contextlib
```

```
import warnings
import gc
import copy

# ****************** Reading Model Equation Files ********************** #

"""
    Read the contents of the kDeficitEq, bijDeltaEq and sigmaEq files,
    removing enters. If the file is not present, the content
    variable (kDeficitEq/bijDeltaEq/sigmaEq) is set to False and the user
    receives a warning.

"""

if os.path.isfile('./kDeficitEq'):
    with open('./kDeficitEq') as fkDeficit:
        kDeficitEq = fkDeficit.read().replace('\n','')

if os.path.isfile('./bijDeltaEq'):
    with open('./bijDeltaEq') as fbijDelta:
        bijDeltaEq = fbijDelta.read().replace('\n','')

if os.path.isfile('./sigmaEq'):
    with open('./sigmaEq') as fSigma:
        sigmaEq = fSigma.read().replace('\n','')


# ****************** Creating Function Dictionary ********************** #

#Dictionary of functions which may appear in the equation strings
funcDict = {'exp' : np.exp, 'abs' : np.abs, 'tanh' : np.tanh,
            'sqrt' : np.sqrt, 'np' : np,
            'rdiv' : rdiv, 'rlog' : rlog, 'sqrt_abs' : sqrt_abs}


# ****** Defining model Function Called by AugumentedkOmegaSST ********* #

def model(inputDict):


    # Make a deep copy of inputDict to use for further calculation.
    # This ensures that the contents of inputDict do not get affected
    # during the creating and usage of the lazy dict featureDcit.

    inputDict = copy.deepcopy(inputDict)

    # Get the number of grid cells
    NCells = inputDict['k'].shape[0]

    # Setup a FlowFeatures object from inputDict, this does not have the
       features
    # set up yet.
    flow = FlowFeatures.from_inputarray(inputDict)

    # Set up the features for the FlowFeatures object. If the program is
       not run
    # directly (but presumably by OpenFOAM), suppress the print output for
       better
```

```
    # performance.Use mean flow time scale to non-dimensionalize.
    if __name__ != "__main__":
        with contextlib.redirect_stdout(None):
            flow.setup_features(meanFlowTimeScale=False)
    else:
        flow.setup_features(meanFlowTimeScale=False)

    # Extract the lazy feature dictionary from the FlowFeatures object.
    featureDict = flow.vv

    # Add "const" to the featureDict as a variable; this was used in some old
    # equations, it is simply 1 everywhere.
    featureDict['const'] = lambda : np.ones(NCells)

    # Computation of bijDelta
    if not os.path.isfile('./bijDeltaEq'):
        featureDict['bijDelta'] = np.ones((6, NCells))
    else:

        featureDict['bijDelta'] = eval(bijDeltaEq, funcDict, featureDict)

    # Computation of kDeficit
    if not os.path.isfile('./kDeficitEq'):
        featureDict['kDeficit'] = np.ones(NCells)
    else:
        featureDict['kDeficit'] = eval(kDeficitEq, funcDict, featureDict)

    # Computation of sigma
    Pk = featureDict['Pk']
    Pk_prop = featureDict['Pk_prop']
    Dk = featureDict['Dk']
    U  = featureDict['U']
    k  = featureDict['k']

    magSqrU = np.sum(U**2, axis=1)
    turb_intensity = k/(0.5*magSqrU+k);
    Pk_Dk_ratio = np.zeros(NCells)

    for i in range(NCells):
        if Pk_prop[i] >= 0:
            Pk_Dk_ratio[i] = np.abs(Dk[i])/(np.abs(Dk[i]) + np.abs(Pk[i]) +
                np.abs(Pk_prop[i]))
        else:
            Pk_Dk_ratio[i] = (np.abs(Dk[i])+np.abs(Pk_prop[i]))/(np.abs(Dk[
                i]) + np.abs(Pk[i]) +np.abs(Pk_prop[i]))

    featureDict['sigma'] = np.zeros(NCells)
    for i in range(NCells):
        if Pk_Dk_ratio[i]<=0.55 and turb_intensity[i]>=0.12:
            featureDict['sigma'][i] = 1

    #Create the return dict with kDeficit, bijDelta and sigma
    ReturnDict = {'kDeficit': featureDict['kDeficit'].reshape((-1,1)),
                  'bijDelta': featureDict['bijDelta'].T,
                  'sigma': featureDict['sigma'].reshape((-1,1))}
```

```
138     #Cleanup to prevent memory leak
139     del inputDict, flow, featureDict, NCells, Pk, Dk, U, k, Pk_prop,
            Pk_Dk_ratio, magSqrU, turb_intensity
140 
141     gc.collect()
142 
143     return ReturnDict
```

Listing C.3: `python_model.py` source code