COMPUTATIONAL MODELING OF HYPERSONIC TURBULENT BOUNDARY LAYERS BY USING MACHINE LEARNING

A Thesis

Submitted to the Faculty

of

Purdue University

by

Abhinand Ayyaswamy

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

August 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF THESIS APPROVAL

Dr. Haifeng Wang, Chair

School of Aeronautics and Astronautics

Dr. Li Qiao

School of Aeronautics and Astronautics

Dr. Jonathan Poggie

School of Aeronautics and Astronautics

Approved by:

Dr. William A Crossley

Head of the School Graduate Program

I dedicate this thesis to my parents, my sister and
to my late grandmother Smt. Rukmani Krishnamoorthy

ACKNOWLEDGMENTS

This thesis would be incomplete without the valuable contribution and suggestions from numerous people in my professional as well as personal life.

First of all, I am eternally grateful to my parents and my sister who have been my pillars of support in encouraging me to pursue my interests and always place my desires above their own. I would also like to thank my grandparents (Shri. Subramanian and Sumathi Subramanian) for their constant love and words of encouragement apart from their financial support which helped me to focus on my research instead of thinking about the qualms revolving my graduate student life. Without the support from my relatives and these individuals, my passion for science and to excel as a researcher would have never been possible.

At Purdue University, I want to acknowledge my supervisor Dr. Haifeng Wang for accepting me as a member of the CEPL (Computational Energy and Propulsion Laboratory) group and my committee members, Dr. Li Qiao and Dr. Jonathan Poggie for agreeing to be part of my thesis defense. The expertise, experience and countless constructive criticisms of Dr. Wang have not only improved the quality of this research but also molded my technical skills and helped me to approach any new problem with the correct attitude. I should also thank my fellow CEPL colleagues: Jie Tao, Utsav Jain, Tianfang Xie, Jinlong Liu, Shashank Kashyap, Tejas Pant and Pei Zhang. Their extensive technical experience and support has helped me in various parts and in the successful accomplishment of this thesis. In addition to my group members, I also would like to recognize the countless hours of support and discussions held with Dr. Gregory Blaisdell, in various aspects of this research.

This thesis would have been impossible without the collective support, source of positive inspiration and encouragement from my college friends at times of emotional despair during different phases of my research. I would like to thank this special group of friends (fondly called as members of "Amber B32") and several close friends from my undergraduate university for their general advice and cheerful attitude in revitalizing a sense of confidence in me to push forward stronger.

Lastly, I would also like to thank the Research Computing Center (Supercomputing clusters), and the Engineering Computer Network (ECN) for helping me with the computational resources for several simulations performed in my thesis. I also thank the members of Information Technology Research Computing (ITaP) for several technical discussions and suggestions in my research work flow.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

SYMBOLS

Latin Alphabets:

$B_i^{(i=1,2,3)}$	Weight functions
$C_f = \frac{2\tau_w}{\rho_\infty U_\infty^2}$	Skin Friction coefficient
k	Turbulent kinetic energy
M	Mach Number
m	Number of wall-normal locations
n	Number of streamwise locations
P	Grid size parameter
p	Pressure
Pr	Prandtl number
R	Recovery factor
$Re_\tau = \frac{u_\tau \delta}{\nu_w}$	Reynolds number based on friction velocity
$Re_* = \frac{u_* \delta}{\nu_w}$	Reynolds number based on density-weighted friction velocity
T	Temperature
u	Stream wise velocity
U	Mean streamwise velocity
u'	Stream wise velocity fluctuation
U^+	Mean streamwise velocity normalized by friction velocity
U^S	Mean transformed streamwise velocity
v	Wall normal velocity
$\tilde{u} = \sqrt{\overline{u'^2}}$	Root-mean square streamwise velocity fluctuations
$\tilde{v} = \sqrt{\overline{v'^2}}$	Root-mean square wall-normal velocity fluctuations
$\overline{u'v'}$	Reynolds stress
u_τ	Friction velocity
u_*	Density-weighted friction velocity (Morkovin Scaling)

x	Stream wise distance
y	Wall-normal distance
$y^+ = \frac{u_\tau y}{\nu_w}$	Wall-normal coordinate in viscous length unit
$y^* = \frac{yu^*}{\nu}$	Semi-local scaling of wall-normal distance

Greek Alphabets:

α	Thermal conductivity
β	Gradient Correction Factor
γ	Ratio of specific heats
δ	Boundary layer thickness
ε	Turbulent-Dissipation Rate
κ	von Kármán constant
μ	Kinematic Viscosity
ν	Dynamic viscosity
ω_{u_τ}	Rescaling based on friction velocity ratio
ω_{ν_w}	Rescaling based on dynamic viscosity ratio
ω_{ρ_w}	Rescaling based on density ratio
ρ	Density
θ	Momentum Thickness
τ	Shear stress

Subscripts:

∞	Free stream conditions
0	Stagnation conditions
l	Laminar quantities
r	Recovery quantities
s	Static conditions
t	Turbulent quantities
w	Wall conditions

ABBREVIATIONS

ANSYS	ANalysis SYStems
ANN	Artificial Neural Networks
BL	Boundary Layer
CFD	Computational Fluid Dynamics
DNS	Direct Numerical Simulation
HTBL	Hypersonic Turbulent Boundary Layer
LES	Large-Eddy Simulation
LSBOOST	Least-Square Boosting
ML	Machine Learning
MATLAB	MATrix LABoratory
RANS	Reynolds Averaged Navier Stokes
RF	Random Forest
RR	Rescaling Recycling
RSM	Reynold's Stress Model
TKE	Turbulence Kinetic Energy
UDF	User Defined Function
ZPGBL	Zero Pressure Gradient Boundary Layer

ABSTRACT

Ayyaswamy, Abhinand MS, Purdue University, August 2020. Computational Modeling of Hypersonic Turbulent Boundary Layers by using Machine Learning. Major Professor: Dr. Haifeng Wang.

A key component of research in the aerospace industry constitutes hypersonic flights ($M > 5$) which includes the design of commercial high-speed aircrafts and development of rockets. Computational analysis becomes more important due to the difficulty in performing experiments and reliability of its results at these harsh operating conditions. There is an increasing demand from the industry for the accurate prediction of wall-shear and heat transfer with a low computational cost. Direct Numerical Simulations (DNS) create the standard for accuracy, but its practical usage is difficult and limited because of its high cost of computation. The usage of Reynold's Averaged Navier Stokes (RANS) simulations provide an affordable gateway for industry to capitalize its lower computational time for practical applications. However, the presence of existing RANS turbulence closure models and associated wall functions result in poor prediction of wall fluxes and inaccurate solutions in comparison with high fidelity DNS data. In recent years, machine learning emerged as a new approach for physical modeling. This thesis explores the potential of employing Machine Learning (ML) to improve the predictions of wall fluxes for hypersonic turbulent boundary layers. Fine-grid RANS simulations are used as training data to construct a suitable machine learning model to improve the solutions and predictions of wall quantities for coarser meshes. This strategy eliminates the usage of wall models and extends the range of applicability of grid sizes without a significant drop in accuracy of solutions. Random forest methodology coupled with a bagged aggregation algorithm helps in modeling a correction factor for the velocity gradient at the first grid points. The training data set for the ML model extracted from fine-grid RANS, includes neighbor

cell information to address the memory effect of turbulence, and an optimal set of parameters to model the gradient correction factor (β) using the greedy algorithm. The successful demonstration of accurate predictions of wall-shear for coarse grids using this methodology, provides the confidence to build machine learning models to use DNS or high-fidelity modeling results as training data for reduced-order turbulence model development. This paves the way to integrate machine learning with RANS to produce accurate solutions with significantly lesser computational costs for hypersonic boundary layer problems.

1. INTRODUCTION

1.1 Hypersonic Turbulent boundary layers

Research in hypersonic flow which indicates analysis and understanding flow physics of $M > 5$ has become extremely important with the growth and technological advancement in space industry and commercial air travel. The visualization and knowledge of Hypersonic Turbulent Boundary Layers (HTBL) through experiments has become increasingly difficult due to the nature of flow conditions. Predictions of flow solutions and analysis through Computational Fluid Dynamics (CFD) solvers are extremely helpful to design devices and equipments for hypersonic applications.

1.1.1 Current modeling approaches for high speed problems

The majority of high-fidelity turbulent boundary layer simulations for a range of flows from incompressible to hypersonic speeds comes from Direct Numerical Simulations (DNS). In DNS, the Navier-Stokes equations are solved by resolving all the scales of motion from the largest scale to the Kolmogorov scale with suitable boundary and initial conditions to solve a particular type of flow (Pope [21]). Conceptually it is the simplest approach and when it can be applied, it is unrivaled in accuracy for each simulation producing a single realization of the flow.

Large-Eddy Simulations (LES) were used for an effective prediction of unsteady and transitional turbulent flows since it is comparatively cheaper than DNS. Earliest modeling approaches included the usage of dynamic Smagorinsky model (Moin [19]) and the the direct modeling of the SGS-stress tensor in the monotonically integrated LES approach (MILES) shown by Urbin [34]. More recent developments described by Stolz [33] used an approximate deconvolution closure model (ADM) where an

approximation of the unfiltered solution is constructed and introduced directly to the non linear terms in the transport equation.

Reynold's Averaged Navier Stokes (RANS) simulations are often preferred due to its lowest computational cost for both industrial and scientific research in high speed flows. Earlier RANS calculations used the Spalart-Allmaras (one equation model), and recent calculations of high-speed boundary layer problems involve turbulence closure models like the $k-\varepsilon$ model, the $k-\omega$ model (Wilcox [38]) and their variations. RANS simulations by Huang [9] and Georgiadis [6] use the modified $k-\omega$ model by the inclusion of shear stress transport closure given by Menter [18]. The SST model is a two-layer model which involves the $k-\omega$ in the inner region and the $k-\varepsilon$ in the outer regions of boundary layers. There is a transformation of the outer region model with a blending function to transition smoothly between the two sets of equations. High-speed flow calculations involve the usage of density-based solvers (Huang [9]) for better convergence and the $k-\omega$ SST models also involve compressibility corrections (Wilcox [37]) which modify the coefficients of destruction term in the $k-\omega$ model. The usage of full Reynold's stress models is not common, since it usually does not offer significant advantages to make up for the associated cost of solving the individual Reynold's stress terms (Sharif [29]) than the conventional two-equation k-ω models.

A few hybrid LES techniques like the LES/RANS (Fulton [5]) are used where the solution is calculated by unifying two simulation concepts. The larger scales of turbulent motion are solved using a spatially filtered Navier-stokes equations with an associated subgrid scale model similar to a standard LES approach. In addition, the near-wall regions employ a standard RANS turbulence closure by adopting any of the aforementioned models. This hybrid techniques are helpful in improving the solutions of separated flow, high shear and shock problems where the turbulent flow structures are not captured sufficiently by the conventional RANS models.

1.1.2 Drawbacks of present approaches and turbulence models

The computer requirements associated with Direct Numerical Simulations (DNS) increase rapidly with Reynold's numbers and hence it is important to understand that the computational cost is extremely high. It is for this reason, DNS are used only for flows pertaining low to moderate Reynold's numbers. Large Eddy Simulations (LES) on the other hand, offer a reasonable improvement in the computational cost since it does not solve all the turbulent scales of motion. However, the subgrid scale models used in LES are not devoid of modeling parameters set by to solve a particular problem. The presence of these models affect the quality of solution when dealing with complex flow conditions and is not comparable to the DNS results. In LES approaches, as the grid sizes are decreased, more turbulent structures are resolved rather than modeled and hence the solutions can change upon grid variation posing its unreliability (Georgiadis [6]). In comparison, RANS techniques produce a grid independent solution upon grid refinement.

RANS models are preferred for its lowest CPU time, but suffer from large predictive inaccuracies compared to the reference (DNS). RANS models often are built by formulations of turbulence closure models which are not effective for a wide range of compressible flows. Constant values of turbulent Prandtl and Schmidt numbers are used which can pose a significant limitation. An important problem with RANS model is the usage of wall-models in the near wall region to avoid solving the full transport equations. Wall functions often involve the introduction of various model constants in order to make the solution stable and capture different flow phenomena (like shock BL interactions). These wall functions are highly dependent on the presence of correct y^+ (wall-distance scaling) values. The presence of wall models make it impossible to work with coarse meshes since the y^+ lies outside the specific range used in the formulation of these wall functions. Additionally, most two-equation models use the boussinesq approximation for the prediction of shear stresses, which poses a

significant problem when dealing with highly compressible flows and hence cannot correctly predict the Reynold's stresses.

With steady improvement in solutions of RANS models, its utility can be applied to various industrial problems where DNS or high-fidelity solutions are used in spite of its significant computational overhead. The prediction of heat-transfer and the wall-shear are extremely important in the design of re-entry vehicles and RANS models do not offer high accuracy in the prediction of wall fluxes. This can be directly attributed to the usage of wall functions and the inherent turbulence closure models used in RANS solvers. Hence there is a need to improve the predictive accuracy of these quantities through novel methods and ideas which can be built into the CFD solvers for accurate solutions and preserving its cost-effectiveness.

1.2 Machine learning - a prospective method for predictive modeling

Machine learning (ML) has seen a growing interest in CFD research due to its versatility and compatibility to solve highly nonlinear equations with just the presence of high-quality data. The utilization of available high-fidelity data from DNS and LES results are suitable candidates for training-data to create machine learning models which can be used either separately or built in conjunction with RANS to improve the RANS results for larger grid sizes. Recent research (Wu [40], [41]) has stemmed from the usage of DNS and LES data to use machine learning in various forms. Physics-informed machine learning for predictive turbulence modeling helps in improving the Reynold's stress discrepancies by adding additional ML regression functions constructed from training data (Wang [36]). The regression functions are constructed by training using Random Forest (RF) models. Zhang [44] uses high-fidelity simulations and Artificial Neural Networks (ANN) to work on the reconstruction of function arising from Bayesian inference applied to bypass transition and channel flows. Machine learning has shown promise to be a viable and source of improving traditional RANS models to improve the efficacy using existing DNS

simulations. The usage of accurate solutions and the quality of data is extremely important in modeling the Random Forest (RF) or ANN algorithms. To extend the applicability of machine learning models to various problems, it is necessary to train the data-set with various test cases to enable the model to understand different flow conditions. The large amount of existing simulations pave the perfect path for machine learning to utilize the data and aid in useful predictions of hypersonic flows where experimental studies are increasingly difficult.

1.3 Thesis Outline and major contributions

This thesis addresses the specific problem to minimize the loss of prediction accuracy of RANS results by using coarser grids. Fine grid RANS simulations will be performed and used to provide the training data for machine learning.

The thesis is divided into a total of five chapters. Chapter 1 focuses on giving a brief introduction and literature review outlining the different traditional approaches used in the simulation of high-mach number simulations. This chapter also presents the need of using cheaper simulation techniques for research and industry thereby also outlining the incapability of RANS models to effectively predict important flow quantities due its conventional closure models and wall functions.

Chapter 2 provides a viable structure and framework to generate fine-grid RANS results. Common techniques and rescaling methods are adapted from DNS simulations to decrease the cost of conducting fine-grid simulations even in RANS models. This chapter is important in generating useful training data from RANS models which will be used in the forthcoming chapters to create a viable machine learning model for coarse grid simulations. The initial part of this chapter talks about the details of the numerical methods, the turbulence closure models and near wall treatment used in RANS solvers.

Chapter 3 introduces the concept of machine learning and the type of algorithms which can be used for CFD applications. It starts with a general discussion of basic

data fitting using linear regression and slowly transitioning to specific details of the random-forest machine learning algorithms.

With the knowledge of the framework of random forest models, Chapter 4 outlines the model methodology in constructing a velocity gradient correction factor using the fine-grid results from Chapter 2. This chapter also includes details on the selection of training data, the type of parameters and the usage of different model strategies to help improve the predictions from the traditional RANS models. The last section of this chapter gives a quantitative performance of the machine learning algorithms in the prediction of wall shear.

The last chapter summarizes the work done in this thesis and posits future work using these strategies.

The major contributions of this thesis include:

- Exploration of the feasibility and applicability of machine learning for modeling of hypersonic boundary layer problems.
- Usage of rescaling methodology to aid in the generation of fine-grid RANS simulations at significantly lesser computational costs.
- Development of a novel approach to use random forest machine learning algorithms to help improve the computational accuracy of coarse grid RANS simulations.
- Build a velocity gradient correction factor model and utilize the fine-grid RANS simulation data conducted using the rescaling methods as training data set. This thesis also presents a successful A priori testing of coarse grid prediction of wall shear from the constructed machine learning model.
- Discuss the possibility of eliminating wall functions and improve turbulence model closures in RANS by using machine learning algorithms to help in accurate predictions of solutions to hypersonic boundary layer problems.

2. FINE GRID RANS SIMULATIONS - DATA FOR MACHINE LEARNING

An important component of building the machine learning models in this work is the availability of accurate fine grid results. These fine grid simulations serve as the training data to build models for accurate prediction of coarser meshes. In this chapter, RANS simulations of fine grid are performed to generate data for training of machine learning models. In addition, a strategy from traditional direct numerical simulations is used to further reduce the computational cost of fine grid calculations.

2.1 Hypersonic boundary layer test cases

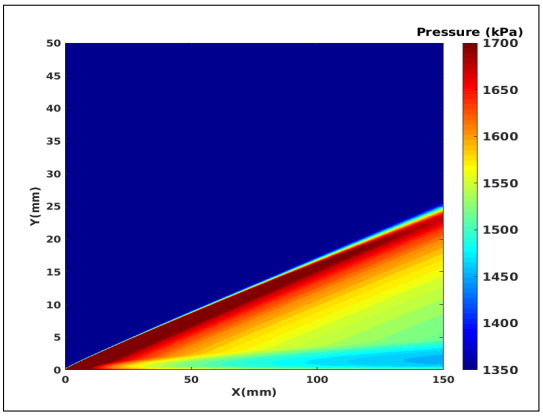
Two similar hypersonic flow conditions were taken as test cases to understand the performance of RANS models. Williams [39] performed experiments on a smooth, flat plate boundary layer with free stream mach number of 7.5. DNS simulations by Priebe [22] was used as a suitable reference for comparison with the experiments owing to the similar flow conditions of high-speed turbulent boundary layer.

Table 2.1. Flow conditions of test cases - An overview

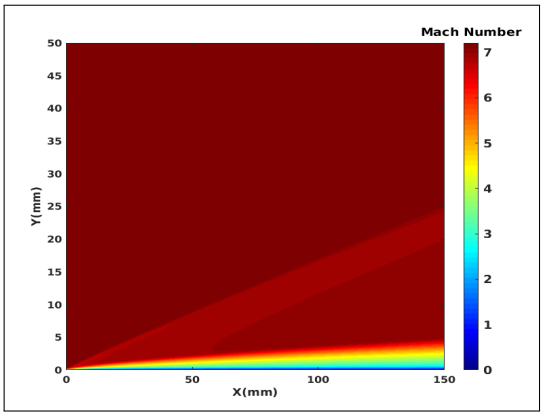
Case	$\boldsymbol{M}$	$\boldsymbol{p_s(kPa)}$	$\boldsymbol{T_0(K)}$	$\boldsymbol{T_w(K)}$	$\boldsymbol{T_w/T_r}$
Expt [39]	7.25	1.24±0.03	756±18	562±12.5	0.83
DNS [22]	7.21	1.39	717	340	0.53

Table 2.1 lists the flow conditions of the hypersonic boundary layer test cases. Here, M is the free stream mach number, p_s is the free stream static pressure, T_0

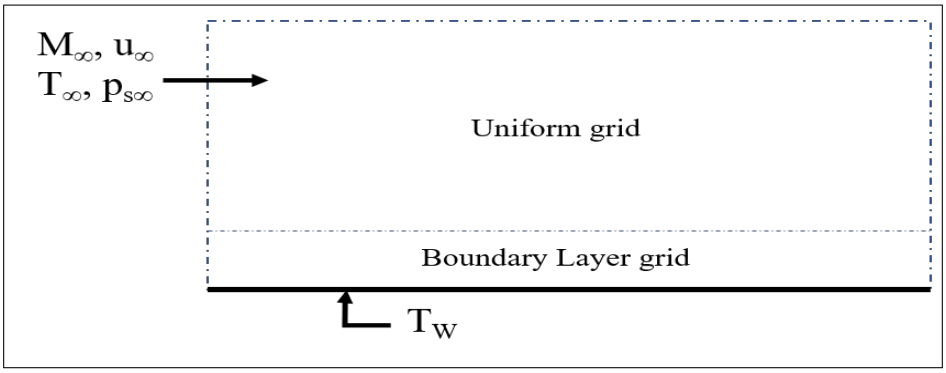
represents the free stream stagnation temperature, T_w is the wall temperature and T_r is the recovery temperature of the flow.



(a) Contour plot of Pressure (p)



(b) Contour plot of Mach number (M)



(c) Sketch of Flow domain

Fig. 2.1. Schematic of Boundary layer flow

Figure 2.1a clearly show the presence of shock which emanates from the stagnation point of the flow over a flat plate. It can also be observed that due to the presence of shock, there is a sudden increase the values of p and M when moving perpendicular to the shock line. Figure 2.1b shows the growth of the boundary layer inside the domain with a no-slip boundary condition at the wall. Figure 2.1c shows a two layer grid construction used in this study. The boundary layer grid where points are arranged in a geometric progression to capture the boundary layer in the flow which is gradually blended with the uniform outer layer grid. For high-speed flow,

Morkovin [20] proposed that the correct velocity scale for the turbulence ought to be the density-weighted velocity scale,

$$u_* = \sqrt{\frac{\tau_w}{\rho}} = u_\tau \sqrt{\frac{\rho_w}{\rho}}, \tag{2.1}$$

where $u_\tau = \sqrt{\tau_w/\rho_w}$ is the friction velocity, τ_w and ρ_w are the wall shear stress and the density at the wall respectively. This morkovin scaling is applied to the turbulent stresses rather than the velocity variances. The experiment performed by Williams [39] was able to show the collapse of the streamwise turbulent intensities, which was an extension to the results of Morkovin for hypersonic flow. Both the experiment and the DNS case serve as accurate references for the solution of the current RANS calculations.

2.2 RANS modeling approaches

Reynolds-Averaged Navier Stokes(RANS) equations are solved to produce results for comparison with the aforementioned experiment and DNS. ANSYS 19.2.0, a commercial computational fluid dynamics code, is used to obtain the results of the hypersonic flow calculations. The DNS adopts a strategy to reduce the computational cost of simulations by employing a recycling-rescaling approach. The same idea is used in RANS by creating auxiliary User Defined Functions(UDF) in C language which can be compiled with ANSYS-Fluent to work in tandem with the flow equations to perform flat plate boundary layer calculations in an inexpensive way. This is an important step in developing a robust method to obtain fine-grid RANS results which are subsequently used as training data for the machine learning model.

2.2.1 Governing Equations

For flows problems, ANSYS Fluent [24] solves conservation equations for mass and momentum. It also becomes important to solve the energy equation in hypersonic

flows since the flow conditions correspond to viscous heating attributed due to high stream-wise velocities at these Mach numbers.

The general equation of conservation of mass, or continuity equation for incompressible and compressible flows, can be written as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = S_m, \tag{2.2}$$

where ρ is the density, $\vec{v}$ is the velocity vector, S_m represents any external mass added to the continuous dispersed second phase or any user-defined sources.

The general equation of conservation of momentum for compressible flows in an inertial (non-accelerating) reference frame is given by

$$\frac{\partial}{\partial t}(\rho \vec{v}) + \nabla \cdot (\rho \vec{v} \vec{v}) = -\nabla p + \nabla \cdot (\overline{\tau}) + \rho \vec{g} + \vec{F}, \tag{2.3}$$

where p is the static pressure, $\overline{\tau}$ is the stress tensor (described below), $\rho \vec{g}$ and $\vec{F}$ are the gravitational body force and external body forces. The stress tensor $\overline{\tau}$ is given by

$$\overline{\tau} = \mu \left[\left(\nabla \vec{v} + \nabla \vec{v}^T \right) - \frac{2}{3} \nabla \cdot \vec{v} I \right], \tag{2.4}$$

where μ is the molecular viscosity and I is the unit tensor.

The general equation for energy conservation is in the following form:

$$\frac{\partial}{\partial t}(\rho E) + \nabla \cdot (\vec{v}(\rho E + p)) = \nabla \cdot \left(k_{eff} \nabla T - \sum_j h_j \overline{J_j} + (\overline{\tau}_{eff} \cdot \vec{v}) \right) + S_h, \tag{2.5}$$

where E is the total energy per unit mass, k_{eff} is the effective conductivity ($k_{eff} = k + k_t$, where k_t is the turbulent thermal conductivity, defined according to the turbulence model being used), and $\overline{J}_j$ is the diffusion flux of species j. The first three terms on the right-hand side of Eq. (2.5) represent energy transfer due to conduction, species diffusion and viscous dissipation, respectively. S_h includes the heat of reaction of external volumetric heat sources defined in a problem.

2.2.2 Numerical methodology

The problem statement concerns highly compressible flow and hence the density based solver is preferred. The density based approach solves the governing equations of continuity, momentum, energy fully coupled. Since the governing equations are highly non-linear and coupled, several iterations of the solution loop must be performed before a converged solution is obtained. Each iteration consists of the steps given by Fig. 2.2

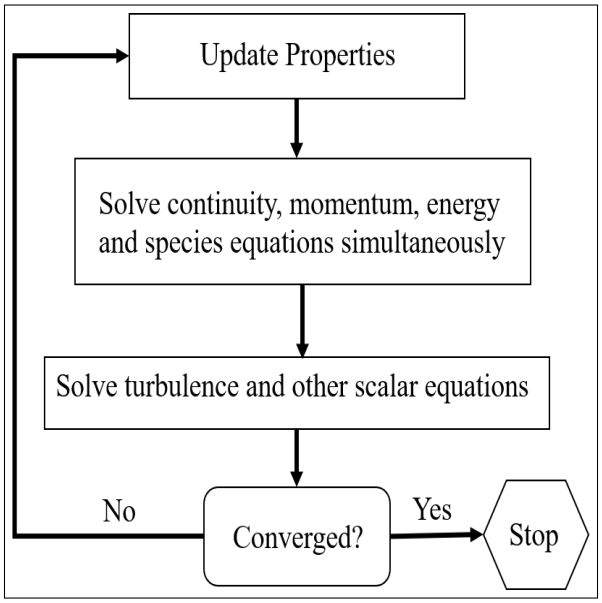


Fig. 2.2. Overview of steps performed on a density-based solver in Fluent

In the density-based solution methods, the discrete, non-linear governing equations are linearized to produce a system of equations for the dependent variables in every computational cell. The resultant linear system is then solved to yield an updated flow-field solution. The discretization approaches used in this study are summarized in Appendix A.

2.2.3 Turbulence Models

Favre averaging is a method by which the solution variables in the instantaneous (exact) Navier-Stokes equations are decomposed into the Favre mean (density-weighted) and fluctuating components. Favre averaging is preferred over Reynolds (ensemble averaging) due to the compressible nature of the flow conditions. For example, the instantaneous velocity and a scalar is split as follows:

$$u_i = \widetilde{u_i} + u_i'' \qquad \phi_i = \widetilde{\phi} + \phi'' \tag{2.6}$$

where $\widetilde{u_i}$ and u_i'' are the mean and fluctuating components of velocity for (i= 1, 2, 3). Similarly, $\widetilde{\phi}$ and ϕ'' correspond to the mean and fluctuating parts of some scalar ϕ.

Cartesian tensor form of the continuity and momentum governing equations after substitution of Reynolds averaging yields:

$$\frac{\partial \overline{\rho}}{\partial t} + \frac{\partial}{\partial x_i}(\overline{\rho}\widetilde{u_i}) = 0 \tag{2.7}$$

$$\begin{aligned}\frac{\partial}{\partial t}(\overline{\rho}\widetilde{u_i}) + \frac{\partial}{\partial x_j}(\overline{\rho}\widetilde{u_i}\widetilde{u_j}) = &-\frac{\partial \overline{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu\left(\frac{\partial \widetilde{u_i}}{\partial x_j} + \frac{\partial \widetilde{u_j}}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial \widetilde{u_k}}{\partial x_k}\right)\right] \\ &+ \underbrace{\frac{\partial}{\partial x_j}(-\rho\overline{u_i''u_j''})}_{\substack{\text{Additional term} \\ \text{which needs modeling}}}\end{aligned} \tag{2.8}$$

where ($\overline{\square}$) represents reynolds averaging and ($\widetilde{\square}$) is favre averaging. Eq. (2.7) and Eq. (2.8) are called the Reynolds Averaged Navier Stokes (RANS) equations. They are similar to Eq. (2.2) and Eq. (2.3) but now the solution variables representing favre-averaged values. An additional term representing the effects of turbulence needs to be modeled to close the Eq. (2.8).

The purpose of turbulence models is to model the last term in Eq. (2.8) to close the momentum equation. This section presents a description of two different types of turbulent models used in the analysis of RANS simulations, Reynolds stress Model (RSM) and the Standard k-ω Model (k-ω). In Reynolds Stress models (RSM) for

2-D flows, five additional transport equations are required solving the shear stresses $\overline{\rho u_i'' u_j''}$.

$$\underbrace{\frac{\partial}{\partial t}(\overline{\rho}\overline{u_i'' u_j''})}_{\text{Local Time Derivative}} + \underbrace{\frac{\partial}{\partial x_k}(\overline{\rho}\widetilde{u_k}\overline{u_i'' u_j''})}_{C_{ij}\equiv\text{Convection}} = \underbrace{-\frac{\partial}{\partial x_k}\left[\overline{\rho u_i'' u_j'' u_k''} + \overline{p'(\delta_{kj}u_i'' + \delta_{ik}u_j'')}\right]}_{D_{T,ij}\equiv\text{Turbulent Diffusion}}$$
$$+ \underbrace{\frac{\partial}{\partial x_k}\left[\mu\frac{\partial}{\partial x_k}(\overline{u_i'' u_j''})\right]}_{D_{L,ij}\equiv\text{Molecular Diffusion}} - \underbrace{\rho\left(\overline{u_i'' u_k''}\frac{\partial \widetilde{u_j}}{\partial x_k} + \overline{u_j'' u_k''}\frac{\partial \widetilde{u_i}}{\partial x_k}\right)}_{P_{ij}\equiv\text{Stress Production}}$$
$$- \underbrace{\overline{\rho}\zeta'\left(g_i\overline{u_j''\theta} + g_j\overline{u_i''\theta}\right)}_{G_{ij}\equiv\text{Buoyancy Production}}$$
$$+ \underbrace{\overline{p'\left(\frac{\partial u_i''}{\partial x_j} + \frac{\partial u_j''}{\partial x_i}\right)}}_{\phi_{ij}\equiv\text{Pressure Strain}} - \underbrace{2\mu\overline{\frac{\partial u_i''}{\partial x_k}\frac{\partial u_j''}{\partial x_k}}}_{\varepsilon_{ij}\equiv\text{Dissipation}}$$
$$+ \underbrace{S_{user}}_{\text{User-Defined Source Term}} \tag{2.9}$$

The terms C_{ij}, $D_{L,ij}$ and P_{ij} do not require any modeling. However the other terms $D_{T,ij}$, G_{ij}, ϕ_{ij} and ε_{ij} need to be modeled to close the equations. The standard modeling procedures for these terms are shown in Appendix B.1.

The standard $k-\omega$ model in Fluent is based on the model developed by Wilcox [38] which incorporates compressibility and modifications to low-Reynolds number effects. This model is an empirical model based on model transport equations for the turbulence kinetic energy (k) and the specific dissipation rate ($\omega = \varepsilon/k$).

$$\frac{\partial}{\partial t}(\rho k) + \frac{\partial}{\partial x_i}(\rho k u_i) = \frac{\partial}{\partial x_j}\left(\Gamma_k\frac{\partial k}{\partial x_j}\right) + G_k - Y_k + S_k \tag{2.10}$$

$$\frac{\partial}{\partial t}(\rho\omega) + \frac{\partial}{\partial x_i}(\rho\omega u_i) = \frac{\partial}{\partial x_j}\left(\Gamma_\omega\frac{\partial \omega}{\partial x_j}\right) + G_\omega - Y_\omega + S_\omega, \tag{2.11}$$

where G_k and G_ω represent the generation of k and ω, respectively Γ_k and Γ_ω represent the effective diffusivity of k and ω, and Y_K and Y_ω represent the dissipation of k and

ω due to turbulence, respectively. The modeling of the unknown terms in Eq. (2.10) and Eq. (2.11) are discussed in Appendix B.2.

2.2.4 Modeling approaches for walls

The presence of walls have a crucial effect on the determination of solution of a turbulent flow problem. The near wall region adds complication and expense to the task of performing turbulence-model calculations of turbulent flows. The idea of the 'wall function' approach is commonly used to apply boundary conditions (based on log-law relations) some distance away from the wall, so that the turbulence-model equations are not solved close to the wall. These semi-empirical formulae are used to bridge the viscosity affected region between the wall and fully-turbulent region.

An important shortcoming of all wall functions is that the numerical results deteriorate on variation of grid sizes. This necessitates the need to move to a new method of determining the variables near the wall region thus completely eliminating the influence of wall functions. The next chapter introduces the concept of Machine Learning which can be helpful in improving the quality of results regardless of the size of grids. It is also important to ensure that the entire boundary layer is sufficiently resolved and the correct y^+ is ensured during the construction of the grids for effective performance of the wall functions.

There are several formulations of wall functions like the Standard Wall functions, Non-Equilibrium wall functions and Scalable wall functions. The choice of wall functions depends on the type of the problem and the model methodology is appropriately different for each of the aforementioned wall functions. This thesis uses an Enhanced Wall function approach to model the near wall cells which is a two layer blending approach for the viscosity-affected region and the outer turbulent flow region. A brief discussion of enhanced wall functions and the modeling equations are given in Appendix C.

2.3 Rescaling and Recycling approach for hypersonic boundary layer simulations

In this section, an adaptation of the Recycling approach is applied to work in conjunction with the RSM and $k-\omega$ model in Fluent by developing user defined functions. Since the problem concerns a compressible flow at $M = 7$, there is a need to improve the calculation of viscosity at extremely low static temperatures and hence Keyes Model of viscosity (Keyes [12]) is adopted. A brief discussion between the existing models is given in Appendix D.

Rescaling and Recycling (RR) is a popular approach used in DNS calculations to extract instantaneous planes of velocity data from an auxiliary simulation of a zero pressure gradient boundary layer. This spatially developing simulation is capable of generating its own inflow conditions by a process of rescaling and recycling from a specific position in the downstream location to the inlet. In DNS simulations of turbulent boundary layer problems, a common technique to improve the computational cost is by the usage of Rescaling methods (Lund [17], Spalart [31], Xu [42]).

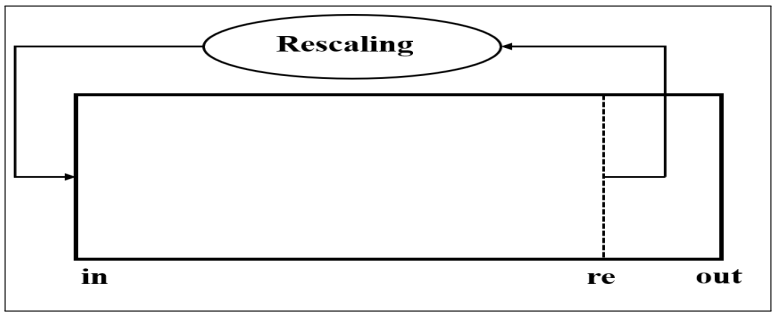


Fig. 2.3. Schematic of the rescaling methodology

This method uses scaling laws by rescaling the flow field at some downstream station and recycling it to the upstream inlet. Earlier methods of rescaling equations did not include the effects of compressibility for high mach numbers and used the same scaling for all the mean and fluctuating variables (Stolz [33], Urbin [34]). More recent approaches (Xu [42]) correctly disregard the Taylor hypothesis and rescale

equations of zero-pressure gradient boundary layers (ZPGBL) using the Morkovin scaling (Morkovin [20]) for density effects. A correct method of using rescaling temperature is also required by the usage of temperature-velocity relationships given by Walz [35].

Figure 2.3 shows a diagrammatic representation of the rescaling and recycling approach. The data is extracted from a recycling station 're', rescaled the variables accordingly and fed back to the inlet as a boundary condition. This work is a adapted to convert and use it for an inbuilt RANS simulation which uses RSM and $k-\omega$ models as turbulence models.

This rescaling method when coupled with RANS simulations, presents a computationally effective way to generate fine-case simulation data which are eventually used as training data for machine learning models discussed in Chapter 4. The developed rescaling and recycling approach and the pertaining equations for RANS modeling of hypersonic boundary layer is described in Appendix E.

2.4 Results and Discussion

This section explains the key results and the performance of traditional RANS models along with the addition of rescaling methods. The main goal is to provide validated fine-grid RANS results for the purpose of machine learning described in later discussions. This section will initially include a parametric study and the reason for choosing a specific turbulence models and wall functions in the computational analysis of Hypersonic Turbulent Boundary Layer (HTBL) flows.

2.4.1 Parametric Analysis

This section shows a parametric study of the performance of different turbulence models and the effect of wall functions pertaining to the DNS test case. In each of the following results, the boundary layer growth, and the profiles of reynolds stress quantities are compared at the suitable locations for a correct comparison.

2.4.1.1 Effect of Turbulence models

Two different turbulence models, $k-\omega$ and RSM models were used to simulate a hypersonic flow.

Table 2.2. Boundary layer results for parametric study on turbulence models

Model	$T_w(K)$	$U_\infty(m/s)$	$\delta(mm)$	$u_\tau(m/s)$	Re_θ	Re_τ	C_f x 10^4
DNS [22]	340	1146.1	4.5	60.7	3300	200	11.73
$k-\omega$	340	1150.2	4.5	52.4	3080	169.4	6.78
RSM	340	1149.5	4.5	67.08	2940	220.18	14.02
Expt [39]	562±12.5	1158	9.5	71.75	4940	180	8.68
$k-\omega$	575	1161.6	9.5	44.32	4230	153.1	4.63
RSM	575	1160.4	9.5	73.12	4110	226.24	8.1

Table 2.2 lists the boundary layer properties taken at two locations $\delta = 4.5mm$ and $9.5mm$. Here, $Re_\theta = U_\infty\theta/\nu_\infty$ is the Reynolds number based on momentum thickness θ, $Re_\tau = u_\tau\delta/\nu_w$ is the Reynolds number based on friction velocity and $C_f = \tau_w/(\rho_\infty U_\infty^2/2)$ is the skin friction coefficient, τ_w is the wall shear stress, and ν_∞, ρ_∞ and U_∞ are the free stream kinematic viscosity, density and velocity respectively.

Figure 2.4 shows that the growth of turbulent boundary layer is vastly different from the RSM model and the $k-\omega$ model shows a significantly lower growth rate in comparison to the other. In Figure 2.5 the $k-\omega$ model cannot provide a good estimate of the wall fluxes in comparison with the RSM model.

Analysis of the results from Table 2.2 taken at $\delta = 4.5mm$ (DNS) and $\delta = 9.5mm$ (EXPT) gives a clearer picture about the inability of the $k-\omega$ model to sufficiently predict the values of fluxes and boundary layer parameters than the RSM. Figure 2.6 also shows that the peak in $\tilde{u}$ cannot be predicted with the $k-\omega$ model even though

the wall-normal fluctuations are pretty good. The RSM model also does better with the profiles of Reynold's stresses inside the boundary layer.

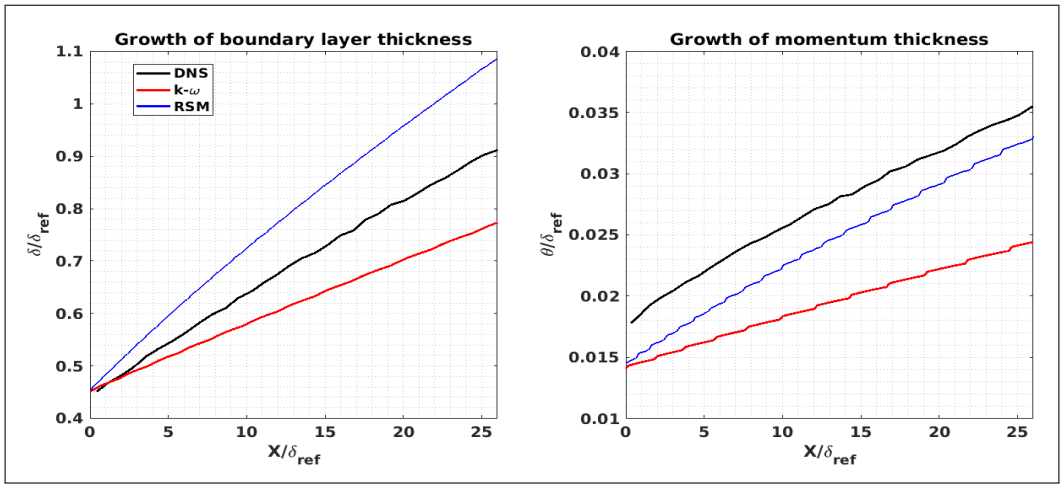


Fig. 2.4. Growth of boundary layer properties (a) Boundary layer thickness(δ) (b) Momentum thickness (θ)

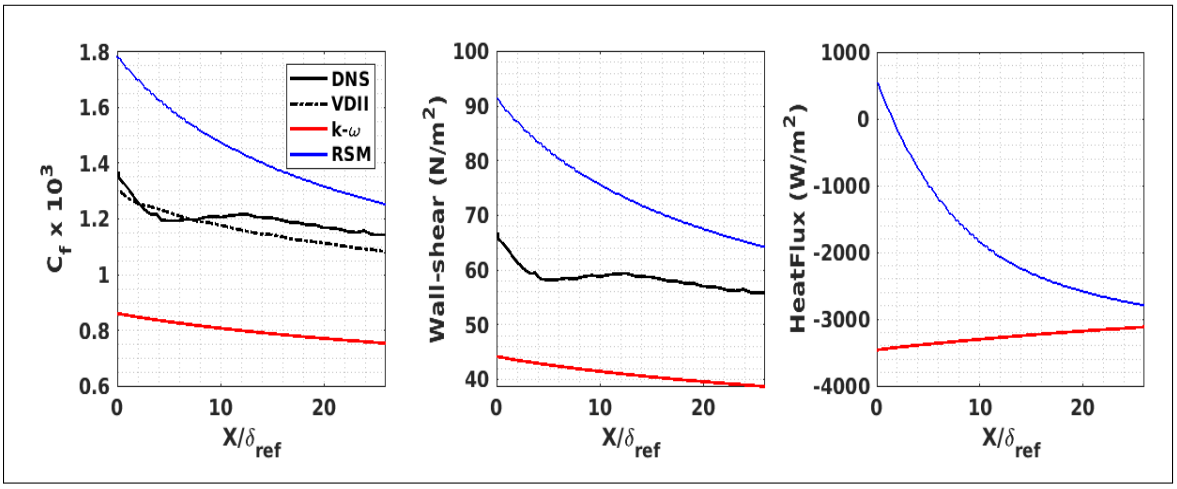


Fig. 2.5. Profiles of wall-fluxes along the wall (a) Skin-friction (b) Wall shear (c) Heat Flux

Figure 2.7 shows that the profiles of the U^+ vs y^+ plot are significantly better in correspondence with the DNS profiles for the RSM model. Although the RSM model cannot sufficiently predict the correct magnitude of the peak, the trend and

the general structure of the profile is predicted better than the $k-\omega$ model. Hence, the RSM model is the chosen turbulence model for the RANS simulations in this section.

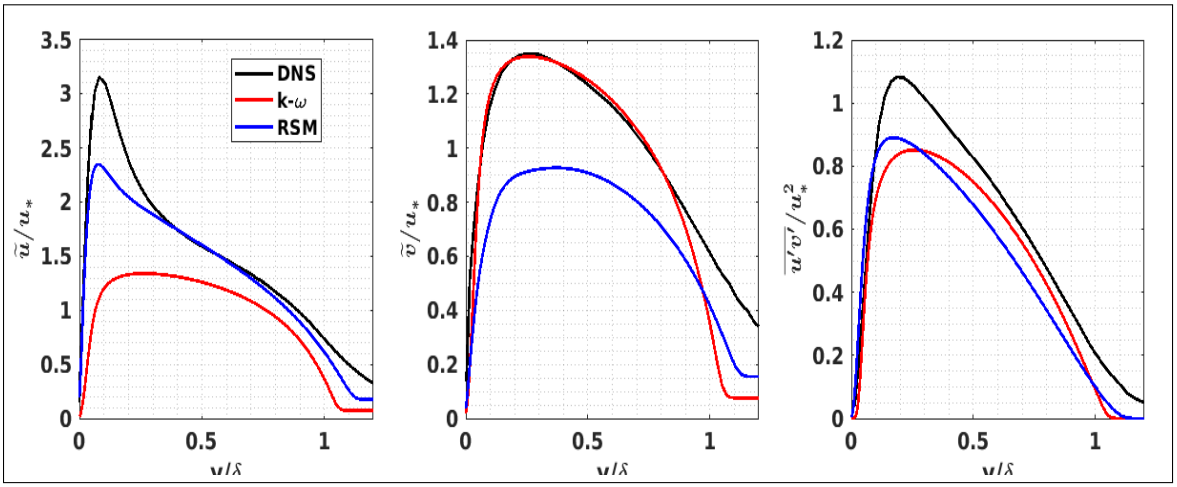


Fig. 2.6. Turbulence profiles inside the boundary layer using Morkovin scaling at $\delta = 4.5mm$ (a) Stream-wise and (b) Wall-normal r.m.s velocities (c) Reynold's shear stress, — DNS by Priebe [22] ($Re_\theta = 3300$, $M = 7.21$, $T_w = 340K$)

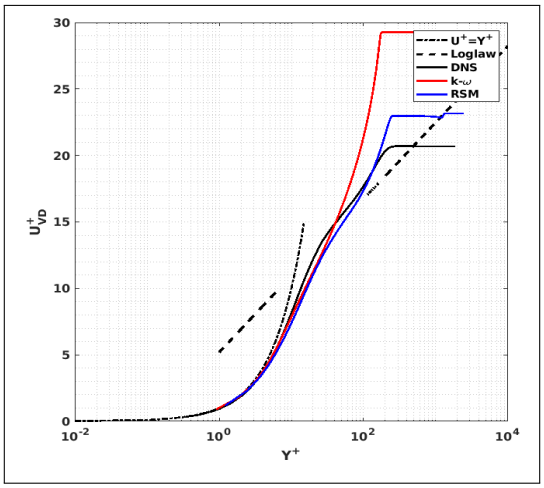


Fig. 2.7. Mean stream-wise velocity normalized by friction velocity vs wall-normal coordinate at $Re_\theta = 3300$, — DNS, Log-law with $\kappa = 0.4$ and B = 5.1

2.4.1.2 Effect Wall function models

The RANS simulation of hypersonic flow was tested with four different wall functions working with the RSM turbulence model namely Standard Wall functions (SWF), Scalable Wall functions (ScWF), Non-equilibrium Wall functions (NEWF) and Enhanced Wall functions (EWF).

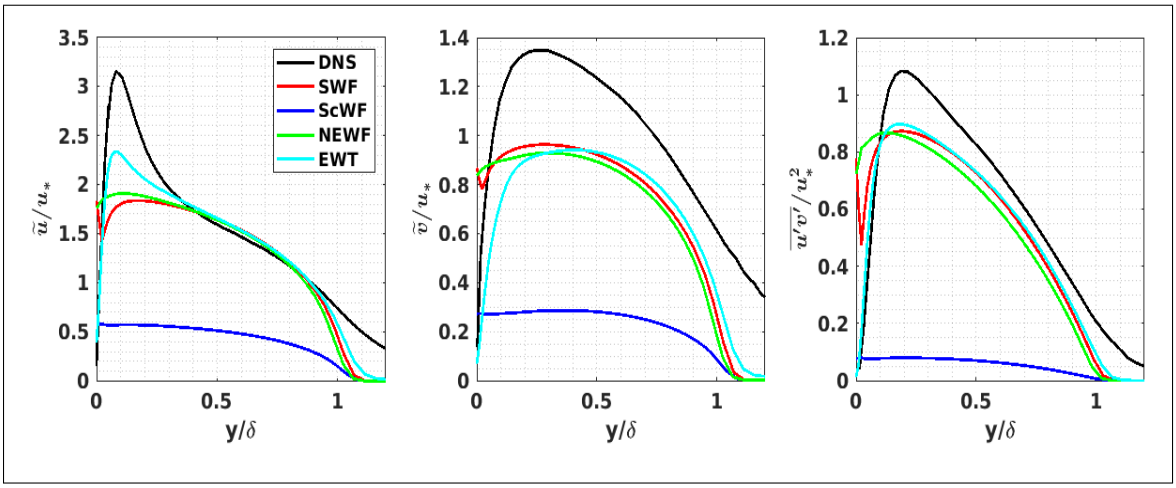


Fig. 2.8. Turbulence profiles for different wall functions (a) Stream-wise and (b) Wall normal r.m.s velocities (c) Reynolds Stress, — DNS at $Re_\theta = 3300$

Figure 2.8 shows the variations with different wall functions. Standard wall functions uses an empirical formulation to construct a value for the turbulence variables and velocity profile in the near wall region. Scalable wall functions forces the usage of the log law in conjunction with the standard wall functions approach. This is achieved by introducing a limiter in the y^+ calculations such that $y^+ = max(y^+, 11.225)$. The non-equilibrium wall functions and enhanced wall functions use a two layer approach to blend the laminar-viscosity affected region and the turbulent region. The above plot, shows that the enhanced wall functions correctly predict the variation of Reynold's stress and r.m.s velocity profiles near the wall. The correct usage of wall functions ensures the production of a streamwise r.m.s velocity peak near the wall region as indicated by the DNS profiles.

2.4.2 Comparison of RANS with DNS and Experimental case

In this section, RANS simulations are done for both the reference conditions with a RSM model and an Enhanced Wall treatment as wall model in the viscosity affected region.

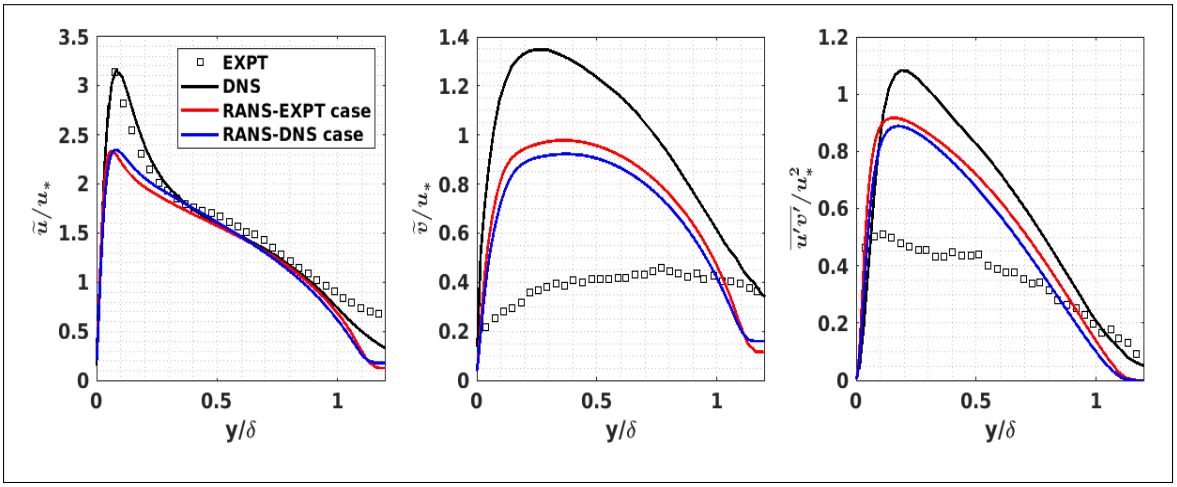


Fig. 2.9. Comparison of turbulence quantities for Expt [39] at $Re_\theta = 4940$ and DNS [22] at $Re_\theta = 3300$

Although the test cases are performed at similar mach numbers and the profiles are analyzed at a similar Re_θ, Figure 2.9 shows there is a significant distinction in the magnitudes of streamwise, wall-normal r.m.s velocity fluctuations and the Reynold's shear stress profiles. The only significant distinction correspond to the wall temperature (T_w) and the T_w/T_r values. The experimental test case was performed at a wall temperature of 570K and T_w/T_r of 0.82, whereas the DNS used a value of 340K and 0.53 indicating a cold-wall test case. We try to narrow down two important things in this section. Since the DNS work used a recycling approach, we try to find if recycling causes any difference to the prediction of turbulence quantities. An important take-away from these plots are the prediction of wall shear stress. A significant component of the Morkovin scaling (see Eq. 2.1) is the friction velocity (u_τ) which contains the value of wall shear stress (τ_{wall}).

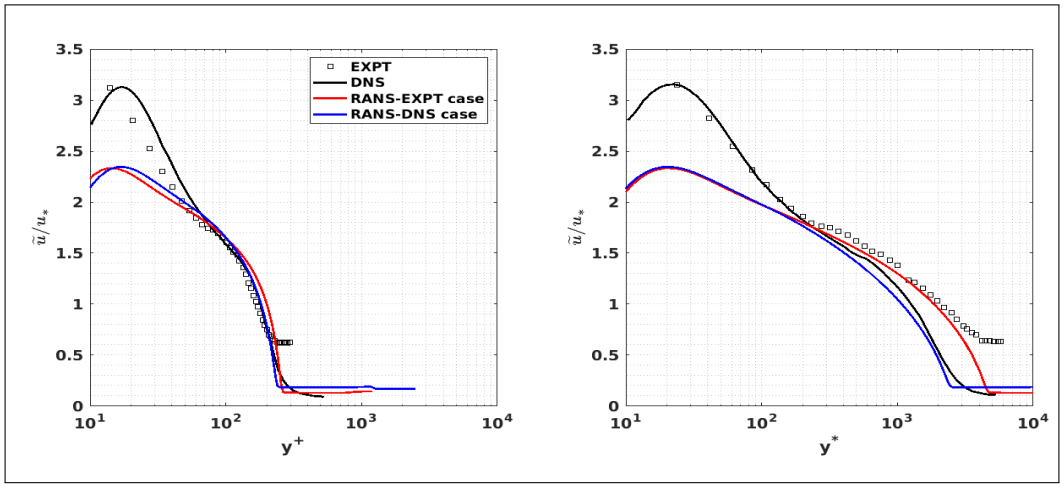


Fig. 2.10. Stream-wise turbulence scalings. The r.m.s velocity is represented in (a) inner and (b) semi-local scaling

In the forthcoming chapters, an effort is made through machine learning to predict the correct value of wall-shear for coarser grids and a strategy is outlined to enable RANS models to capture the value of wall shear and thereby the turbulence profiles effectively.

2.4.3 Effect of Rescaling-Recycling (RR) mechanism

This section describes the comparison of simple-RR and modified-RR method on RANS simulations of the DNS (Priebe [22]) test conditions. These results are simultaneously compared with a test case where no rescaling is implemented (no-RR). At the end of this section, important uses and advantages in utilizing the RR method is discussed.

The DNS test conditions represent a hypersonic flat plate boundary layer flow with $\delta_{inlet} = 2.25mm$, $\theta_{inlet} = 0.0875mm$ and the outlet conditions with $\delta_{outlet} = 4.5mm$, $\theta_{outlet} = 0.175mm$.

Figure 2.11 shows the growth of boundary layer using rescaling to be comparable with the DNS results. It should also be noted that the modified RR method does slightly better in predicting the correct boundary layer growth although for simplicity any of these methods are equally efficient.

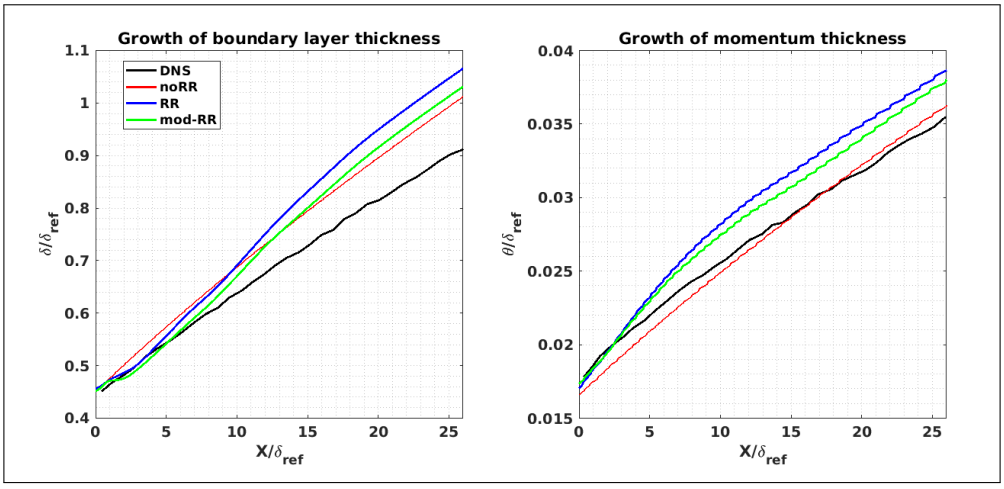


Fig. 2.11. Effect of Rescaling-Recycling (RR) on boundary layer growth

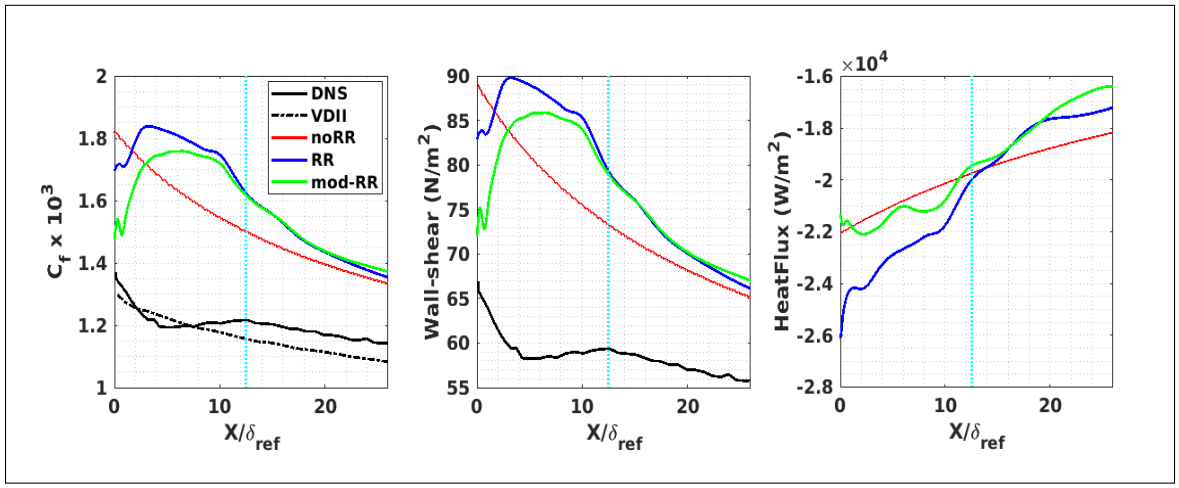


Fig. 2.12. Effect of Rescaling-Recycling (RR) on wall shear stress and heat transfer (a) Skin-friction (b) Wall-shear (c) Heat Flux

Figure 2.12 shows the magnitude of skin friction, wall shear and heat flux along the streamwise direction. An important observation is the vertical line on these figures.

This line represents the start of solutions which are unaffected due to the process of recycling. The initial portion of the computational domain is affected by the recovery of the boundary layer due to the rescaling process and must be discarded (refer Priebe [22]). This is also observed with the reference DNS data. We can definitely see a good comparison and agreement of the modified RR method with a test case where rescaling is not done (noRR case). An area which needs improvement is the magnitude of wall shear stress, which essentially is a key component in the prediction of turbulence profiles.

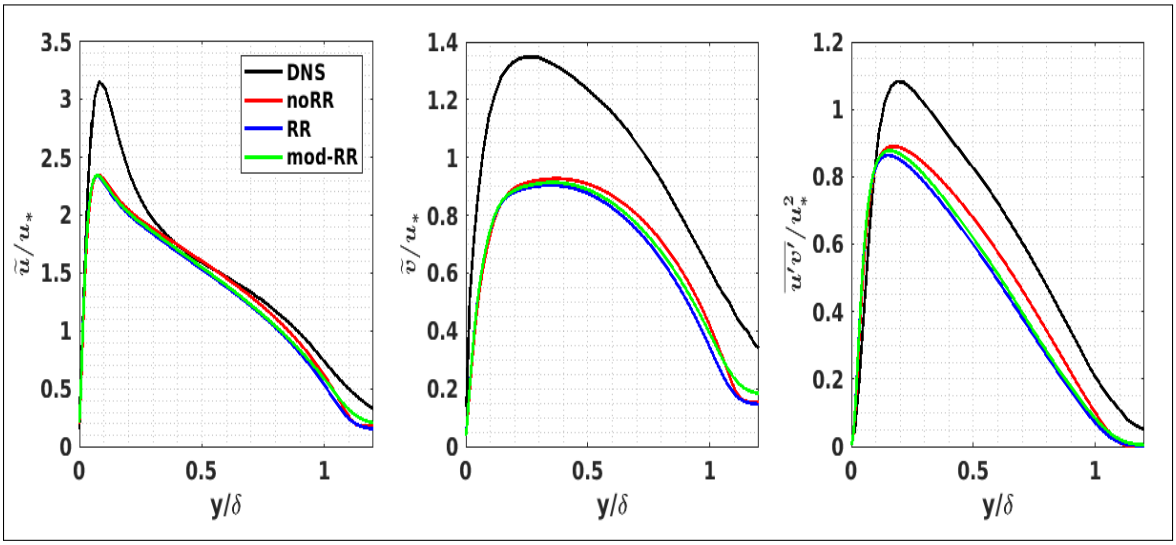


Fig. 2.13. Effect of Rescaling-Recycling (RR) on turbulence profiles. (a) Streamwise and (b) wall-normal velocity r.m.s (c) Reynold's shear stress, — DNS [22] at $Re_\theta = 3300$

The streamwise, wall normal r.m.s velocities, and Reynold's shear stress are comparable with the no-RR case, in reproducing the correct trend of results using the RSM model. Figure 2.13 also shows although the rescaling approach for modified and simple cases, are able to replicate the no-RR case, the process does not change the solution from a generic test no-RR test case. The overall profiles of rescaling are slightly lower in comparison with the no-RR case, however none of the RANS results are able to accurately predict the magnitude represented in the DNS results. The

poor prediction of wall shear which is an effect of the nature of turbulence model influences the overall comparison with the DNS results.

Table 2.3. Boundary layer results to compare the effect of recycling methods

Model	U_∞	$\delta(mm)$	$u_\tau(m/s)$	Re_θ	Re_τ	C_f x 10^4
DNS [22]	1146.1	4.5	60.7	3300	200	11.73
no-RR	1149.5	4.5	67.08	2940	220.18	14.02
Simple RR	1146.1	4.5	68.13	3153.4	230.65	14.87
Modified RR	1146.1	4.5	67.2	3288.3	218.44	14.18

Table 2.3 shows that the Rescaling-Recycling methodology is helpful in capturing the correct free stream velocity, Re_θ in addition to predicting a decent estimate of u_τ and C_f which is directly affected by the value of wall-shear stress. Figure 2.14 shows the streamwise r.m.s velocity fluctuations normalized by the morkovin scaling used in the inner and semi-local scaling. The inner scaling is equivalent to the traditional y^+ values, whereas the semi-local scaling is a new coordinate scaling given by $y^* = (yu^*)/\nu$.

Figure 2.15 shows good comparison and the structure of the U^+ vs y^+ plot. Although the magnitude of the streamwise scaling is not accurate as the DNS, an accurate estimation of the wall-shear stress which directly affects the scaling of U will help in improving the value in the outer region.

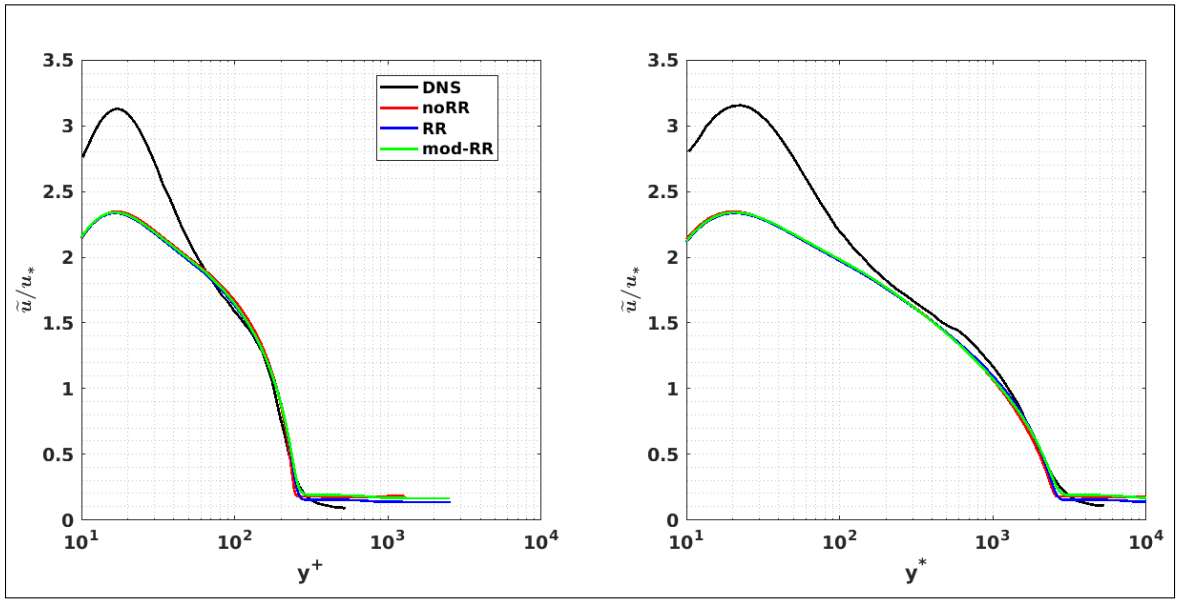


Fig. 2.14. Effect of Recycling method on streamwise fluctuations using morkovin scaling. The r.m.s velocity is represented in (a) inner and (b) semi-local scaling

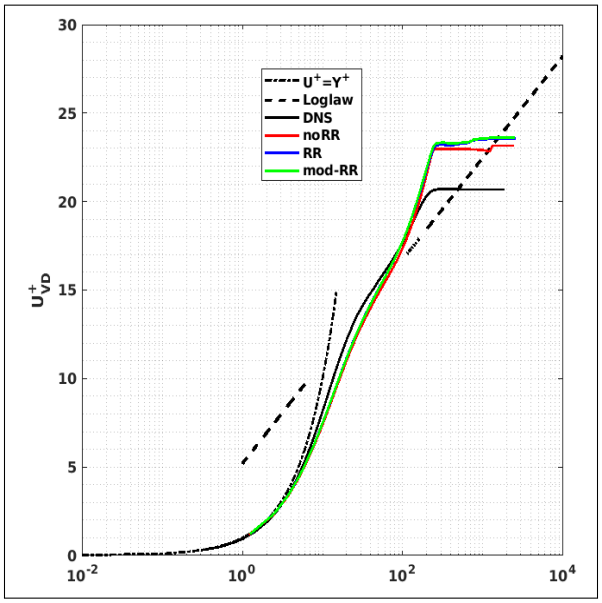


Fig. 2.15. Effect of recycling on streamwise velocity normalized by friction velocity vs wall-normal coordinate (inner scaling)

2.4.3.1 Advantages and Disadvantages

The previous section showed a fair estimation of the RANS simulation with/without recycling methods. However, there are some important reasons why RR methods are helpful and useful in RANS simulations that the conventional method.

- The Rescaling-Recycling (RR) method is computationally much cheaper to simulate a turbulent boundary layer simulation. The cost of calculation of a flat plate boundary layer increases several folds when it is necessary to extract useful information from locations far from the leading edge of the plate.

- This method of rescaling eliminates the presence and production of laminar-boundary layer transition during simulation of flat plate boundary layer simulations. The location of transition can affect the solution downstream due to the inability of the turbulence model to accurately capture the physics associated with transition.

- The usage of RR method helps in a cost-effective preparation of fine-grid results which will essentially be the main source of training data for the machine learning models discussed in Chapter 4.

- When RANS results are used directly as training data for machine learning without any rescaling, it poses a problem due to the presence of stagnation point. There is a spike in the values of wall-shear, heat flux, pressure and other turbulence values near the stagnation point and makes it difficult for machine learning models to adapt to such regions, in addition to the formulating a model for the overall structure of the flow domain.

One important concern regarding the RR methods is the presence of a small region in the streamwise direction from the inlet which is affected due to the rescaling of downstream quantities to the inlet boundary. This is observed even in DNS simulations (Priebe [22], Xu [42]) and is essential to discard these values when extracting and analyzing useful information from the CFD solution. However, the advantages

associated with using the RR methods outweighs this minor fact making it a useful tool to generate high-quality fine grid results for machine learning.

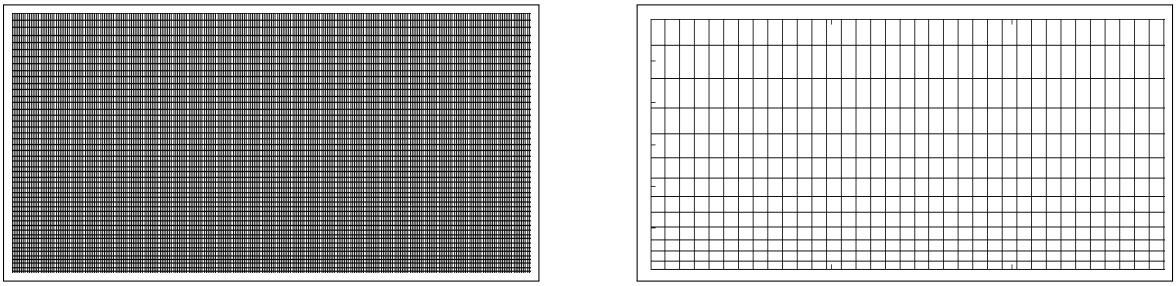
2.5 How does the grid size affect the CFD solution?

The purpose of this section is to identify the problems in using coarse grids and illustrate the need to develop a robust method to tackle the problems and help in accurately predicting the results for inexpensive meshes. Computational cost is extremely important when simulations are done at an industrial level to help in solving complex fluid flow simulations efficiently.

2.5.1 Grid preparation

This section describes the method of generation of coarse grids for problems concerning flat plate boundary layer flows. Since the flow domain under consideration (see Figure 2.1c) is a rectangular domain, there is a need to properly create coarse-grids which preserve the aspect ratio, similar to the generation of coarse meshes for a square flow domain. To construct a boundary layer mesh in the wall-normal direction, there are two important parameters to specify namely Distance of first grid point (dS) and Growth Rate (GR).

The mesh in wall-normal distance is composed of a fine boundary layer mesh which is blended with a uniform mesh in the far-field region. This is to reduce the computational cost, since the variation in solution in the outer region is minimal. The points in a boundary layer mesh are constructed as a geometric progression with points ($y_1 = dS, y_2 = dS \cdot GR, y_3 = dS \cdot GR^2 ...$) up to a distance of 10mm, a value chosen to fully enclose the boundary layer for the entire domain. The stream-wise direction consists of points corresponding to a traditional uniform grid.



(a) Fine **(b)** Coarse

Fig. 2.16. Illustration of a boundary layer mesh in the wall-normal direction (Y-axis)

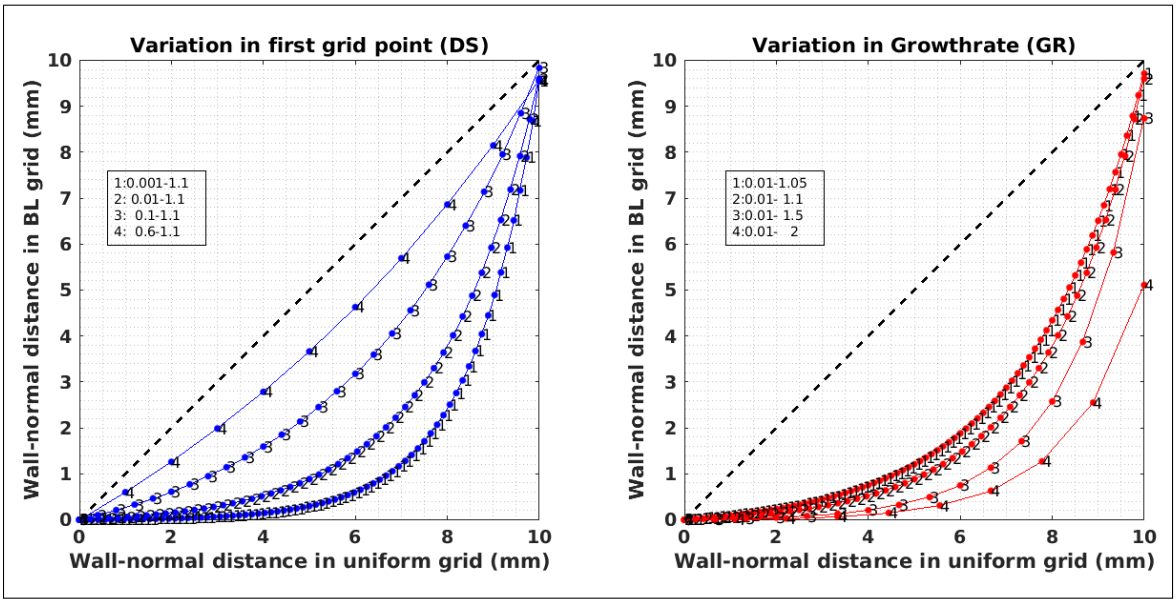


Fig. 2.17. Effect of boundary layer parameters in the construction of a mesh. (a) Variation in Distance of first grid point (dS) with fixed GR = 1.1 (b) Variation in Growth Rate (GR) with fixed dS = 0.01mm

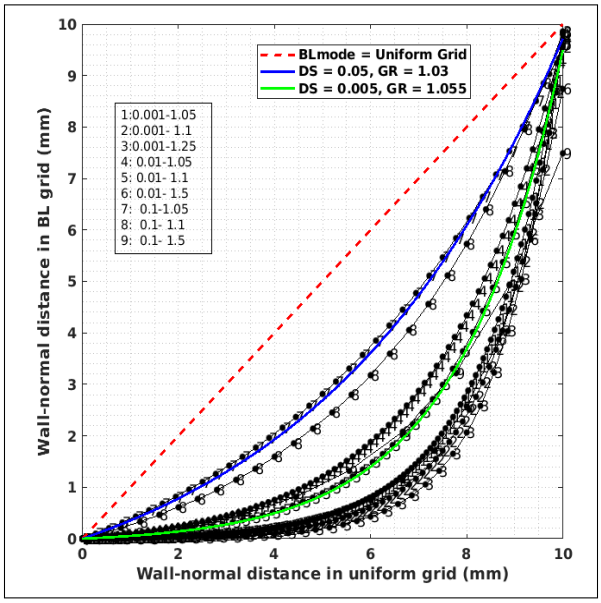


Fig. 2.18. Selection of a boundary layer parameters to pick a reference equation. — Reference trajectory equation used for construction of various grids

Figure 2.18 shows the trajectory equation used in constructing coarse grids. The strategy of generating a fine/coarse mesh is with a multiplication factor 'P'. The number of points in the base mesh $[N_x \text{ x } (N_{BL} + N_{outer})]\cdot P$ is given by $[150 \text{ x } (66 + 30)]\cdot P$, where

$$P \equiv \begin{cases} > 2, & \text{Fine-grids} \\ 1, & \text{Baseline case} \\ < 1, & \text{Coarse-grids} \end{cases} \tag{2.12}$$

Figure 2.19 illustrates the method of determining the points for constructing a coarse mesh. The reference equation is chosen to have $dS = 0.05$ and $GR = 1.03$. To determine the location of points, the number of points calculated using the base mesh (with P and N_{BL}), seen as points from a square mesh are mapped on to the trajectory to generate the spacing for the new boundary layer mesh.

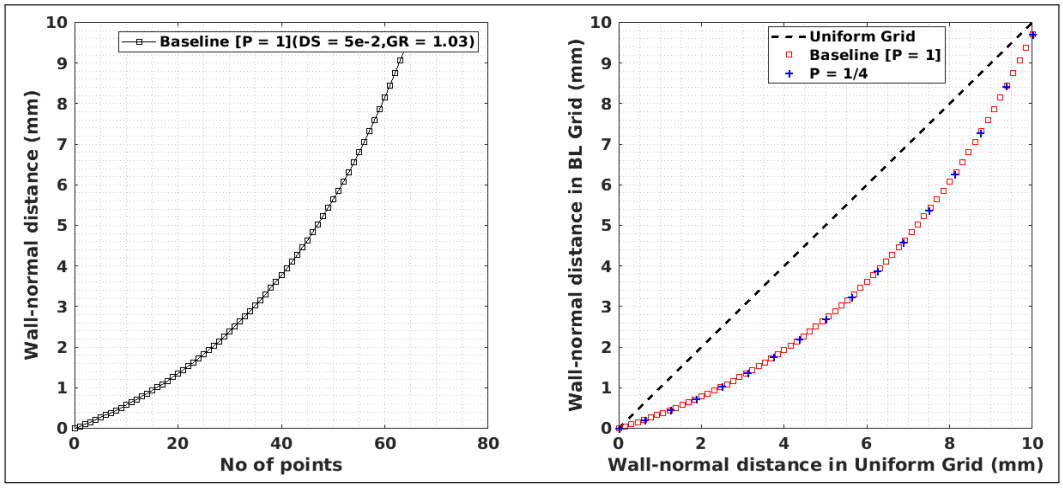


Fig. 2.19. Generation of coarse grids using a reference trajectory using mapping (a) Reference equation trajectory using dS = 0.05 and GR = 1.03 (b) Construction of points for a coarse grid (P=1/4)

2.5.2 Overview of grids for analysis

Table 2.4 gives the set of meshes constructed and taken into consideration for computational analysis and performance in prediction of wall-shear and other important variables for hypersonic flow conditions. The test case with $P = 2$ is considered to be the fine-case simulation and the other test cases serve as coarse meshes with the coarsest mesh $P = 0.125$ to have a size $(1/16)^{th}$ relative to the fine-grid.

Table 2.4. Number of points in grids using reference trajectory (dS = 0.05 and GR = 1.03). N_x, N_{BL} and N_{outer} represent number of points in streamwise, boundary layer and uniform grid region in the wall-normal direction respectively.

Grid multiplier (P)	$\boldsymbol{N_x}$	$\boldsymbol{N_{BL}}$	$\boldsymbol{N_{outer}}$
2	300	132	80
1	150	66	40
0.5	75	33	20
0.25	38	17	10
0.167	25	11	7
0.125	19	8	5

2.6 Performance of RANS models on coarse grids

This section provides an qualitative analysis on the performance of grids on the RANS simulation of test conditions represented by the DNS studies. The grids constructed using the approach outlined in the previous section is used in conjunction with the modified-RR method to calculate the solution of a flat plate hypersonic turbulent boundary layer. Figure 2.20 clearly indicates the gradual deterioration in prediction of overall boundary layer growth structure and there is a trend of over-prediction of the boundary layer thickness (δ). Figure 2.21 is an excellent representation of how the values of wall-shear are not predicted accurately when the solver is subjected to variation in grid sizes. We can also see a trend in under-prediction of the values of heat fluxes which are also important in the analysis of hypersonic flow.

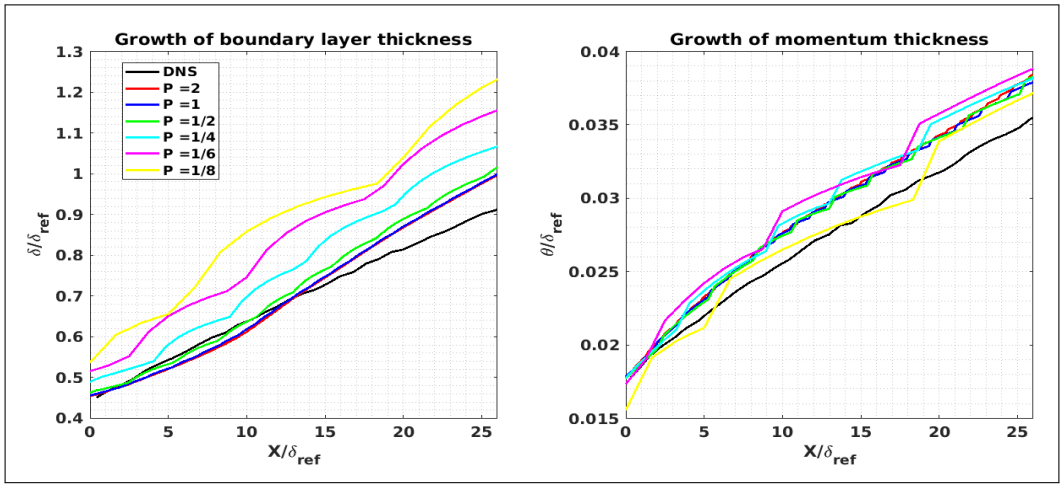


Fig. 2.20. Effect of Grid sizes on growth of boundary layer (a) Boundary layer thickness (δ) (b) Momentum Thickness (θ)

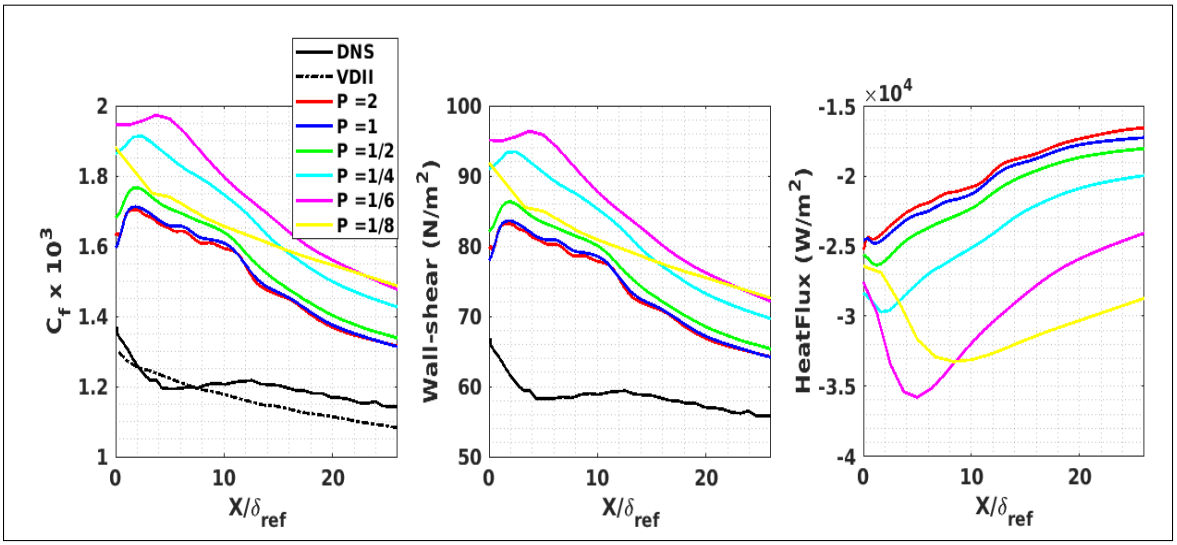


Fig. 2.21. Effect of Grid sizes on the prediction of (a) Skin-friction (b) Wall-shear (c) Heat-Flux on the wall surface

Figure 2.22 shows a clear trend in the turbulence profiles illustrating that the poor prediction in wall shear stress manifests itself as a trend of decreasing r.m.s fluctuations and shear stress profiles. Figures 2.23a and 2.23b also show a trend from the fine-grid cases, owing to the changes in wall-shear. The analysis of these plots and Table 2.5 indicate the poor prediction of wall-shear which is caused by the values

of gradient at the first grid point is responsible for variation in turbulence profiles with increasing mesh sizes.

Table 2.5. Number of points in each sub-layer of turbulent boundary layer for different grid sizes

Grid Multiplier	**Number of points**				**Wall properties**	
P	$y^+ < 5$	$5 < y^+ < 30$	$30 < y^+ < 300$	$y^+ > 300$	$\tau_w(N/m^2)$	$u_\tau(m/s)$
2	4	17	84	26	66.25	65.8
1	2	8	42	13	67.17	66.42
1/2	1	4	20	7	68.08	68.19
1/4	0	2	10	4	74.57	72.4
1/6	0	1	7	2	81.62	75.88
1/8	0	1	4	2	80.57	77.1

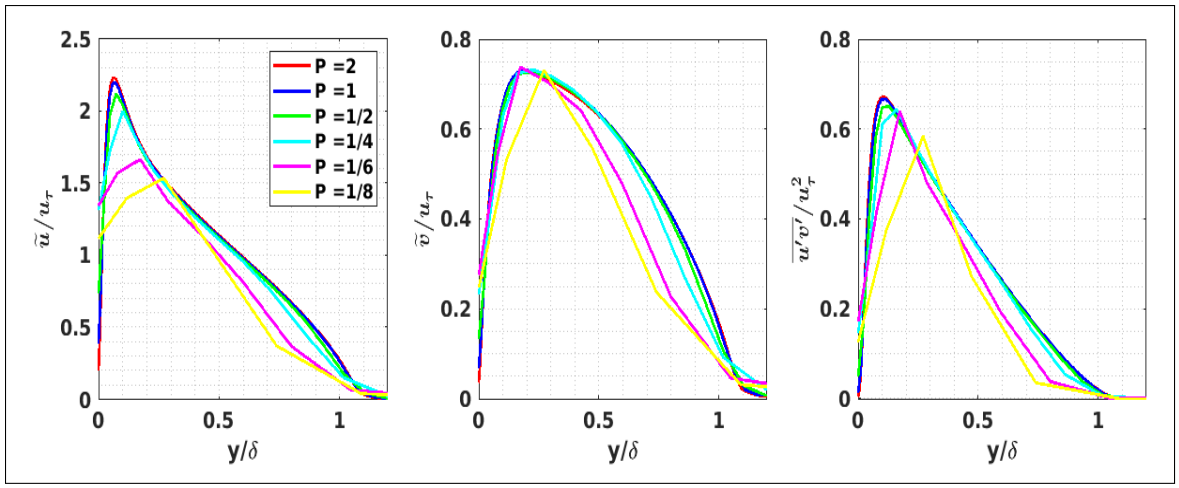
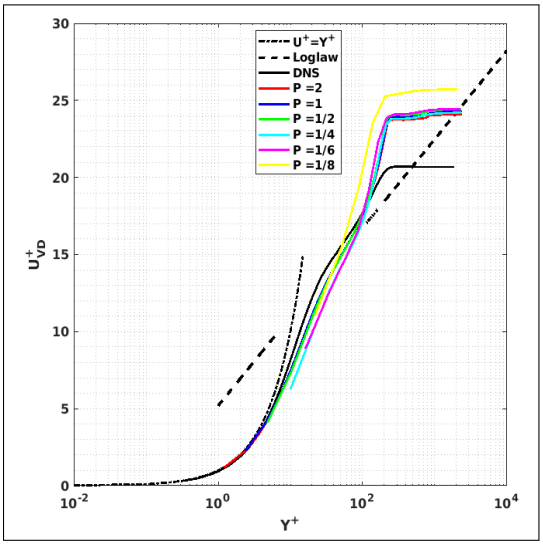
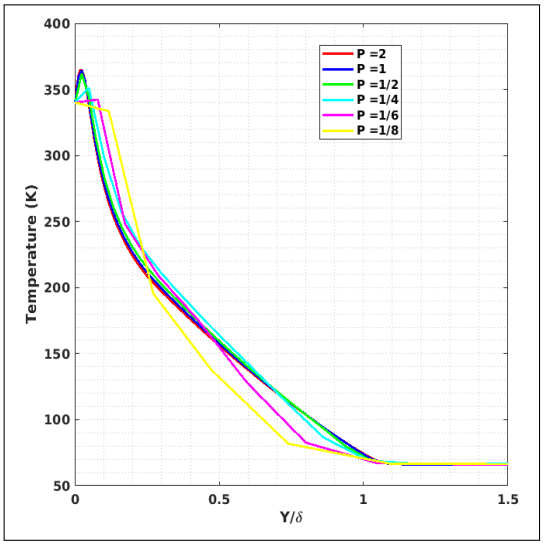


Fig. 2.22. Effect of Grid sizes on turbulence profiles scaled by friction velocity (u_τ) (a) Stream-wise and (b) Wall-normal r.m.s velocity fluctuations (c) Reynold's shear stress



(a) Stream-wise velocity scaled by u_τ vs y^+ (inner scaling) at $Re_\theta = 3300$



(b) Temperature inside boundary layer at $Re_\theta = 3300$

Fig. 2.23. Effect of grid sizes

In the forthcoming chapters, we will illustrate a method using machine learning by utilizing the fine-grid ($P = 2$) as the training data to help in the correction of gradient at the first grid point of the mesh, which in turn ensures proper estimation of wall shear for larger size grids.

3. MACHINE LEARNING - A NEW TOOL FOR SOLVING PROBLEMS

Machine learning is a field of study which helps machines (or computers) to mimic problem solving abilities and learning as exhibited by a human mind. A concise definition by Tom Mitchell states that (Andrew Ng [23]) :

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E".

There is a clear distinction in the way of defining a traditional classical algorithm and a machine learning algorithm. Classical algorithms are programmed methodically with exact rules to complete a particular type of task. On the other hand, machine learning algorithms are given general guidelines that define the model, along with a specific dataset. This data should contain information to create a model to complete that specific task. Essentially, we try to adjust the model to fit the data, or that "the model has to be trained on the data" (Rocca [26]). In general, machine learning is incredibly useful for difficult tasks when we have incomplete fields or information that is too complex to code manually. In these cases the model "learns" the missing information that it needs by itself.

3.1 Types of machine learning techniques

Machine learning techniques are divided into two categories Supervised Learning and Unsupervised Learning. Supervised learning works by building a relationship between the input and the output features of a given problem, however Unsupervised learning try to infer the natural structure present in the data. Supervised learning is used to predict the value of dependent variable using the model parameters. For

example fitting a curve through a set of points is the simplest regression problem. In a 2-D space, to fit a 3rd degree curve through a set of points the model formulation will be given as

$$y = a_0 + a_1 x + a_2 x^2 + a_3 x^3. \tag{3.1}$$

A model is fit to the given data-set thereby determining model constants a_0 to a_3 for predictions with new data.

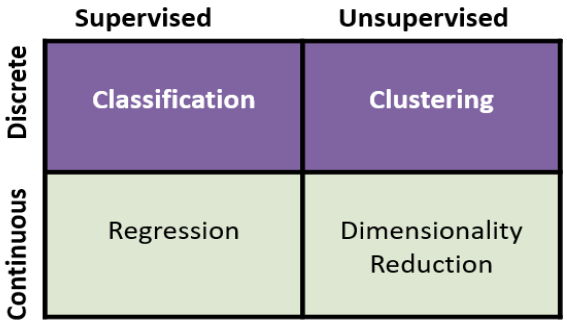


Fig. 3.1. Types of machine learning problems

Figure 3.1 represents a map of machine learning problems. Since engineering simulations deal with continuous data and prediction of parameters rather than labeling, numerous techniques are implemented to solve regression problems. Decision trees help in solving problems where conventional mathematical functions cannot fit the variation in data corresponding to non-linear equations.

3.2 Decision Trees and Ensemble learning

An important category of machine learning techniques which are widely used in complex regression problems are Decision trees. Computational Fluid Dynamics (CFD) simulations are solutions of complex non-linear partial differential equations and hence the method of linear or logistic regressions are not a viable or practical to predict the solution of these equations. Decision trees are extremely useful in calculating variables which involve complex relationships with the parameters.

Figure 3.2 shows the structure of a regression tree for a single parameter variation of drug effectiveness with drug usage. There are three main components of a decision tree namely Root, Node and Leaf. The first decision of the tree is termed as Root and the subsequent subtree decisions are called Nodes. The terminal decision at the end of each subtree where it does not connect to any other decision is called the Leaf or Terminal Node. Decision tree splitting is based on an important quantity called the Residual Sum of Squares (RSS).

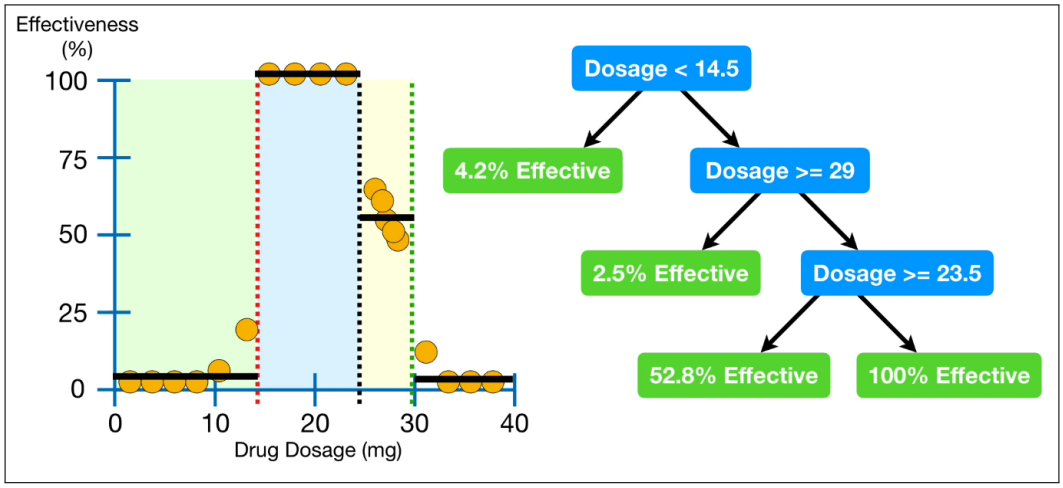


Fig. 3.2. Regression tree for a single parameter dataset [Root, Node = Blue and Leaf = Green] (Starmer [32])

Residual Sum of Squares (RSS)

The Residual sum of squares (RSS) is an important parameter which decides the location of split and the decision which characterizes every node on a decision tree.

$$R_1(j,s) = \{X|X_j \leq s\} \quad \text{and} \quad R_2(j,s) = \{X|X_j \geq s\} \tag{3.2}$$

$$RSS = \sum_{i:x_i \in R_1(j,s)} (y_i - \overline{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \overline{y}_{R_2})^2 \tag{3.3}$$

The decision tree algorithm works by splitting the data into two regions R_1 and R_2 to the left and right of each true data. From Eq.(3.2) and Eq.(3.3) the value of RSS

is calculated for each independent variable in the dataset. The root and the nodes of subsequent trees are decided by finding the variable which produces the *least residual sum of squares*. This process is done recursively to split the regions determined in the parent tree to produce subtrees.

A machine learning model will have high variance if it fits the training data perfectly and has poor performance with new data. Tree Pruning is a method by which decision trees stop creation of extremely dense trees which over-fit the data. This is done by stopping the splitting process if a region contains a certain maximum number of true data. The final values of the leaf node is always calculated by the average of true values in that specific subregion of that subtree. For datasets with multiple parameter variations, the decisions for the nodes are calculated by taking the least value of RSS amongst the values considering each feature separately.

Ensemble Learning use multiple learning techniques/algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. The usage of a combination of decision tree ensembles increases the flexibility in the functions which they represent. This amount of flexibility is crucial in reducing over-fitting of data, which is a common problem in machine learning. The next section discusses the most important type of ensemble learning methods used for prediction of wall-shear in simulating Computational Fluid dynamics(CFD) problems.

3.3 Random Forest approach

Random forests or random decision forests are an ensemble learning method which combines multiple decision trees in a single framework and produce an output which is the average of decision trees (regression) or the mode of the tree-classes (classification). A random forest can be used in a regression problem by constructing $\{h(x, \theta_k), k = 1, ...\}$ where the $\{\theta_k\}$ are independent identically distributed random vectors and k unique collection of regression-trees, where the final decision is the

mean of the individual subtrees at a specific input x (Breiman [3]). Individual tree classifier or predictor will have only an accuracy only slightly better than a random choice of class. But combining tree predictors can produce improved accuracy and this becomes the basis of any random forest model. Two important forms of random forests are Bagged Aggregation method and Boosted Random forest method. The boosted version involves bagged estimations but also includes weights for each decision tree and dynamically uses a learning parameter to adjust the weights with the creation of every new subtree. A brief description of these models will be given in the next sections.

An important feature of the random forest (bagged models) is the ability to prevent over-fitting. Random forest models employ the concept of *Strong Law of Large numbers* from probability where, the mean of the trials converge towards the expectation. Extrapolating this concept to machine learning, Breiman [3] proves that as the number of trees are increased in building a random forest, the final prediction using the subtrees always converges and just produces a limiting value of the generalization error given by

$$E_{X,Y}(Y - h(X))^2 \tag{3.4}$$

where Y is the true value, $h(X)$ is the hypothesized model and E denotes the generalization error with the true value. In conclusion, a random forest combines several decision trees which are built separately using the concepts of bagged aggregation and produces the mean of constructed subtrees as the final prediction for some input.

3.3.1 Bagged vs Least Squared Boost

Random forests uses two key concepts which makes this method more reliable for accurate predictions (Koehrsen [13])

1. Creation of bootstrapped dataset for each tree.

2. Selection of random features for each tree.

Decision trees are highly dependent on the type and quality of data they are trained and the generation of trees vary with changes in the training dataset. Random forests creates a bootstrapped data by creating a random subset of input from the training dataset with replacement. This concept of drawing random samples with replacement is called Bootstrapping or Bagging which enables samples to be duplicates when building a specific subtree (Yiu [43]). The concept of bagging helps to reduce overall variance which essentially denotes a model which can predict training set perfectly but performs poor with new data.

The bagging algorithm works by generating new set of data from a primary set S by drawing random samples with replacement for each decision tree. For large sizes of S, each subset S_i for individual trees i is expected to have the fraction $(1-1/e)(\sim 63.2\%)$ of the unique samples of S and the rest being duplicates (Javed [1]). The generation of each subtree also involves sampling random subset of features for a multivariate dataset which helps in improving the flexibility of the model. Finally the bootstrap dataset is combined by taking the mean of the predictor values of subtrees in the forest. The minimum number of observations per leaf is 5 to implement tree pruning and the number of trees is set to 100 for balance of computational cost and predictive accuracy. A commercial computational and data analysis tool- MATLAB is used for generating a ML model using the random forest algorithm by feeding a fine-case training data-set using the methods described in Chapter 2.

Boosting is a modification done to bagged aggregation algorithm where weights are added to individual learning trees. The boosting method improves the prediction of each subtree by using a parameter to learn the difference between observed prediction and aggregated prediction of all subtrees which are grown in the previous iteration ([25]). This method fits to minimize the mean-squared error given by Eq. (4.7) in the next chapter. The level of learning can be adapted by

$$y_n - P_{learn} f(x_n) \tag{3.5}$$

where y_n is the observed response, $f(x_n)$ is the aggregated prediction from all weak learners grown up to a particular iteration for observations x_n, and P_{learn} is the learning rate. The final prediction from the boosted model is a weighted average of the individual decision where some trees might be favored more than other.

Advantages and Disadvantages

- Random forests use bagging to reduce the variance of individual trees and use a multitude of decision trees for the final prediction. This algorithm provides an optimum balance between making ineffective training data predictions (high bias) and inability of the model to adapt to generalized data (high variance).

- Linear regression problems often associate with renormalization to fit the ranges of different features within a suitable limit to extract maximum performance. Since Random forests work with decisions rather than the actual values of data, the individual ranges of data become irrelevant.

- One of the biggest advantages of random forest is its ability to circumvent extreme over-fitting due to the nature of its methodology. Also, tuning inherent hyper-parameters can also improve the result, however the default methods provide enough accuracy for general tests.

- The introduction of large number of trees makes this algorithm difficult to make fast real time predictions. The computational time is affected more for new predictions than for training. The adjustments in properties like leaf size, number of trees and the provision of rich data-set alleviates these minor problems to create a robust model for CFD simulations to dynamic problems.

The next chapter will see the modeling approach of the gradient correction factor using MATLAB, which essentially is used in the prediction of wall-shear for coarse meshes.

4. MACHINE LEARNING FOR HYPERSONIC BOUNDARY LAYERS

This chapter develops from the idea of machine learning discussed in Chapter 3, to help in the formulation of a tangible machine learning model to correct the value of wall-shear stress. Section 2.6 shows a clear trend in the deterioration of predictive accuracy when the grid size is gradually increased to computationally coarser grids. When the size of the mesh increases, the solver cannot predict the gradients (velocity and temperature) due to the large distance of the first grid point from the wall. The purpose of Chapter 2 is to help in the generation of training data which will be used to create a robust ML model.

4.1 Modeling Approach

The modeling approach constitutes the determination of a gradient correction factor at each streamwise location using the value of wall-shear from the fine case as a reference. As the mesh becomes more finer, the effect of wall-functions are irrelevant and the value of wall shear is represented as a finite difference formulation:

$$(\tau_w)^{fine} = \mu_w \left.\frac{dU}{dy}\right|_w = \mu_w \frac{U_1 - U_0}{y_1 - y_0}, \tag{4.1}$$

where U_1, y_1 are the values of velocity and wall coordinate at the first grid point and U_0, y_0 represent the values at the wall node. Another advantage of modeling the gradients near the wall, is to eliminate the calculation of RANS solutions using traditional wall function models which are highly dependent on type of grids and do not cater to all type of flow conditions.

4.1.1 Gradient Correction Factor

$$\beta(x) = \frac{(\tau_w(x))^{fine}}{(\tau_w(x))^{coarse}} \tag{4.2}$$

$$\beta^{true}(x) = \frac{(\tau_w(x))^{fine}}{\mu_w \frac{U_i(x)}{y_i(x)}} \tag{4.3}$$

where $\beta(x)$ is the velocity gradient correction factor at a particular stream-wise location x, U_i, y_i are the series of imaginary first grid points in the wall-normal direction inside the mesh structure of a fine-grid case for each streamwise location. From the no-slip condition, $U_0 = 0$ and $y_0 = 0$ at the wall for each streamwise location. Eq. (4.3) shows the modeling structure for obtaining the correction factor, where the series of points chosen vary from distance of first grid point of fine-grid (P=2, dS = 0.0246mm) to the coarsest-grid under consideration (P=1/8, dS = 0.52mm). From Eq. (4.3) it is easy to understand that the gradient correction factor (β) will be used as a multiplicative parameter to the value of wall-shear for coarse grids. The gradient correction factor at each x is multiplied with the wall-shear to predict the correct value of shear stress at the wall which matches with fine-grid results.

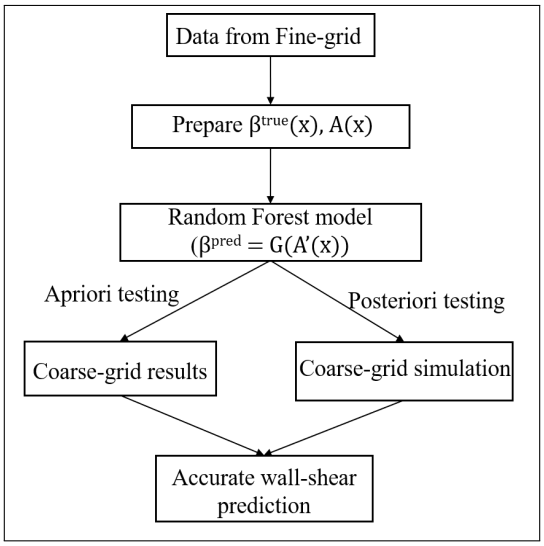


Fig. 4.1. Outline of Machine learning methodology and implementation

Figure 4.1 illustrates the steps involved in building and using machine learning to predict the wall-shear of coarse grids. After obtaining fine-grid data the next step involves creation of a dataset which includes the calculation of β^{true} from Eq. (4.3) and corresponding interpolation of all other variables (flow,turbulence and properties) for each specific location. This exhaustive dataset is fed into a Random-Forest (RF) model implementation which gives us a model to utilize certain variables from the data-set to generate the correction factor based on the variables given by β^{pred}. The ML-corrected value of wall-shear is given by

$$(\tau_w)^{fine} = (\tau_w)^{coarse}_{ML-corrected} = \beta^{pred} \cdot (\tau_w)^{coarse} \tag{4.4}$$

and

$$\beta^{pred}(x) = G(A'(x)) \tag{4.5}$$

where $A'(x)$ is a subset of the exhaustive set of all variables (A(x)) from a fine-grid RANS solution given to the ML model (G(x)) by the user.

Table 4.1. Data collection strategy for training ML model using fine-grid RANS simulations

$\boldsymbol{\beta^{true}}$	$\mathbf{y}$	$\mathbf{X}$	$\boldsymbol{\delta}$	$\boldsymbol{\varepsilon}$	**A,..**
β_{11}	y_{11}			ε_{11}	$A_{11},..$
.	.	X_1	δ_{X_1}	.	.
β_{1m}	y_{1m}			ε_{1m}	$A_{1m},..$
β_{21}	y_{21}			ε_{21}	$A_{21},..$
.	.	X_2	δ_{X_2}	.	.
β_{2m}	y_{2m}			ε_{2m}	$A_{2m},..$
...	...	...	...	...	.,..
β_{n1}	y_{n1}			ε_{n1}	$A_{n1},..$
.	.	X_n	δ_{X_n}	.	.
β_{nm}	y_{nm}			ε_{nm}	$A_{nm},..$

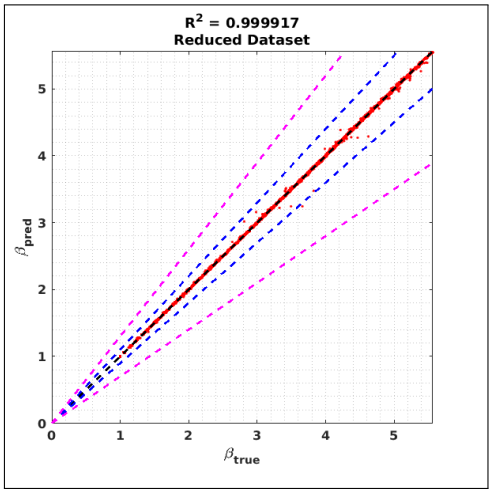
Eq.(4.4) and Eq.(4.5) illustrate the method of correcting the wall shear values for coarse grid cases after modeling the correction factor using machine learning. Table 4.1 shows the method of collecting data as a matrix which will be fed to the machine learning model for training. Here, β^{true} is the true value of gradient computed using Eq. (4.3), m is the number of representative wall-normal coarse grid locations interpolated, n is the number of streamwise locations, X is the streamwise location, δ is the corresponding boundary layer thickness, y is the set of 'M' interpolated wall-normal distances, ε is the interpolated dissipation rate at each streamwise location and wall-normal location and A represents the set of all other solution variables from the RANS simulations.

4.1.2 Training ML model

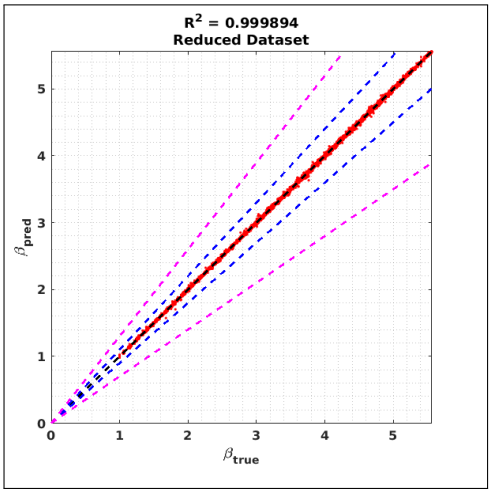
Data collected in the structure given by Table 4.1 is used to feed to the random forest model using MATLAB, a commercial mathematical software used for data analysis and computational work. An important step in machine learning is to choose the number of parameters to be selected as independent variables. The objective is to maximize the predictive accuracy while choosing the minimum number of variables. Variables taken into consideration are

- General $(\beta, X, y, \delta, \theta)$
- Flow and Thermodynamic $(U, V, P, T, \rho, ..)$
- Turbulence $(k, \varepsilon, \overline{u_i' u_j'}, \mu_t ...)$
- Properties $(Cp, Pr, \gamma, \alpha, ...)$

We can observe from Figures 4.2a and 4.2b the usage of all parameters produce excellent correlation with the true value of wall-shear. It should also be noted that the bagged aggregation model reduced the spread of data which reflects in higher value of R^2.



(a) Bagged RF model



(b) Least-Squared Boosted RF model

Fig. 4.2. Comparison of β^{true} with ML-predicted β^{pred} using all parameters, **- -** Line of equality, **- -** 5% error, **- -** 10% error

4.1.2.1 Greedy Algorithm for selection of parameters

The method of selection of parameters uses the Greedy Algorithm (Hiremath [8]), where parameters are selected in each iteration until there are no more parameters which fit a particular selection criteria. The method of selection is based on two important parameters:

- Coefficient of determination or R-squared (R^2)
- Mean squared Error (MSE)

If $(\beta_i)^{pred}$ is the predicted value using ML model and $(\beta_i)^{true}$ is the true value of correction factor obtained from Eq. (4.3) then,

$$SS_{res} = \sum_{i}^{nm}((\beta_i)^{true} - (\beta_i)^{pred})^2, \qquad SS_{tot} = \sum_{i}^{nm}((\beta_i)^{true} - \overline{\beta_i}^{true})^2 \tag{4.6}$$

and

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}, \qquad MSE = \frac{SS_{res}}{n} \tag{4.7}$$

where $\overline{\beta_i}^{true}$ is the mean of true values, nm is the total number of data points.

The greedy algorithm works by initializing two sets, Set {G: Good-set} as empty and Set {A: Set of all variables} which contains all the parameters in the dataset.

1. Iterate through the set of all parameters to find the parameter which records the highest R^2 or lowest MSE value when training the dataset with the chosen parameter.
2. The chosen parameter is removed from Set A and transferred to Set G while recording the R^2 value and this completes one stage (iteration).
3. The next stage, follows the same procedure by choosing the parameter from Set A, and run the ML model along with the previous parameter chosen in Set G.
4. The parameter is selected if the R^2 of this stage is greater than the previous stage. The stage is completed upon recording the new updated R^2 for comparison.
5. This operation is repeated until the Set A is empty or the addition of parameters does not change the R^2 or MSE value from the previous stage.

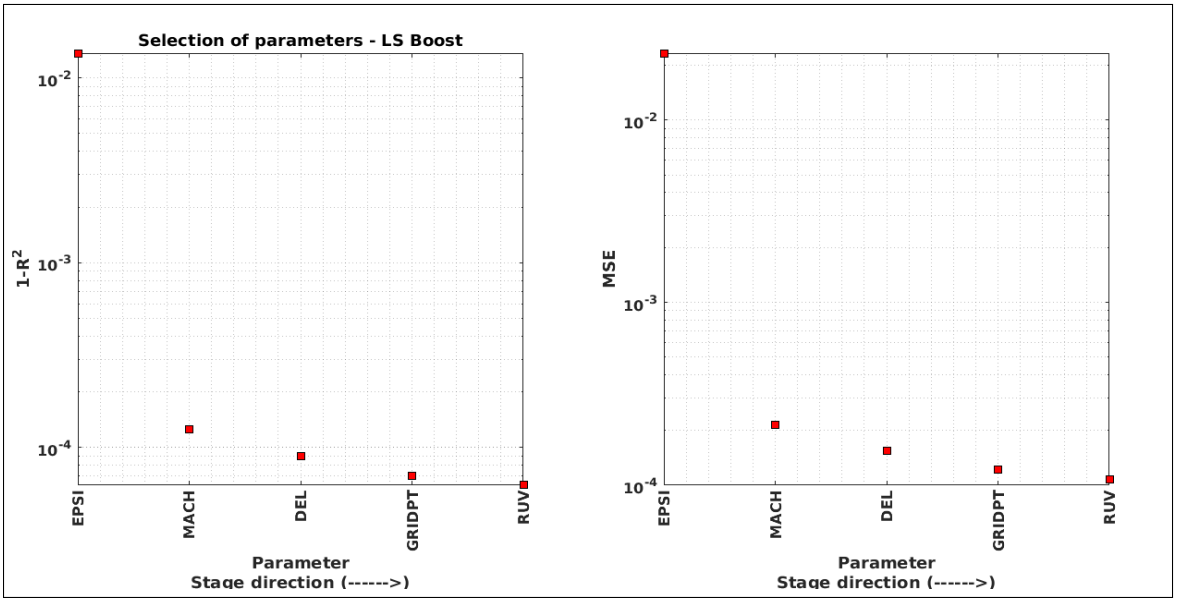


Fig. 4.3. Selection of parameters using Greedy Algorithm

Figure 4.3 shows the selection of 5 important parameters which influence the correct prediction of β^{true}. The selected parameters are $(\varepsilon, M, \delta, y, \overline{u'v'})$ and the low values of MSE and $1 - R^2$ after the selection of 5 parameters indicate a good fit from the constructed machine learning model.

4.1.2.2 Inclusion of neighbor cell information

There has been significant amount of research on the memory effect of turbulence, which indicates that the information calculated downstream is dependent on the value of variables upstream to that specific location. A similar methodology is discussed in this section, where the inclusion of neighboring cell information while training the data helps in understanding the flow structure and essentially help in improving the predictive accuracy of the ML model.

If there are N_P parameters chosen in a conventional method, the inclusion of neighbor cell increases the size of training data-set to $2N_P + 1$. For each parameter chosen for modeling represented by A(x), the neighbor cell model includes two extra parameters A(x-dx) and dx (small streamwise distance). Hence the training data-set becomes

$$A(x) -> (A(x - dx), A(x), dx) \tag{4.8}$$

The gain in the predictive accuracy allows us to use the neighbor cell model over the traditional method even though the cost of computation becomes marginally higher. The inclusion of neighbor cells provides a richer dataset for better prediction of the training variable.

4.1.2.3 How much data is enough data?

In machine learning based models, it is a distinctive fact that more data equates to better prediction. However the size of the data-set and the equivalent number of parameters increase the computational cost of model development and operation. So

it is necessary to find a balance in feeding the model with an optimum amount of data which is capable of producing accurate results and lowering the CPU time in ensemble learning of the decision trees. The data collection increases by choosing finer intervals of data interpolation in the wall normal direction effectively increasing the value of m in Table 4.1. The dataset can also be increased in streamwise direction by increasing n, however the increase in the length of fluid domain adds to the computational cost apart from training the ML model using the dataset. Figure 4.4 shows a saturation in the values of MSE and R^2 while increasing the value of points in the wall-normal direction for data collection. Hence an optimum range of $m = (180 - 300)$ would be sufficient to obtain a good prediction of the gradient correction factor using the parameters chosen in the ML model.

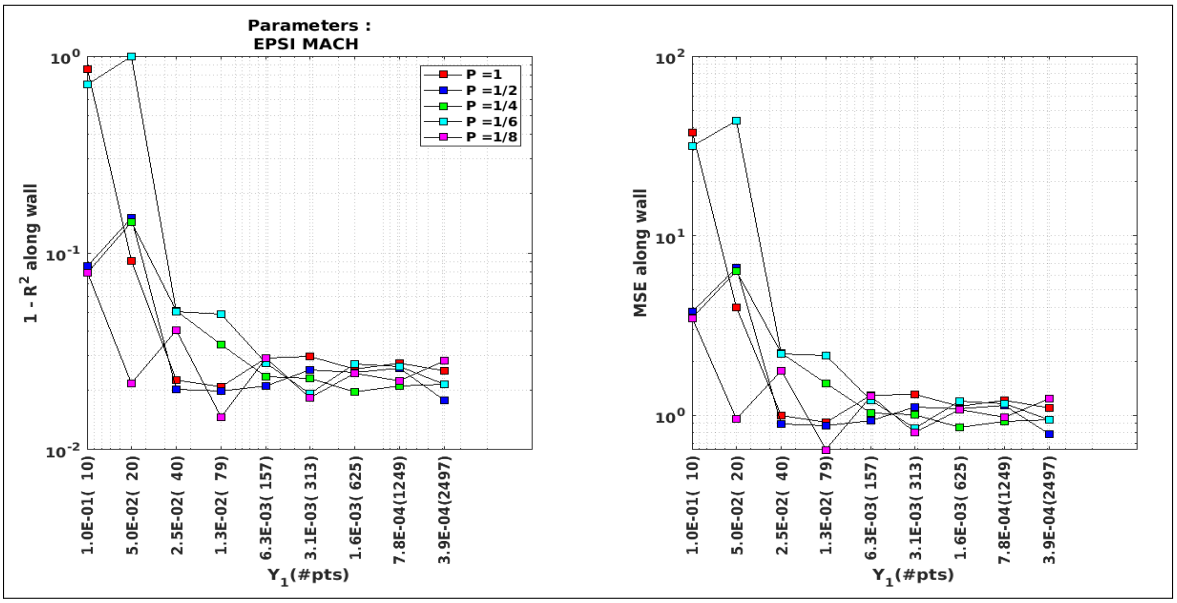


Fig. 4.4. Convergence test to understand the optimum size of training dataset. X-axis represents [Interval size(Number of pts)]

In Figure 4.4 represents the y-axis represents the size of interval in choosing the points along the wall-normal direction. The total number of points chosen for that test-case is also provided inside the parenthesis.

4.2 Results and Discussion

In line with the previous sections the parameters are chosen using the Greedy Algorithm and 4 important parameters are chosen to be sufficient to produce a good prediction of the gradient correction factors.

- Turbulent Dissipation Rate (ε)
- Mach Number (M)
- Boundary Layer Thickness (δ)
- Distance of first grid point (dS)

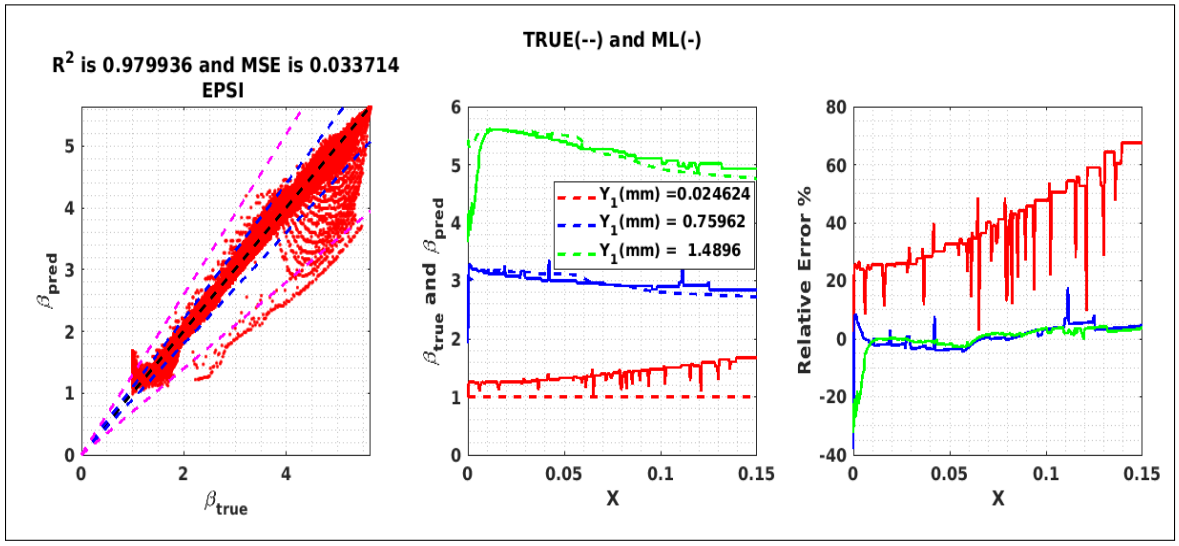
In other words,

$$\beta^{pred} = G(\varepsilon, M, \delta, dS) \tag{4.9}$$

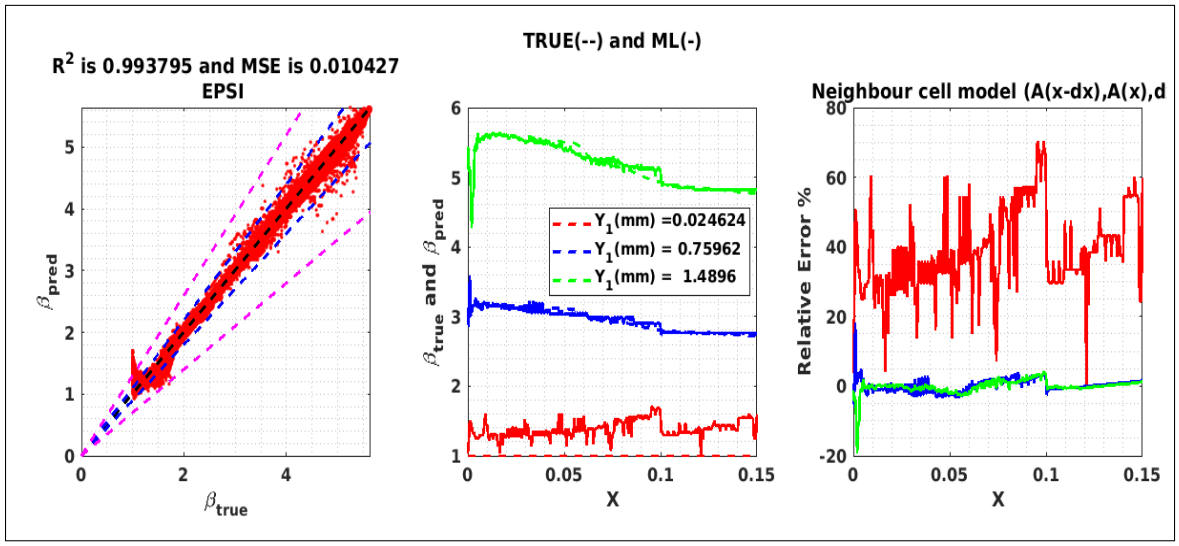
For the purpose of computational cost effectiveness, we perform A priori testing with the first two parameters, but also include a plot to compare the effect of inclusion of all four aforementioned parameters as a comparison.

The following plots clearly illustrate the importance of including the neighbor cell information. Figure 4.5b clearly has a better fitting with β^{true} with the substantial decrease in the MSE value of Figure 4.5a. The inclusion of the neighbor cell model helps in creating a better fit using the information of upstream locations for a specific streamwise-location (X).

We saw in earlier discussions that bagged aggregated model performs slightly better than the boosted model when all parameters are taken for modeling. The next two plots shows the behaviour of the two random forest modeling techniques in the prediction of first grid point gradients when only two parameters are considered i.e., $\beta = G(\varepsilon, M)$



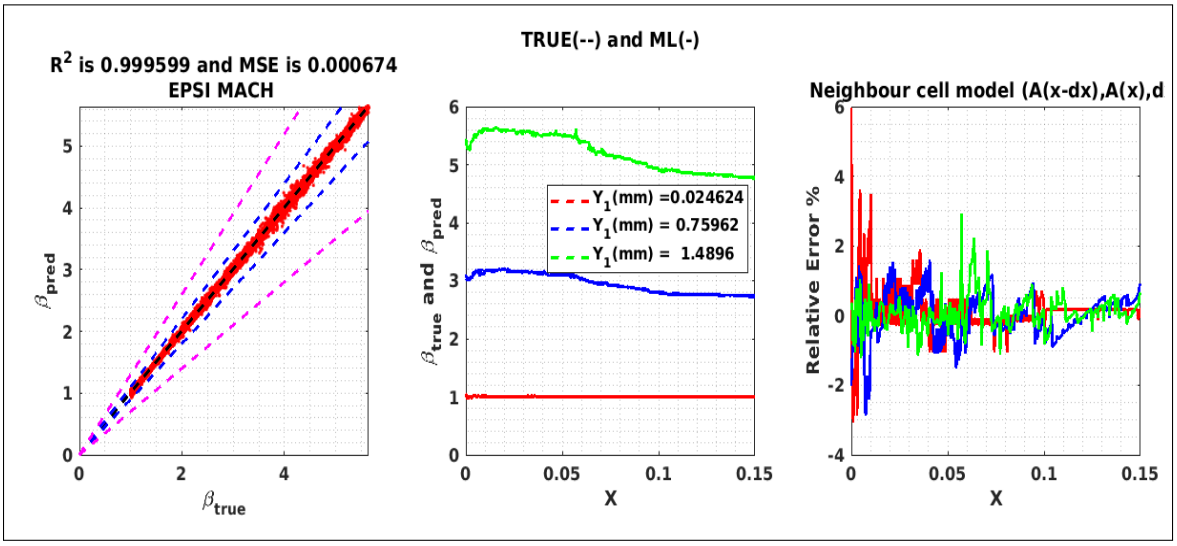
(a) General method with no neighbor cell inclusion



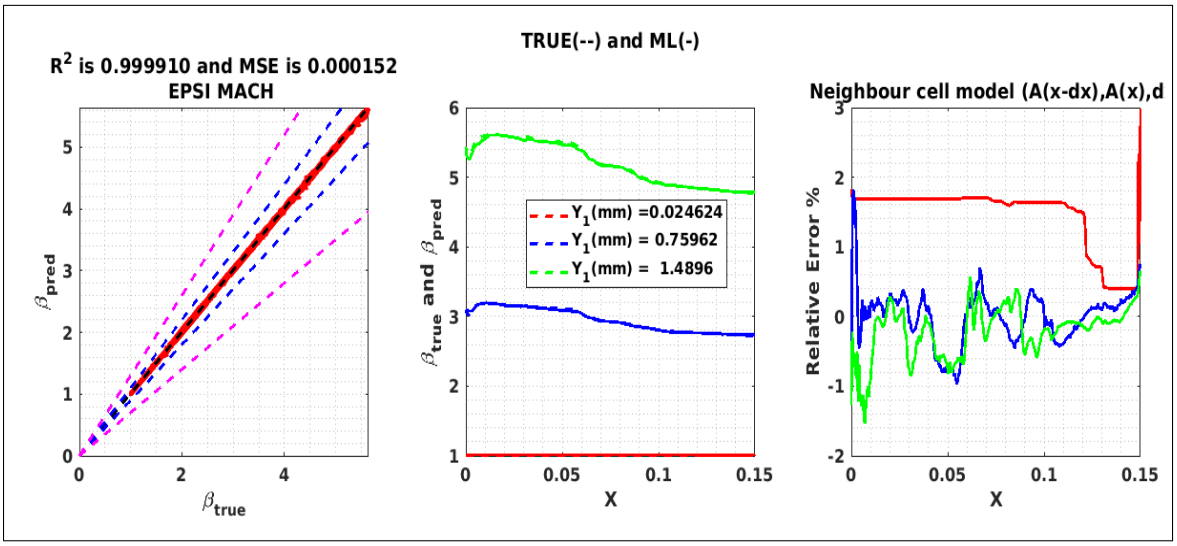
(b) Inclusion of neighbor cell information

Fig. 4.5. Effect of neighbor-cell information on a single parameter random forest model $\beta^{pred} = G(\varepsilon)$, (a) - - Line of equality, - - 5% error, - - 10% error (b) β comparison on the wall for 3 values of 'dS'

Figure 4.6b shows an improvement in the R^2 value with lower relative errors when using a bagged aggregate random forest model in comparison with Figure 4.6a.



(a) Least-Square Boosted Random forest model



(b) Bagged Random Forest model

Fig. 4.6. Different types of random forest model for 2 parameter model given by $\beta^{pred} = G(\varepsilon, M)$ with inclusion of neighbor-cell information, (a) - - Line of equality, - - 5% error, - - 10% error (b) β comparison on the wall for 3 values of 'dS'

In later sections we will show that the Least square boosted model is helpful in getting a fit for any set of parameters, but is a lot less smoother than the bagged model due to extensive weighting of individual decision trees. (see Section 4.2.1)

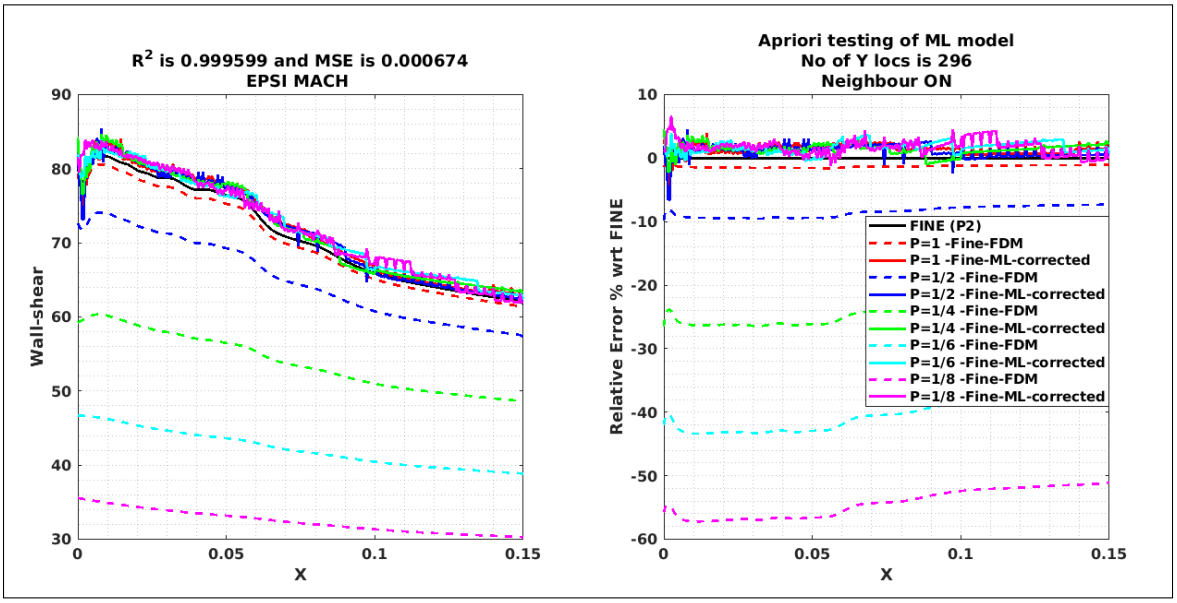
4.2.1 A priori testing of ML model on coarse grids

This section provides an A priori testing of the ML model on the interpolated coarse grid information from the training data set to see its performance on the prediction of wall-shear.

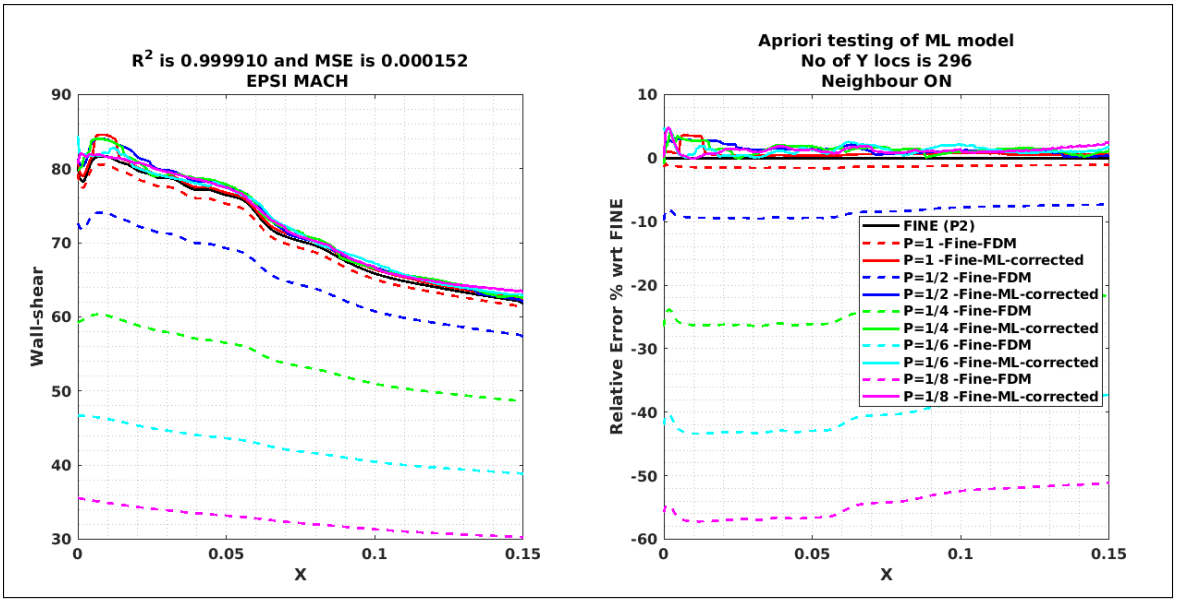
Table 4.2. Performance of ML model (Average of streamwise locations) for $\beta = G(\varepsilon, M)$

Grid multiplier (P)	$\boldsymbol{\tau_w^{COARSE}}$	$\boldsymbol{\tau_w^{ML}}$	**Relative Error reduction**
2 (FINE)	71.158	71.158	-
1	70.21	71.17	2.07
0.5	65.1	72.12	9.74
0.25	53.88	72.16	25.49
0.167	42.21	72.02	41.71
0.125	32.39	72.05	55.5

Table 4.2 shows the prediction of wall-shear (mean of streamwise locations) for coarser grids before and after machine learning. The last column depicts the also improvement in the relative error of wall-shear from coarse-grid with the usage of machine learning. A good predictive estimation for coarse grids can be made with just two parameters ($\beta = G(\varepsilon, M)$) which helps in reducing the modeling complexity. We also observe that even though Figure 4.7a can predict the trend of the wall-shear profile well, the prediction consists of a noises (small spikes) in the reconstruction of the profile using the coarse grid parameters. However, Figure 4.7b which implements the bagged aggregate model has a much smoother curve even for small number of parameters.

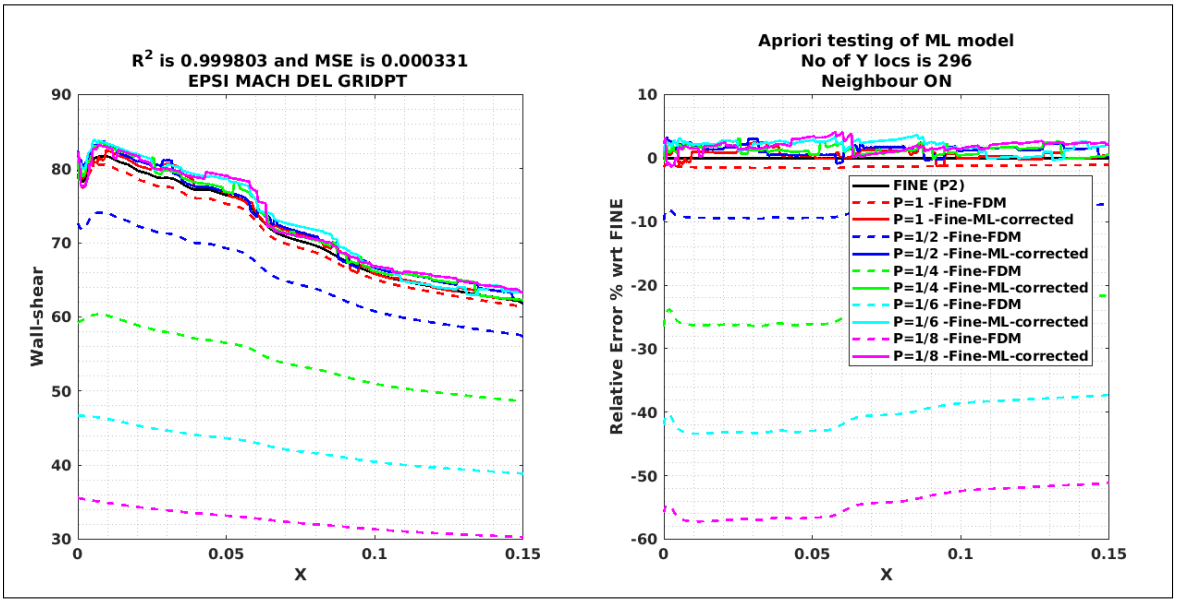


(a) Least-Square Boosted Random forest model

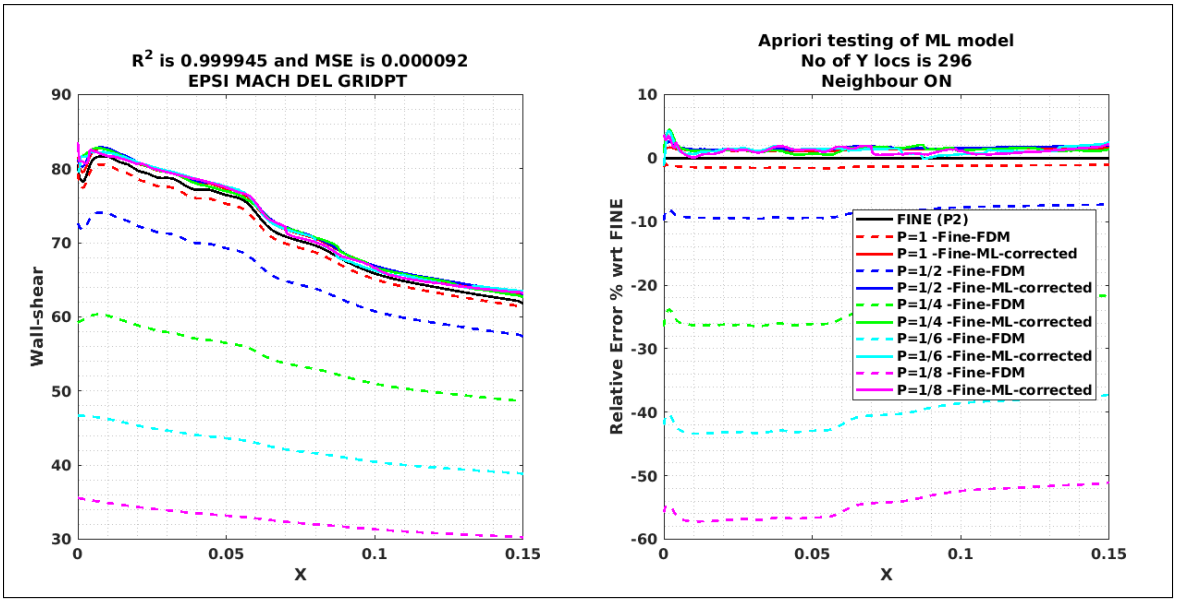


(b) Bagged Random Forest model

Fig. 4.7. Prediction of wall-shear using 2 parameter model given by $\beta^{pred} = G(\varepsilon, M)$ with inclusion of neighbor-cell information, — ML-corrected, - - Coarse-grid



(a) Least-Square Boosted Random forest model



(b) Bagged Random Forest model

Fig. 4.8. Prediction of wall-shear using 4 parameter model given by $\beta^{pred} = G(\varepsilon, M, \delta, dS)$ with inclusion of neighbor-cell information, — ML-corrected, - - Coarse-grid

The increase in number of parameters directly influences the modeling complexity but can help in accurate prediction of the dependent variable. Figure 4.8a does not show the trend of spikes during prediction of the values as seen in comparison with Figure 4.7a. The bagged model shows its superiority over the LS-boosted model in this particular type of problem with smooth curves and accurate prediction of interpolated coarse grid information from the training data set. Hence it becomes a choice of the user to use the applicable model depending on the type of flow and modeling circumstances.

These results suggest a successful validation of the model in A priori testing and gives us the confidence to utilize this model in conjunction with an in-situ simulation with any CFD solver.

5. CONCLUSIONS

This chapter summarizes this work based on the results from Chapters 2 and 4 and presents the future prospects of this research.

5.1 Results and Discussion

- We observe a difference in the values of turbulence and reynolds stress quantities between the DNS and experimental test conditions even though the flow conditions are very similar. One possible reason for this difference is the poor prediction of wall shear and other wall fluxes in RANS models which gives us the motivation for finding a viable solution.

- Two methods of rescaling are used in conjunction with the RANS simulations through an external UDF. This method has proved to be effective in reducing the computational cost of fine-grid simulations which are imperative for the machine learning model construction. Rescaling methods not only improve the computational cost, but also help in eliminating problems caused due to laminar-boundary transition when simulating flat plate boundary layer problems. The rescaling technique also helps in providing a rich dataset for training machine learning algorithms.

- The proposed machine learning methodology works by correcting the velocity gradient at the first grid point in coarse-grids using the fine-grid results as training dataset. The data set considers the memory effect of turbulence by using information from neighboring cells to produce a richer dataset. An optimal set of parameters are chosen using the greedy algorithm for the bagged random forest model which has an important attribute of eliminating data over-fit.

- We find that the model gives very good predictive accuracy in capturing the correct value of β^{pred} for different coarse grids when compared with expected value of β^{true} from fine case solutions using just two parameters (ε and M). A higher quality solution can be obtained by using 4 parameters ($\varepsilon, M, \delta, dS$) to model the gradient correction factor (β).

- The choice of the type of random forest model (bagged aggregation) or LS-boosted bag aggregation depends on the type of flow. For this type of problem discussed in this thesis, bagged aggregation method has proved to be a better predictive model than the LS-boosted algorithm. It is also indicative from the thesis that the memory effect of turbulence described in DNS simulations is used in the machine learning model which helps in improving the accuracy. Methods to determine the size of optimal dataset and finding a balance between the modeling complexity and computational cost are also important in building a robust machine learning model which can be used in any CFD solver.

- A priori testing of the machine learning model done on coarse grid points from the training dataset produce excellent improvement in the prediction of wall-shear giving vast improvements, reducing the relative error from 54% to 2% even for extremely coarse meshes (P=1/8, dS = 0.52mm).

5.2 Future Work

Due to time limitations, only A priori testing of the machine learning model was performed in this thesis. The prospective future work is discussed below:

- Posteriori testing of the proposed machine learning model by building the model inside any CFD solver and finding its total overall performance.

- Development of a machine learning model to simulate flows for all ranges (incompressible, supersonic and hypersonic) so that the model is not restricted to only certain type of problems.

- The incorporation of DNS results as training data instead of fine-grid RANS results for improving the predictions of turbulence closure models present in RANS simulations and reduce the usage of wall functions.
- Extend this machine learning model methodology to also produce accurate predictions of heat transfer in compressible turbulent boundary layer flows.

REFERENCES

REFERENCES

[1] J. A. Aslam, R. A. Popa, and R. L. Rivest. On estimating the size and confidence of a statistical audit. *Proceedings of the Electronic Voting Technology Workshop (EVT '07)*, 2007.

[2] P. Bradshaw. Compressible turbulent shear layers. *Annual Review of Fluid Mechanics*, 9(33), 1977.

[3] L. Breiman. Random forests. *Machine Learning*, 45(5), 2001.

[4] D. Coles. The law of the wake in the turbulent boundary layer. *Journal of Fluid Mechanics*, 1(2):191–226, 1956.

[5] J. Fulton, J. Edwards, H. Hassan, J. McDaniel, C. Goyne, and R. Rockwell. Continued hybrid les /rans simulation of a hypersonic dual-mode scramjet combustor. *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, 2013.

[6] N. J. Georgiadis, D. A. Yoder, M. A. Vyas, and W. A. Engblom. Status of turbulence modeling for hypersonic propulsion flowpaths. *47th Joint Propulsion Conference and Exhibit*, 2012.

[7] M. M. Gibson and B. E. Launder. Ground effects on pressure fluctuations in the atmospheric boundary layer. *Journal of Fluid Mechanics*, 86(3):491–511, June 1978.

[8] V. Hiremath, Z. Ren, and S. B. Pope. A greedy algorithm for species selection in dimension reduction of combustion chemistry. *Combustion Theory and Modelling*, 14(5):619–652, Sep 2010.

[9] J. Huang, J.-V. Bretzke, and L. Duan. Assessment of turbulence models in a hypersonic cold-wall turbulent boundary layer. *Fluids*, 4, 2019.

[10] T. Jongen. *Simulation and Modeling of Turbulent Incompressible Flows*. PhD thesis, EPF Lausanne,Lausanne, Switzerland, 1992.

[11] B. Kader. Temperature and concentration profiles in fully turbulent boundary layers. *International Journal of Heat and Mass Transfer*, 24(9):1541–1544, 1981.

[12] F. Keyes. A summary of viscosity and heat-conduction data for He, A, H_2, O_2, N_2, CO, CO_2, H_2O, and air. *Transactions of the American Society of Mechanical Engineers*, 73(7):589–596, 1951.

[13] W. Koehrsen. An Implementation and Explanation of the Random Forest in Python, 2018. Available at https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76.

[14] B. E. Launder. Second-moment closure and its use in modeling turbulent industrial flows. *International Journal for Numerical Methods in Fluids*, 9:963–985, 1989.

[15] F. S. Lien and M. A. Leschziner. Assessment of turbulence-transport models including non-linear rng eddy-viscosity formulation and second-moment closure for flow over a backward-facing step. *Computers & Fluids*, 23(8):983–1004, Nov 1994.

[16] H. Ludwig and W. Tillmann. Untersuchungen uber die wandschubspannung turbulenter reibungsschichten. *Ing.-Arch [Transl. as NACA Tech. Memo 1285, 1949]*, 17, 1949.

[17] T. S. Lund, X. Wu, and K. D. Squires. Generation of Turbulent Inflow Data for Spatially-Developing Boundary Layer Simulations. *Journal of Computational Physics*, 140:233–258, March 1998.

[18] F. R. Menter. Two equation eddy-viscosity models for engineering applications. *AIAA journal*, 32(8):1598–1605, Aug 1994.

[19] P. Moin, K. Squires, W. Cabot, and S. Lee. A dynamic subgrid-scale model for compressible turbulence and scalar transport. *Physics of Fluids A*, 3:2746–2757, 1991.

[20] M. Morkovin. Effects of compressibility on turbulent flows. *In Mecanique de la Turbulence (ed. A. J Favre)*, pages 367–380, 1961.

[21] S. B. Pope. *Turbulent Flows*. Cambridge University Press, Cambridge, 2000.

[22] S. Priebe and P. Martin. Direct numerical simulation of a hypersonic turbulent boundary layer on a large domain. *Journal of Fluid Mechanics*, 2011.

[23] Andrew Ng. Machine learning by Stanford University, 2017. Available at https://www.coursera.org/learn/machine-learning/home/welcome.

[24] ANSYS Inc. *ANSYS® Fluent Theory guide, Release 19.2*. ANSYS Inc., Southpointe 2600 Ansys Drive Canonsburg, PA 15317 USA, 2018.

[25] Mathworks®. *Statistics and Machine Learning Toolbox™, Release 2019Ra*. Mathworks, Southpointe 2600 Ansys Drive Canonsburg, PA 15317 USA, 2019.

[26] J. Rocca. A simple introduction to machine learning - towards data science introductory post about ML, 2019. Available at https://towardsdatascience.com/introduction-to-machine-learning-f41aabc55264.

[27] C. J. Roy and F. G. Blottner. Review and assessment of turbulence models for hypersonic flows. *Progress in Aerospace Sciences*, 42, 2006.

[28] S. Sarkar and B. Lakshmanan. Application of a Reynolds stress turbulence model to the compressible shear layer. *AIAA Journal*, 29(5):743–749, June 1991.

[29] M. Sharif and G. Guo. Computational analysis of supersonic turbulent boundary lyaers over rough surfaces using the k-ω and the stress-ω models. *Applied Mathematical Modelling*, 31:2655–2667, Dec 2007.

[30] K. Sinha and G. V. Candler. Turbulent Dissipation-Rate Equation for Compressible Flows. *AIAA Journal*, 41:1017–1021, June 2003.

[31] P. R. Spalart and A. Leonard. Direct numerical simulation of equilibrium turbulent boundary layers. *In Proc. 5th Symp. on Turbulent Shear Flows*, 1985.

[32] J. Starmer. Regression trees, clearly explained, 2019. Available at https://statquest.org/.

[33] S. Stolz and N. A. Adams. Large-eddy simulations of high reynolds number supersonic boundary layers using the approximate deconvolution model and a rescaling and recycling technique. *Physics of Fluids*, 15, 2003.

[34] G. Urbin and D. Knight. Large-eddy simulation of a supersonic boundary layer using an unstructured grid. *American Institute of Aeronautics and Astronautics*, 39(1288), 2001.

[35] A. Walz. *Boundary Layers Of Flow And Temperature*, volume 1. The M.I.T Press, 1969.

[36] J.-X. Wang, J.-L. Wu, and H. Xiao. A physics informed machine learning approach for reconstructing reynolds stress modeling discrepancies based on dns data. *Physics Review Fluids*, 2017.

[37] D. Wilcox. Progress in hypersonic turbulence modeling. *In Proceeding of the 22nd Fluid Dynamics, Plasma Dynamics and Lasers Conference*, page 1785, June 1991.

[38] D. Wilcox. *Turbulence Modeling for CFD*. DCW Industries, Inc., La Canada, California, USA, 1998.

[39] O. J. H. Williams, D. Sahoo, M. L. Baumgartner, and A. J. Smits. Experiments on the structure and scaling of hypersonic turbulent boundary layers. *Journal of Fluid Mechanics*, 834:237–270, 2018.

[40] J. Wu. *Predictive Turbulence Modeling with Bayesian Inference and Physics-Informed Machine Learning*. PhD thesis, Virginia Tech, Sep 2018.

[41] J.-L. Wu, J.-X. Wang, and H. Xiao. A Bayesian Calibration–Prediction Method for Reducing Model-Form Uncertainties with Application in RANS Simulations. *Flow, Turbulence and Combustion*, 97(3):761–786, Oct 2016.

[42] S. Xu and M. P. Martin. Assessment of inflow boundary conditions for compressible turbulent boundary layers. *Physics of Fluids*, 16(7):2623–2639, June 2004.

[43] T. Yiu. Understanding random forest - how the algorithm works and why it is so effective, 2019. Available at https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

[44] Z. J. Zhang and K. Duraisamy. Machine Learning Methods for Data-Driven Turbulence Modeling. In *22nd AIAA Computational Fluid Dynamics Conference*, Dallas, TX, June 2015. American Institute of Aeronautics and Astronautics.

APPENDIX A: DISCRETIZATION APPROACHES

A.1 Spatial Discretization

This section describes the approach taken to discretize the RANS equations for a steady-state hypersonic flow over a 2D flat plate.

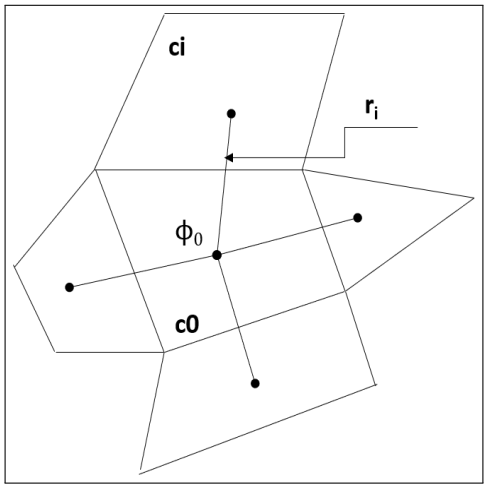


Fig. A.1. Gradient evaluation from cell centers for some scalar ϕ

The CFD solver stores discrete values of the scalars ϕ (eg., Temperature) at the cell centers. A Second-Order Upwind (SOU) Scheme is used for computing the scalars ϕ at cell faces using a multi-dimensional linear reconstruction approach. In this approach, a Taylor series expansion of the cell-centered solution about the centroid is performed by the solver. It is computed by the following expression:

$$\phi_{face,SOU} = \phi + \nabla\phi \cdot \vec{r}, \tag{A.1}$$

where ϕ and $\nabla\phi$ are the cell-centered value and its gradient in the upstream cell, and $\vec{r}$ is the displacement vector from the upstream cell centroid to the face centroid. A

brief summary of the calculation of gradient is discussed in the next section. The diffusion terms are central-differenced and are always second order accurate.

A.2 Least Squares Cell-Based Gradient Evaluation

The calculation of gradient is needed for the construction of scalar at the cell faces, and for computing the secondary diffusion terms and velocity derivatives. The change in values between the cell c_0 and c_i along the vector Δr_i from the centroid of these cells can be expressed as

$$(\nabla\phi)_{c_0} \cdot \Delta r_i = (\phi_{c_i} - \phi_{c_0}). \tag{A.2}$$

Extending Eq. (A.2) to all surrounding cells we obtain a matrix form

$$[J](\nabla\phi)_{c_0} = \Delta\phi, \tag{A.3}$$

where $[J]$ is the coefficient matrix which is purely a function of geometry. In order to determine the cell gradient $\nabla\phi_{c0} = \phi_x\hat{i} + \phi_y\hat{j} + \phi_z\hat{k}$, a least squared approach solution is determined resembling a minimization problem. The coefficient matrix $[J]$ is solved using a Gram-Schmidt process to calculate 3 weights for the three components $W^x_{i_0}, W^y_{i_0}, W^z_{i_0}$ for each individual cell c_0. Therefore, the gradient at the cell center can then be computed as follows:

$$(\phi_x)_{c_0} = \sum_{i=1}^{nx} W^x_{i_0} \cdot (\phi_{c_i} - \phi_{c_0}), \tag{A.4}$$

$$(\phi_y)_{c_0} = \sum_{i=1}^{ny} W^y_{i_0} \cdot (\phi_{c_i} - \phi_{c_0}), \tag{A.5}$$

$$(\phi_z)_{c_0} = \sum_{i=1}^{nz} W^z_{i_0} \cdot (\phi_{c_i} - \phi_{c_0}). \tag{A.6}$$

APPENDIX B: MODELING OF UNKNOWN TERMS USED IN RANS TURBULENCE MODELS

B.1 Reynolds Stress Model

B.1.1 Modeling Turbulent Diffusive Transport

The traditional gradient-diffusion model is modeled as:

$$D_{T,ij} = C_s \frac{\partial}{\partial x_k} \left(\rho \frac{k \overline{u_k' u_l'}}{\varepsilon} \frac{\partial \overline{u_i' u_j'}}{\partial x_l} \right). \tag{B.1}$$

A simplification to the gradient-diffusion model done by Fluent [24] gives:

$$D_{T,ij} = \frac{\partial}{\partial x_k} \left(\frac{\mu_t}{\sigma_k} \frac{\partial \overline{u_i' u_j'}}{\partial x_k} \right) \tag{B.2}$$

and

$$\mu_t = \rho C_\mu \frac{k^2}{\varepsilon}, \tag{B.3}$$

where $C_\mu = 0.09$. A value for $\sigma_k = 0.82$ was derived by Lien [15] by applying the gradient-diffusion model to planar homogeneous shear flow.

B.1.2 Modeling the Pressure-Strain term

A Linear Pressure Strain Model is used to model the Pressure-Strain term in the transport equation. It is modeled using the classical approach given by Gibson [7] and Launder [14]: The pressure strain term is decomposed as

$$\phi_{ij} = \phi_{ij,1} + \phi_{ij,2} + \phi_{ij,w}, \tag{B.4}$$

where $\phi_{ij,1}$ is the slow pressure-strain term, also known as return-to-isotropy term, $\phi_{ij,2}$ is called the rapid pressure strain term, and $\phi_{ij,w}$ is the wall-reflection term.

The slow-pressure strain term $\phi_{ij,1}$, is modeled as

$$\phi_{ij,1} \equiv -C_1 \rho \frac{\varepsilon}{k} \left[\overline{u_i' u_j'} - \frac{2}{3} \delta_{ij} k \right], \tag{B.5}$$

with $C_1 = 1.8$.

$$\phi_{ij,2} \equiv -C_2[(P_{ij} + F_{ij} + 5/6 G_{ij} - C_{ij}) - \frac{2}{3}\delta_{ij}(P + 5/6 G - C)], \tag{B.6}$$

where $C_2 = 0.60$,P_{ij}, F_{ij}, G_{ij} and C_{ij} are defined as per Eq. (2.9), P = $\frac{1}{2}P_{kk}$,G = $\frac{1}{2}G_{kk}$ and C = $\frac{1}{2}C_{kk}$.

The wall-reflection term, $\phi_{ij,w}$ is one of the most important terms to model correctly. It is responsible for the redistribution of normal stresses near the wall. Its works on dampening the normal stress perpendicular to the wall, while enhancing the stresses parallel to the wall. This term is modeled as

$$\begin{aligned} \phi_{ij,w} \equiv C_1' \frac{\varepsilon}{k} \left(\overline{u_k' u_m'} n_k n_m \delta_{ij} - \frac{3}{2} \overline{u_i' u_k'} n_j n_k - \frac{3}{2} \overline{u_j' u_k'} n_i n_k \right) \frac{C_l k^{3/2}}{\varepsilon d}, \\ + C_2' \left(\phi_{km,2} n_k n_m \delta_{ij} - \frac{3}{2} \phi_{ik,2} n_j n_k - \frac{3}{2} \phi_{jk,2} n_i n_k \right) \frac{C_l k^{3/2}}{\varepsilon d}, \end{aligned} \tag{B.7}$$

where $C_1' = 0.5$, $C_2' = 0.3$, n_k is the x_k component of the unit normal to the wall, d is the normal distance to the wall, and $C_l = C_\mu^{3/4}/\kappa$, where $C_\mu = 0.09$ (model-constant) and κ is the von Kármán constant(=0.4187).

B.1.3 Modeling the Turbulence Kinetic Energy

The Turbulence Kinetic Energy(TKE) is obtained by taking the trace of the Reynolds stress tensor:

$$k = \frac{1}{2} \overline{u_i' u_i'}. \tag{B.8}$$

The values of TKE at all locations in the computational domain is calculated by using the Eq. (B.8). The near-wall values are computed by employing the appropriate wall-functions instead of solving a transport equation. A detailed description of the wall-functions used in the RANS code is discussed in section 2.2.4.

B.1.4 Modeling the Turbulence Dissipation Rate

The dissipation tensor, ε_{ij} is modeled as

$$\varepsilon_{ij} = \frac{2}{3}\delta_{ij}(\rho\varepsilon + Y_M), \tag{B.9}$$

where $Y_M = 2\rho\varepsilon M_t^2$ is a "dilatation dissipation" adapted from the model by Sarkar [28]. The turbulent mach number in this term is defined as

$$M_t = \sqrt{\frac{k}{a^2}}, \qquad a = \sqrt{\gamma RT}. \tag{B.10}$$

Since we are solving a compressible flow, this dilatation dissipation is very important in the calculation of dissipation tensor. The scalar dissipation rate, ε is solved as a standard transport equation given by:

$$\frac{\partial}{\partial t}(\rho\varepsilon) + \frac{\partial}{\partial x_i}(\rho\varepsilon u_i) = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\varepsilon}\right)\frac{\partial\varepsilon}{\partial x_j}\right] + C_{\varepsilon 1}\frac{1}{2}[P_{ii} + C_{\varepsilon 3}G_{ii}]\frac{\varepsilon}{k} - C_{\varepsilon 2}\rho\frac{\varepsilon^2}{k} + S_\varepsilon, \tag{B.11}$$

where $\sigma_\varepsilon = 1.0, C_{\varepsilon 1} = 1.44, C_{\varepsilon 2} = 1.92$ and $C_{\varepsilon 3}$ is evaluated as function of local flow relative to the gravitational vector and S_ε is a user-defined source term.

B.2 $k - \omega$ model

B.2.1 Modeling of Effective Diffusivity terms

$$\Gamma_k = \mu + \frac{\mu_t}{\sigma_k}, \qquad \Gamma_\omega = \mu + \frac{\mu_t}{\sigma_\omega}, \tag{B.12}$$

where σ_k and σ_ω are the turbulent Prandtl numbers of k and ω. The turbulent viscosity term is modeled as

$$\mu_t = \alpha^* \frac{\rho k}{\omega}, \tag{B.13}$$

where α^* in high-reynolds number form is 1.0.

B.2.2 Modeling of Production terms

The generation term from the exact equation for the transport of k is given by

$$G_k = \rho \overline{u_i' u_j'} \frac{\partial u_j}{\partial x_i}. \tag{B.14}$$

With Boussinesq hypothesis,

$$G_k = \mu_t S^2 \tag{B.15}$$

where S is the modulus of the mean rate of strain tensor, defined as $S \equiv \sqrt{2 S_{ij} S_{ij}}$ and μ_t is defined as per Eq. (B.3).

The production of ω is given by

$$G_\omega = \alpha \frac{\omega}{k} G_k, \tag{B.16}$$

where G_k is given by Eq. (B.14). The coefficient α is given by

$$\alpha = \frac{\alpha_\infty}{\alpha^*} \left(\frac{\alpha_0 + Re_t/R_w}{1 + Re_t/R_w} \right), \qquad Re_t = \frac{\rho k}{\mu k}, \tag{B.17}$$

where $R_\omega = 2.95$.

B.2.3 Modeling of Dissipation terms

The dissipation of k is given by

$$Y_k = \rho \beta^* f_{\beta^*} k \omega, \tag{B.18}$$

where

$$f_{\beta^*} = \begin{cases} 1, & \chi_k \leq 0 \\ \frac{1+680\chi_k^2}{1+400\chi_k^2}, & \chi_k > 0 \end{cases} \tag{B.19}$$

where

$$\chi_k = \frac{1}{\omega^3}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, \tag{B.20}$$

and

$$\beta^* = \beta_i^*[1 + 1.5 \cdot F(M_t)], \qquad \beta_i^* = 0.09\left(\frac{4/15 + (Re_t/8)^4}{1 + (Re_t/8)^4},\right) \tag{B.21}$$

Re_t is given by Eq. (B.17).

The compressibility correction(F_{M_t}) is important in the determination of the dissipation terms (Wilcox [37])

$$F(M_t) = \begin{cases} 0, & M_t \leq 0.25 \\ M_t^2 - M_{t0}^2, & M_t > 0.25 \end{cases} \tag{B.22}$$

where

$$M_t = \frac{2k}{a^2}, \qquad a = \sqrt{\gamma RT}. \tag{B.23}$$

The dissipation of ω is given by,

$$Y_\omega = \rho\beta' f_\beta \omega^2 \quad \text{and} \quad f_\beta = \frac{1 + 70\chi_\omega}{1 + 80\chi_\omega}. \tag{B.24}$$

$$\chi_\omega = \left|\frac{\Omega_{ij}\Omega_{jk}S_{ki}}{(0.09\omega)^3}\right|, \quad \Omega_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right) \quad \text{and} \quad S_{ij} = \frac{1}{2}\left(\frac{\partial u_j}{\partial x_i} - \frac{\partial u_i}{\partial x_j}\right). \tag{B.25}$$

Also in the context of compressible flows,

$$\beta' = 0.09\left[1 - \frac{\beta_i^*}{\beta_i}1.5 \cdot F(M_t)\right], \tag{B.26}$$

β_i^* and $F(M_t)$ are defined by Eq. (B.21) and Eq. (B.22).

APPENDIX C: WALL TREATMENT

C.1 Enhanced Wall functions

Enhanced Wall functions are used in conjunction with the turbulence model to calculate the flow variables in the viscosity-affected (near-wall) and the outer turbulent region. Enhanced Wall Treatment for the ε-equation constitutes a near-wall modeling approach which uses a two layer model. This approach is useful in an integral part used to specify both ε and μ_t in the wall boundary cells. The two layers are separated using a wall-distance based parameter, turbulent Reynolds number, Re_y define as

$$Re_y \equiv \frac{\rho y \sqrt{k}}{\mu}, \qquad y \equiv \min_{\vec{r}_w \in \Gamma_w} \|\vec{r} - \vec{r}_w\|, \tag{C.1}$$

where y is the wall-normal distance calculated using cell-centers, $\vec{r}$ & $\vec{r}_w$ correspond to the position vector at the near field point and the position of wall boundary respectively and Γ_w is the union of all the wall boundaries involved in the computational domain.

$$\mu_{t,2layer} = \rho C_\mu l_\mu \sqrt{k}, \qquad l_\mu = y C_l^* (1 - e^{-Re_y/A_\mu}). \tag{C.2}$$

Here l_μ is the length scale used for modeling turbulent viscosity. The two layer blending approach given by Jongen [10] suggests the effective turbulent viscosity given by:

$$\mu_{t,enh} = \lambda_\varepsilon \mu_t + (1 - \lambda\varepsilon)\mu_{t,2layer}, \tag{C.3}$$

where μ_t is defined as per Eq. (B.3). The blending is done in a way that λ_ε is unity away from wall boundaries and zero near the walls to combine the viscous region and the outer region. Also,

$$\lambda_\varepsilon = \frac{1}{2}\left[1 + tanh\left(\frac{Re_y - Re_y^*}{A}\right)\right], \qquad A = \frac{|\Delta Re_y|}{arctanh(0.98)}, \tag{C.4}$$

where $Re_y^* = 200$ and ΔRe_y is usually 5-20% of Re_y^*. The ε field in the viscosity affected region is calculated as:

$$\varepsilon = \frac{k^{3/2}}{l_\varepsilon}, \qquad l_\varepsilon = yC_l^*(1 - e^{-Re_y/A_\varepsilon}). \tag{C.5}$$

If the flow domain is completely inside the viscosity affected region, ε is solved algebraically using Eq. (C.5) instead of solving the transport equation. The blending of the ε is similar to the model used in Eq. (C.3). The constants used in Eq. (C.2) and Eq. (C.5) are set as

$$C_l^* = \kappa C_\mu^{-3/4}, A_\mu = 70, A_\varepsilon = 2C_l^*. \tag{C.6}$$

C.1.1 Modeling momentum and heat transfer

Fluent uses the law of the wall formulation for the mean velocity as

$$U^* = \frac{1}{\kappa} ln(Ey^*), \tag{C.7}$$

where

$$U^* \equiv \frac{U_P C_\mu^{1/4} k_P^{1/2}}{\tau_w/\rho}, \qquad y^* \equiv \frac{\rho C_\mu^{1/4} k_P^{1/2} y_P}{\mu}. \tag{C.8}$$

Here,

U^* = dimensionless velocity

y^* = dimensionless distance from the wall

κ = von Kármán constant(= 0.4187)

E = Empirical constant(= 9.793)

U_P = mean velocity of the fluid at the near-wall node P

k_P = turbulence kinetic energy at the near-wall node P

y_P = distance from point P to the wall

μ = dynamic viscosity of the fluid

The blending formulation for the velocity variable extending its applicability throughout the near-wall region is given by Kader [11] is given by:

$$\frac{du^+}{dy^+} = e^{\Gamma}\frac{du^+_{lam}}{dy^+} + e^{1/\Gamma}\frac{du^+_{turb}}{dy^+}, \tag{C.9}$$

$$\Gamma = \frac{a(y^+)^4}{1+by^+}, \tag{C.10}$$

where $a = 0.01$ and $b = 5$.

The model for the enhanced turbulent law of the wall for compressible flow is given by [24]

$$\frac{du^+_{turb}}{dy^+} = \frac{1}{\kappa y^+}\left[S'(1-\beta u^+ - \gamma(u^+)^2)\right]^{1/2}, \tag{C.11}$$

where

$$S' = \begin{cases} 1+\alpha y^+ & for y^+ < y^+_s \\ 1+\alpha y^+_s & for y^+ \leq y^+_s \end{cases} \tag{C.12}$$

and

$$\alpha \equiv \frac{\nu_w}{\tau_w u^*}\frac{dp}{dx} = \frac{\mu dp}{\rho^2(u^*)^3 dx}, \qquad \beta \equiv \frac{\sigma_t q_w}{\rho C p u^* T_w}, \qquad \gamma \equiv \frac{\sigma_t (u^*)^2}{2CpT_w}. \tag{C.13}$$

In Eq. (C.12) $y^+_s = 60$ where it represents the location at which log-law slope is fixed. The coefficients α, β and γ in Eq. (C.13) correspond to pressure gradient and thermal effects respectively in a compressible flow problem.

The laminar law of the wall is determined by:

$$\frac{du^+_{lam}}{dy^+} = 1 + \alpha y^+. \tag{C.14}$$

The determination of heat transfer for a compressible flow using EWF follows a similar approach to the momentum methodology given by Kader [11]

$$T^+ \equiv \frac{(T_w - T_P)\rho C p u_P}{\dot{q}} = e^{\Gamma}T^+_{lam} + e^{1/\Gamma}T^+_{turb}. \tag{C.15}$$

The notations are similar to the ones described in Eq. (C.8) and the blending function is given by

$$\Gamma = -\frac{a(Pry^+)^4}{1 + bPr^3y^+}, \tag{C.16}$$

where Pr is the molecular Prandtl number, and the coefficients a and b is same as the blending function for the model used for momentum.

$$T^+_{lam} = Pr\left(u^+_{lam} + \frac{\rho u^*}{2\dot{q}}u^2\right), \tag{C.17}$$

$$T^+_{turb} = Pr_t\left\{u^+_{turb} + P + \frac{\rho u^*}{2\dot{q}}\left[u^2 - \left(\frac{Pr}{Pr_t} - 1\right)(u^+_c)^2(u^*)^2\right]\right\}, \tag{C.18}$$

where u^+_c is the crossover between the laminar and turbulent region. The function P is given by

$$P = 9.24\left[\left(\frac{Pr}{Pr_t}\right)^{3/4} - 1\right][1 + 0.28e^{-0.007Pr/Pr_t}]. \tag{C.19}$$

APPENDIX D: MODELING OF LAMINAR VISCOSITY IN HYPERSONIC CALCULATIONS

Sutherland's law for viscosity for air as a working fluid is given by:

$$\mu_{lam} = \mu_0 \left(\frac{T}{T_0}\right)^{3/2} \frac{T_0 + S}{T + S}, \tag{D.1}$$

where T is the static temperature of the flow, $\mu_0 = 1.716$ x 10^{-5} kg/m-s, $T_0 = 273.11K$, and $S = 110.56K$.

A model for a more robust calculation of viscosity as given by Keyes [12] is used in the code.

$$\mu_{lam} = 1.488 \text{ x } 10^{-6} \frac{\sqrt{T}}{1 + \left(\frac{122.1}{T}\right) 10^{-5/T}}, \tag{D.2}$$

The two viscosity laws agree for temperatures $T > 150K$, for which the relative error is $< 1\%$. At lower temperatures, the viscosity laws diverge. Figure D.1 shows that Keyes law is more accurate and its use is recommended over Sutherland's Law at extremely low temperatures as suggested by Roy [27].

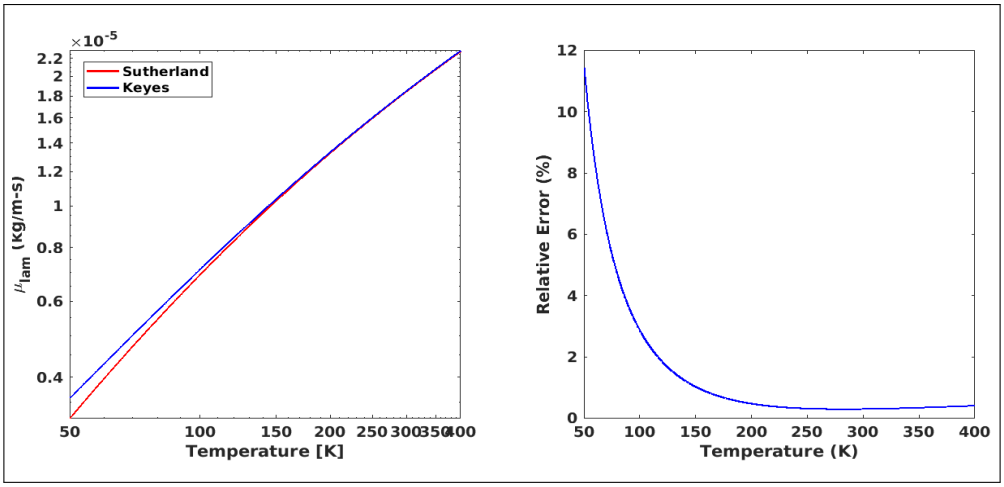


Fig. D.1. Comparison of Laminar viscosity models for hypersonic flows

APPENDIX E: RESCALING EQUATIONS

Throughout the discussion of Rescaling-Recycling (RR) methods in this section, here are some important information. The equations described are for any general 3-D flow simulation, however it is appropriately used to derive and implement for a planar 2-D hypersonic turbulent boundary layer simulation.

The streamwise, wall-normal and span-wise velocity components are given by $u(= U + u'), v(= V + v')$ and $w(= W + w')$ where the capital letters represent the mean quantities and the lower-case letters with a prime represent the fluctuating components. It is also noted that $(.)_{in}$ represents the inlet station and $(.)_{re}$ represents the recycle station. In order to fully specify suitable boundary conditions at the inlet station of RANS calculations with RSM and $k-\omega$ model, the rescaling equations for the following variables are completely essential:

- Mean flow quantities (U, V, W)
- Turbulent Kinetic Energy (u', v', w', k)
- Turbulent Dissipation (ε)
- Reynolds Stresses $(\overline{u'u'}, \overline{v'v'}, \overline{w'w'}, \overline{u'v'})$
- Mean Temperature (T)
- Mean Pressure and Density (P, ρ)

E.1 Simple RR method

The simple Rescaling-Recycling(RR) method is the first work done after significant modifications to the work of Spalart [31]. Lund [17] worked on developing a methodological way to convert the idea from Spalart to a more easy and programmable way without the need for coordinate transformation of the Navier stokes equations. The

main complications of [31] method is the complications which arise out of the need to introduce a coordinate transformation that minimizes streamwise inhomogeneity. This method circumvents the problem of evaluating the growth terms throughout the solution domain by just transforming the boundary conditions. The calculation of growth terms necessitates multiple simulations to estimate streamwise gradients of mean flow quantities.

Although the method by Spalart [31] gives a ingenious transformation to calculate spatially evolving boundary layers with periodic boundary conditions in streamwise direction, and is highly accurate, it poses the problem of being complicated for the purpose of generating just the inflow data for simulation. This method involves estimation of velocity at the inlet plane as a boundary conditions using the solution downstream. The use of periodic boundary conditions in the streamwise direction is removed and therefore a Fourier transformation is impossible. The existing inflow-outflow simulation code can be made use to generate valuable inflow boundary conditions to create a solution domain which starts from a specified boundary layer thickness or user defined boundary conditions extracted from the downstream solution. The outlet boundary condition remain unchanged however the velocity field in a chosen downstream location (Recycle plane) is extracted, rescaled using the concepts of Spalart [31] and recycled to give as a boundary condition to the inlet.

E.1.1 Scaling of Mean Velocity

The simple RR method consists of rescaling the mean velocity according to two regions, law of the wall (INNER) and defect law (OUTER) region. The law of the wall yields:

$$U^{inner} = u_\tau(x) f_1(y^+), \tag{E.1}$$

where $u_\tau = \sqrt{\nu_w(\partial u/\partial y)_w}$ is the friction velocity, $y^+ = (u_\tau y)/\nu_w$ is the wall coordinate and f_1 is a universal function. Similarly, the defect law formulated by Coles [4]:

$$U_\infty - U^{outer} = u_\tau(x) f_2(\eta), \tag{E.2}$$

where $\eta = y/\delta$ is the outer coordinate (δ is the boundary layer thickness), U_∞ is the free-stream velocity and f_2 is another universal function. The actual formulation of $f_1 \& f_2$ is not essential for the derivation of the rescaling equations for the mean flow quantities.

The rescaling of mean streamwise velocities(U) is given by:

$$U_{in}^{inner} = \omega_{u_\tau} U_{re}(y_{in}^+) \quad \text{and} \quad U_{in}^{outer} = \omega_{u_\tau} U_{re}(\eta_{in}) + (1 - \omega_{u_\tau}) U_\infty, \tag{E.3}$$

where

$$\omega_{u_\tau} = \frac{(u_\tau)_{in}}{(u_\tau)_{re}} \quad \text{and} \quad \omega_{\nu_w} = \frac{(\nu_w)_{in}}{(\nu_w)_{re}}. \tag{E.4}$$

The independent variables y^+ and η are the inner and outer coordinates of the grid nodes at the inlet station. The location at the inlet in the wall-normal direction will not be the same as the inlet and a suitable location of the variables is taken for each individual inlet location given by: If $f_1 \& f_2$ are assumed as a universal functions

$$y_{in} = \frac{\omega_{\nu_w}}{\omega_{u_\tau}} y_{re}, \qquad y_{in} = \frac{\delta_{re}}{\delta_{in}} y_{re}. \tag{E.5}$$

According to Eq. (E.5) the location of extraction of variables are chosen and recycled for the inner and outer regions respectively.

Similarly, for the mean wall-normal velocities are assumed to scale by,

$$V^{inner} = U_\infty f_3(y^+), \qquad V^{outer} = U_\infty f_4(\eta), \tag{E.6}$$

where $f_3 \& f_4$ are assumed as universal functions. The rescaling is given by:

$$V_{in}^{inner} = V_{re}(y_{in}^{+}), \qquad V_{in}^{outer} = V_{re}(\eta_{in}). \tag{E.7}$$

The span wise velocity should be zero in the mean quantities and hence recycling is not required.

E.1.2 Scaling of Velocity Fluctuations

The velocity fluctuations in the inner and outer regions are assumed to have this form (Lund [17]):

$$(u')_i^{inner} = u_\tau g_i(x, y^+, z, t) \quad \text{and} \quad (u')_i^{outer} = u_\tau h_i(x, \eta, z, t). \tag{E.8}$$

Since there is an explicit dependence on u_τ it is assumed that g_i and h_i are approximately homogeneous in the streamwise direction. There is a one-way coupling between recycle station and inlet through Eq. (E.8) but no downstream transfer of information from inlet. The recycling equations for all the fluctuating variables $u_i'(= u', v', w')$ are given by:

$$(u_i')_{in}^{inner} = \omega_{u_\tau}(u_i')_{re}(y_{in}^{+}, z, t) \tag{E.9}$$

and

$$(u_i')_{in}^{outer} = \omega_{u_\tau}(u_i')_{re}(\eta_{in}, z, t).. \tag{E.10}$$

Using Eq .(E.3),(E.7),(E.9) and Eq .(E.10) a composite velocity profile can be constructed using suitable weight functions to blend the inner and the outer regions.

$$(u_i)_{in} = [(U_i)_{in}^{inner} + (u_i')_{in}^{inner}][1 - W(\eta_{in})] + [(U_i)_{in}^{outer} + (u_i')_{in}^{outer}]W(\eta_{in}). \tag{E.11}$$

The weighting function $W(\eta)$ is defined as (see Lund [17])

$$W(\eta) = \frac{1}{2}\left\{1 + \frac{\tanh\left[\frac{a(\eta-b)}{(1-2b)\eta+b}\right]}{\tanh(\mathrm{a})}\right\}, \tag{E.12}$$

where $a = 4$ and $b = 0.2$. The weighting function is such that its unity at $\eta = 1$ and zero at $\eta = 0$. Also, since the RANS simulations cater to only the mean quantities in the overall boundary conditions, the RANS form requires just the mean version of the composite velocity profiles from Eq. (E.11) given by:

$$(U_i)_{in} = (U_i)_{in}^{inner}[1 - W(\eta_{in})] + (U_i)_{in}^{outer} W(\eta_{in}). \tag{E.13}$$

An important step in the rescaling operation requires the value of scaling parameters u_τ and δ at both the inlet and recycle stations. These quantities can be determined during calculation from the mean velocity profile. Lund [17] and Xu [42] say that the problem becomes over-constrained if we determine the values of the scaling parameters both from the solution of the mean velocity profile. Hence the usage of an empirical relation for the value of u_τ is recommended since, defining a boundary layer thickness at the inlet according to the problem is more beneficial than specifying the friction velocity at the inlet. The friction velocity follows an empirical relation given by Ludwig [16]

$$(u_\tau)_{in} = (u_\tau)_{re}\left(\frac{\theta_{re}}{\theta_{in}}\right)^{[1/2(n-1)]}, \qquad n = 5, \tag{E.14}$$

where θ is the momentum thickness. This relation is a direct correlation derived from standard power law approximations of $C_f \sim Re_x^{-1/n}, \theta/x \sim Re_x^{-1/n}$.

E.1.3 Scaling of Turbulence quantities

The Turbulent kinetic energy is calculated from its standard definition given by:

$$k = \frac{1}{2}\overline{u_i' u_i'} = \frac{1}{2}(\overline{u'^2} + \overline{v'^2} + \overline{w'^2}). \tag{E.15}$$

The rescaling equation is of the form

$$k_{in}^{inner} = \omega_{u_\tau}^2 k_{re}(y_{in}^+, z, t), \quad \text{and} \quad k_{in}^{outer} = \omega_{u_\tau}^2 k_{re}(\eta_{in}, z, t). \tag{E.16}$$

The Turbulent Dissipation for a homogeneous compressible flow can be split into a solenoidal part and a dilatational part. The magnitude of dilatational dissipation is much smaller than the solenoidal part for wall bounded compressible flows (see Sinha [30]). In an incompressible flow the turbulent dissipation is given by

$$\varepsilon_I = 2\nu\overline{S'_{ij}S'_{ij}} \simeq 2\nu\overline{u'_{i,j}u'_{i,j}}. \tag{E.17}$$

However for a compressible flow the solenoidal dissipation can be derived from the enstrophy equation as:

$$\varepsilon_S = \overline{\nu}(\overline{u'_{i,j}u'_{i,j}} - \overline{u'_{i,j}u'_{j,i}}) = \overline{\nu}\overline{\omega'_i\omega'_i}. \tag{E.18}$$

The paper by Sinha [30] gives a convincing argument to prove that the magnitude of the second part of ε_S is insignificant and for most general compressible flows the dissipation is equivalent to the incompressible version.

$$\begin{aligned} \overline{\omega'_i\omega'_i} &= \overline{u'_{i,j}u'_{i,j}} - \overline{u'_{i,j}u'_{j,i}} \\ &= \overline{u'_{i,j}u'_{i,j}} - \frac{\partial}{\partial x_j}\overline{u'_iu'_{j,i}} - \frac{\partial}{\partial x_j}\overline{u'_i\theta'} + \overline{\theta'^2}. \end{aligned} \tag{E.19}$$

where θ' is the fluctuating dilatation. It is seen in the paper that after comparing with the budget terms from the transport equation of the enstrophy equation,

$$\overline{u'_{j,i}u'_{i,j}} \ll \overline{u'_{i,j}u'_{i,j}}.$$

Thus ε_S can be split to a part identical to ε_I and a second term which represents the effect of inhomogeneity and compressibility. So

$$\varepsilon_S \simeq \overline{\nu u'_{i,j} u'_{i,j}}. \tag{E.20}$$

Thus the equation for ε is given by

$$\varepsilon = \overline{\nu \frac{\partial u_i}{\partial x_j} \frac{\partial u_i}{\partial x_j}}. \tag{E.21}$$

The recycling equation is given by:

$$\varepsilon_{in}^{inner} = \omega_{u_\tau}^2 \varepsilon_{re}(y_{in}^+, z, t), \quad \text{and} \quad \varepsilon_{in}^{outer} = \omega_{u_\tau}^2 \varepsilon_{re}(\eta_{in}, z, t). \tag{E.22}$$

The same blending function given by Eq. (E.12) is used to produce a single composite value of k and ε. The reynolds stress quantities are computed using the rescaling equations Eq. (E.9) and Eq. (E.10) to construct a rescaling factor for the stress variables according to the need of the boundary conditions in RANS.

E.1.4 Scaling for Mean Temperature, Pressure and Density

There is a stark correlation between the velocity and temperature for turbulent boundary layer flows with constant wall temperature, given by a temperature-velocity coupling relation (see Walz [35]). For a zero-pressure gradient boundary layer, Walz's equation is given by:

$$\frac{T}{T_\infty} = \frac{T_w}{T_\infty} + \frac{T_r - T_w}{T_\infty} \left(\frac{U}{U_\infty} \right) - R \frac{\gamma - 1}{2} M_\infty^2 \left(\frac{U}{U_\infty} \right)^2, \tag{E.23}$$

where T_r is the recovery temperature, γ is the ratio of specific heats and M_∞ is the free stream mach number. The recovery temperature is the temperature corresponding to the total enthalpy contained by a system and is given by:

$$\frac{T_r}{T_\infty} = \left(1 + R\frac{\gamma - 1}{2}M_\infty^2\right), \tag{E.24}$$

where R is the recovery factor of the flow. The rescaling of the temperature is computed from the mean velocity field as per Eq. (E.23)

$$\frac{T_{in}^{inner}}{T_\infty} = \frac{U_{re}(y_{in}^+)}{U_\infty}, \qquad \frac{T_{in}^{outer}}{T_\infty} = \frac{U_{re}(\eta_{in})}{U_\infty}. \tag{E.25}$$

The density of the recycling is performed by a direct interpolation in the temperature field which correspond to the same value of density. Since the thermodynamic variables do not change during the spatial evolution of the boundary layer, the density extracted from the downstream location at the same temperature preserves itself in the inlet domain. The temperature of each wall normal coordinate is calculated by the aforementioned method, and a direct interpolation of that specific temperature is performed on the temperature field in the recycle station to obtain a value of the density at the inlet grid coordinate.

The recycle equation for pressure follows the general ideal gas equation given by:

$$P = \overline{\rho} R_g T, \tag{E.26}$$

where R_g denotes the gas constant for air.

E.2 Modified RR method

Modified RR method is a method inspired from the scaling methodology discussed in the previous section, but include slight improvements to improve the rescaling equations for the individual variables (see Xu [42]). Firstly, the mean streamwise

velocities are split into 3 individual regions than 2 regions in the simple RR method. The mean streamwise velocities are split into 3 regions (VISCOUS, LOG & WAKE), and the rescaling equations for the other variables consist of just two regions (INNER, OUTER). A rigorous treatment for the wall normal velocity is not done because it is not a dynamically dominant quantity. For a flat plate zero-pressure gradient boundary layer (ZPGBL) the mean spanwise velocity is zero dye to the spanwise statistical symmetry.

E.2.1 Scaling of Mean streamwise velocity

The rescaling equations are given for the three different regions separately and then blended together using weights (B_i) to construct a composite velocity profile. In the VISCOUS region, it is assumed that

$$\frac{U^S}{u_\tau} = y^+, \tag{E.27}$$

where u_τ and y^+ follow the usual definition of friction velocity and the wall coordinate. The term U^S refers to the transformed velocity which directly takes into account the effect of viscosity variation inside the boundary layer. The mean streamwise transformed velocity is defined by

$$U^S = \int_0^U \frac{\mu}{\mu_w} dU. \tag{E.28}$$

It is assumed that the for the viscous and the log region, rescaling happens when the wall coordinate indices of the inlet and recycling station are equal whereas for the wake region, the location of extraction correspond to the outer region as the simple RR method where $\eta = y/\delta$ is equal for both the stations. The location of extractions are similar to Eq. (E.5). The rescaling equations for viscous region is given by:

$$(U^S)_{in} = \omega_{u_\tau} (U^S)_{re}. \tag{E.29}$$

In the LOG region or the inertial sublayer, the logarithmic law is relevant for turbulent flows.

$$\frac{U^{**}}{u_\tau} = \frac{1}{\kappa} ln y^+ + C, \tag{E.30}$$

where C is a constant, and U^{**} is the van Driest transformed velocity defined as,

$$U^{**} = \int_0^U \sqrt{\frac{T_w}{T}} dU. \tag{E.31}$$

The logarithmic region follows a self-similar expression

$$\frac{U^{**}}{u_\tau} = f_{log}(y^+), \tag{E.32}$$

where f_{log} is a universal function. So when $(y^+)_{in} = (y^+)_{re}$, the rescaling equations for this region is given by,

$$(U^{**})_{in} = \omega_{u_\tau} (U^{**})_{re}. \tag{E.33}$$

For the outer layer or the WAKE region, a different kind of similarity law is used where,

$$\frac{U_e^{**} - U^{**}}{u_\tau} = f_{wake}(\eta), \qquad \eta = y/\theta. \tag{E.34}$$

Here f_{wake} is yet another universal function, with the normalizing constant for η taken as the momentum thickness (θ). The rescaling equations for this region is given by,

$$(U^*)_{in} = \omega_{u_\tau} (U^*_{re}), \qquad U^* = U_e^{**} - U^{**} = \int_U^{U_e} \sqrt{\frac{T_w}{T}} dU. \tag{E.35}$$

E.2.2 Scaling of wall-normal velocity

An approximation from the mean continuity equation can be made as,

$$V = -\frac{1}{\overline{\rho}} \int_0^y \frac{\partial \overline{\rho} U}{\partial x} dy. \tag{E.36}$$

Hence an appropriate scaling for the wall normal velocity is given by,

$$\frac{V}{u_\tau}\sqrt{\frac{\overline{\rho}}{\overline{\rho_w}}} = f_{inner}(y^+), \quad \text{and} \quad \frac{V}{u_\tau}\sqrt{\frac{\overline{\rho}}{\overline{\rho_w}}} = f_{outer}(\eta). \tag{E.37}$$

Hence the rescaling equations are given by,

$$(V)_{in} = \omega_{u_\tau}\omega_{\rho_w}\sqrt{\frac{\overline{(\rho)}_{re}}{\overline{(\rho)}_{in}}}(V)_{re}, \qquad \omega_{\rho_w} = \sqrt{\frac{\overline{(\rho_w)}_{in}}{\overline{(\rho_w)}_{re}}}. \tag{E.38}$$

E.2.3 Scaling of Turbulence Quantities

The scaling of the turbulence quantities are based on the work of Morkovin [20] and Bradshaw [2] where a mean-density variation is important in the analysis of hypersonic turbulent boundary layers. When the velocity fluctuations are normalized by the inclusion of $\sqrt{(\overline{\rho_w}/\overline{\rho})}$ to the friction velocity, the plots of compressible flows collapse clearly with the incompressible data. The recycling equations are therefore given by,

$$(u_i')_{in} = \omega_{u_\tau}\omega_{\rho_w}\sqrt{\frac{\overline{(\rho)}_{re}}{\overline{(\rho)}_{in}}}(u_i')_{re}. \tag{E.39}$$

The rescaling of the reynolds stresses are based on the scaling factor for velocity fluctuations given by,

$$\zeta = \omega_{u_\tau}\omega_{\rho_w}\sqrt{\frac{\overline{(\rho)}_{re}}{\overline{(\rho)}_{in}}}. \tag{E.40}$$

The rescaling of the P, ρ, T, k and ε are followed on a similar methodology as described in section E.1.4 and section E.1.3.

Using Eq .(E.29), (E.33), (E.35) and Eq .(E.39), a composite velocity profile is constructed using weight functions. The streamwise velocity has 3 distinct layers whereas the other variables have only 2 distinguished regions where the blending is necessary. For example, the mean streamwise velocity is computed as:

$$U_{in} = U^{visc}B_1(y) + U^{log}B_2(y) + U^{wake}B_3(y), \tag{E.41}$$

and the velocity fluctuations are computed as

$$u'_{in} = (u')^{inner}[1 - B_3(y)] + (u')^{outer} B_3(y). \tag{E.42}$$

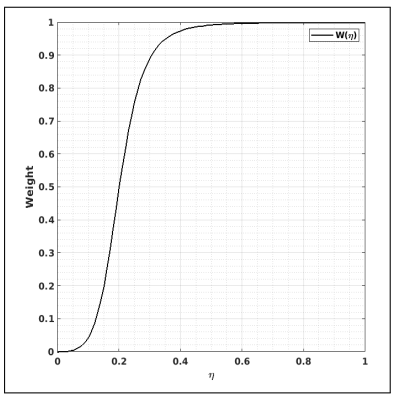
The weight functions are constructed from hyperbolic-tangent functions as

$$B_1(k) = \frac{1}{2}\left\{1 - \tanh\left[c_1 \frac{k - k_{a1}}{k_{logs} - k_{visc}}\right]\right\}, \tag{E.43}$$

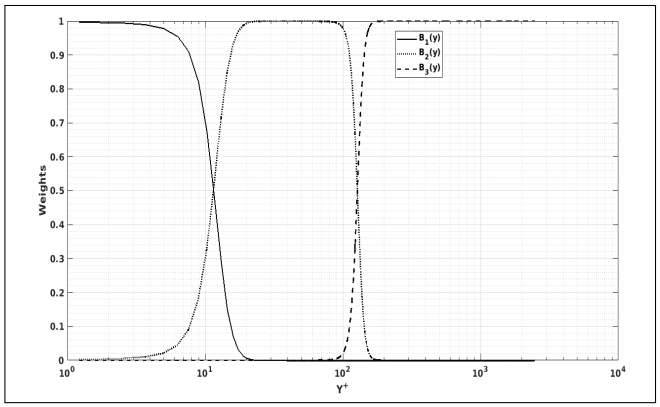
$$B_2(k) = \frac{1}{2}\left\{\tanh\left[c_1 \frac{k - k_{a1}}{k_{logs} - k_{visc}}\right] - \tanh\left[c_2 \frac{k - k_{a2}}{k_{wake} - k_{loge}}\right]\right\}, \tag{E.44}$$

$$B_1(k) = \frac{1}{2}\left\{1 + \tanh\left[c_2 \frac{k - k_{a2}}{k_{wake} - k_{loge}}\right]\right\}, \tag{E.45}$$

Here, k is the wall coordinate, $c_1 = 5.1$ and $c_2 = 6.2$ are user defined constants which specify the steepness of the weight functions. It is to be noted that k_{visc},k_{logs},k_{loge} and k_{wake} are the wall normal indices which denote the different regions inside the boundary layer, $k_{a1} = (k_{visc} + k_{logs})/2$ and $k_{a2} = (k_{loge} + k_{wake})/2$ The values of k_{visc}, k_{logs}, k_{loge} and k_{wake} are chosen to correspond $y^+ = 3$, $y^+ = 25$, $y/\delta = 0.4$ and $y/\delta = 0.8$ respectively. Figures E.1a and E.1b show the variation in the value of weights across the boundary layer in terms of normalized wall normal distances.



(a) Weight function used for simple RR method (Lund [17])



(b) Weight functions used for modified RR method

A comprehensive tabulation of the scaling approaches used in the aforementioned methods of rescaling is given in Table E.1.

Table E.1. Rescaling equations for flow and turbulence quantities

Variable	**Simple RR**	**Modified RR**
U	$U_{in}^{inner} = \omega_{u_\tau} U_{re}(y_{in}^+)$	$(U^S)_{in}^{visc} = \omega_{u_\tau}(U^S)_{re}(y_{in}^+)$
		$(U^{**})_{in}^{log} = \omega_{u_\tau}(U^{**})_{re}(y_{in}^+)$
	$U_{in}^{outer} = \omega_{u_\tau} U_{re}(\eta_{in})$	$(U^*)_{in}^{wake} = \omega_{u_\tau}(U^*)_{re}(\eta_{in})$
V	$V_{in}^{inner} = V_{re}(y_{in}^+)$	$V_{in}^{inner} = \zeta V_{re}(y_{in}^+)$
	$V_{in}^{outer} = V_{re}(\eta_{in})$	$V_{in}^{outer} = \zeta V_{re}(\eta_{in})$
T	$T_{in}^{inner} = f_{walz}(U_{in}^{inner}(y_{in}^+))$	$T_{in}^{visc} = f_{walz}(U_{in}^{visc}(y_{in}^+))$
		$T_{in}^{log} = f_{walz}(U_{in}^{log}(y_{in}^+))$
	$T_{in}^{outer} = f_{walz}(U_{in}^{outer}(\eta_{in}))$	$T_{in}^{wake} = f_{walz}(U_{in}^{wake}(\eta_{in}))$
$\boldsymbol{u', v', w'}$	$(u_i')_{in}^{inner} = \omega_{u_\tau} u_{i,re}'(y_{in}^+)$	$(u_i')_{in}^{inner} = \zeta u_{i,re}'(y_{in}^+)$
	$(u_i')_{in}^{outer} = \omega_{u_\tau} u_{i,re}'(\eta_{in})$	$(u_i')_{in}^{outer} = \zeta u_{i,re}'(\eta_{in})$
k	$k_{in}^{inner} = \omega_{u_\tau}^2 k_{re}(y_{in}^+)$	$k_{in}^{inner} = \zeta^2 k_{re}(y_{in}^+)$
	$k_{in}^{outer} = \omega_{u_\tau}^2 k_{re}(\eta_{in})$	$k_{in}^{outer} = \zeta^2 k_{re}(\eta_{in})$
$\boldsymbol{\varepsilon}$	$\varepsilon_{in}^{inner} = \omega_{u_\tau}^2 \varepsilon_{re}(y_{in}^+)$	$\varepsilon_{in}^{inner} = \zeta^2 \varepsilon_{re}(y_{in}^+)$
	$\varepsilon_{in}^{outer} = \omega_{u_\tau}^2 \varepsilon_{re}(\eta_{in})$	$\varepsilon_{in}^{outer} = \zeta^2 \varepsilon_{re}(\eta_{in})$
$\boldsymbol{\rho}$	$\rho_{in}^{inner} = \rho_{re}(T^{inner}, y_{in}^+)$	$\rho_{in}^{visc} = \rho_{re}(T^{visc}, y_{in}^+)$
		$\rho_{in}^{log} = \rho_{re}(T^{log}, y_{in}^+)$
	$\rho_{in}^{outer} = \rho_{re}(T^{outer}, \eta_{in})$	$\rho_{in}^{wake} = \rho_{re}(T^{wake}, \eta_{in})$
P	$P = \rho R_g T$	$P = \rho R_g T$