The Pennsylvania State University
The Graduate School
College of Engineering

IMPLEMENTATION OF HIGH FST TURBULENT KINETIC ENERGY DIFFUSION MODEL IN OPENFOAM CODE FOR SIMULATION OF HEAT TRANSFER OVER A GAS TURBINE BLADE

A Thesis in
Mechanical Engineering
by
Vedant Chittlangia

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2017

The thesis of Vedant Chittlangia was reviewed and approved* by the following:

Savas Yavuzkurt
Professor of Mechanical Engineering
Thesis Advisor, Chair of Committee

Anil Kulkarni
Professor of Mechanical Engineering
Thesis Reader

Karen Thole
Distinguished Professor of Mechanical Engineering
Head of the Department of Mechanical Engineering

*Signatures are on file in the Graduate School.

Abstract

Several low-Reynolds number turbulence models are compared for their performance to predict the effect of free-stream turbulence on skin friction coefficient and Stanton number. Initial efforts were made to implement a turbulent kinetic energy diffusion model in FLUENT but due to unavailability of the source code the focus was shifted to OpenFOAM, an open-source CFD software. Effect of initial values of free-stream parameters is studied on a flat plate. Launder-Sharma model is modified to account for the effect of free-stream turbulence and length scale by incorporating variable c_μ and an additional diffusion term in turbulent kinetic energy transport equation. Comprehensive study is carried for better predictions of skin friction coefficient, Stanton number and turbulent kinetic energy profile on a flat plate for different initial turbulent intensities of 1%, 6.53% and 25.7%. The new model is validated for its better performance on a flat plate to predict Stanton number, skin friction coefficient and turbulent kinetic energy profile with close match to the data points. Prediction of Stanton number shows an improvement of 20% as compared to baseline Launder-Sharma (LS) model at highest turbulent intensity of 25.7% with corresponding 15 % and 38 % improvement in skin friction coefficient and peak turbulent kinetic energy values respectively. The application of the new model is extended by implementing it on C3X stationary turbine vane cascade at low (0.5%) and high (20%) turbulent intensities. Prediction for Nusselt number shows an improvement as compared to other low Reynolds number models. At lower intensity, Nusselt number is predicted similar to the baseline LS model but the suction side measurements shows an improvement of over 20 %. In the real turbine like scenario of high initial turbulent intensity, Nusselt improved by over 8 % on the pressure side and showed 50 % improvement on suction side along with correct capture of the trend. It seems that the additional diffusion term enables the model to capture the physics of high freestream turbulence effects by improving kinetic energy diffusion from the freestream to the near wall region and variable c_μ prevents higher values of turbulent viscosity near the wall.

Table of Contents

List of Figures

List of Tables

List of Symbols

Nomenclature

C	Vane Chord, m
C_{FST}	Present model constant
C_1, C_2	Constants in two-equation model
Cf	Skin friction coefficient
Pr	Prandtl number
Pe	Peclet number
R	Reynolds stress tensor in OpenFOAM
St	Stanton number
U	Velocity, m/s
c_μ	Constant in two-equation turbulence model and eddy viscosity
f_1, f_2	Low Reynolds Number model functions
f_μ	Damping function in Low Reynolds Number models
k	Turbulent kinetic energy, m^2/s^2
s	Distance from vane's leading edge stagnation point, m
u_*	Friction velocity, m/s
$\bar{u}$	Ensemble average velocity, m/s
$\overline{u'v'}$	Reynolds stress
x	Streamwise distance, m
y	Normal distance to streamwise, m
y^+	Inner variable

Greek letters

ε	Turbulent dissipation rate, m^s/s^3
ω	Turbulent dissipation time scale, s^{-1}

θ	Momentum thickness / Non-dimentional temperature, m
δ	Velocity boundary layer thickness, m
δ_T	Thermal boundary layer thickness, m
μ	Molecular viscosity, Pa-s
ν	Kinematic viscosity,
σ	Schmidt number

Subscripts

i	Initial value or location
∞	Free-stream conditions
x	Based on streamwise distance
θ	Based on momentum thickness
T	Thermal
t	Turbulent
1	First grid location

List of Abbreviations

AKN	Abe-Kondoh-Nagano model
CFD	Computational Fluid Dynamics
CHC	Chang-Hsieh-Chen
FST	Free-stream turbulence
GUI	Graphic User Interface
LB	Lam-Bremhorst model
LRN	Low Reynolds Number
LS	Launde-Sharma model
RANS	Reynolds Average Navier-Stokes
SIMPLE	Semi-Implicit Method for Pressure Linked Equations
SST	Shear Stress Transport
TDR	Turbulent Dissipation Rate
TKE	Turbulent Kinetic energy
Tu	Turbulence intensity
UDF	User Defined Function
YS	Yang-Shih model

Acknowledgments

Firstly, I would thank my parents, grandparents and kaka-kaki who always believed in me and motivated me to complete this work. I am thankful to my siblings Sona, Mona and Vaibhav who always look up to me for inspiration and this helps me to be strong in tough times. I am extremely grateful my adviser Prof. Savas Yavuzkurt and thank him for guiding me through the completion of this work and his valuable insights in the project. I would thank my colleagues Hosein, Peter, Feiyan and especially Seung who were helpful for the use the software and sit with me to help me debug my code. I am thankful to ICS Penn State to provide high speed computing on Linux clusters and their help in debugging several job submission queries. I appreciate the help of IT staff of Aerospace department, Ben and Kirk, for their help in data recovery. I also thank graduate office staff in Mechanical Engineering, Jason and Julie, for their timely reminders to help me keep up with the submission deadlines and their help in the paperwork. I am thankful to the Argentine Tango Community at State College for making me feel at home away from home. I am grateful to my roommates, Mayank and Lalit, and my friends Swapnil, Karthik, Santosh and others who were my support here throughout all ups and downs.

Chapter 1
INTRODUCTION

1.1 Introduction

Turbine inlet temperature has a direct influence on the turbine efficiency in modern gas turbine systems. Modern turbine blades operate near their thermal limits with the inlet temperature higher than 1700^oC [4] being limited by the turbine blade material and the effectiveness of the turbine blade cooling system. Thus it is critical to have accurate predictions of hot mainstream to blade surface heat transfer. The inlet to the turbine has a flow with high free-stream turbulence (FST) coming from the combustor which affects the near wall characteristics of the flow near the turbine blades and thus has significant importance in study of heat transfer and drag forces on the blades [5].

Free-stream turbulence (FST) is the turbulence in the approaching stream. The effect of free stream turbulence on the turbulent boundary layer is determined by the turbulent intensity of the free-stream:

$$Tu_\infty = \left(\frac{\bar{u^2} + \bar{v^2} + \bar{w^2}}{U}\right)_\infty \approx \frac{\sqrt{\bar{u^2}}}{U_\infty} \tag{1.1}$$

Turbulent boundary layers with FST include nozzle guide vanes, gas turbine engines, electronic cooling passages, and many industrial system that such as turbo-machines, turbo-compressors, and combustion chambers [6].

FST occur naturally as the property of the flow. Experimental wind tunnels uses very low incoming turbulent intensity (Tu). Turbulence intensity is increased by grids of varying shapes and sizes at upstream of the test section or measurement area. The generated intensity varies between 1% and 12%, with latter being the practical limitation of grid generated turbulent intensity. To generate higher intensities, high speed jet based systems are used to stimulate the turbine like behavior
citekohli1998effects.

The high FST intensity and large length scales are one of the major factors impacting the heat transfer. Even if the intensity decreases at downstream locations but the length scale are still large to impact the heat transfer rates. FST affects the boundary layer development and the convective heat transfer from the surfaces. FST also increases surface shear stress and thus the losses. Thus it is extremely important to identify this behavior and study the effects of high FST for correct designs [7].

Low cost investigation and availability of detailed set of data make computational approach in analyzing such complex problem desirable over the experiments. Moreover, experimental investigations are limited to a finite distance near the jet. So, to completely analyze the behavior of the flow field it is important to model the problem using advanced computational tools. Though it is a matter of concern to select the turbulence model to correctly predict the behavior of the flow under varying free stream turbulence.

The aim is to provide realistic gas side predictions of heat transfer for a turbine airfoil. The flow in turbine passage is three dimensional however at this point a two-dimensional study is adequate to predict heat transfer since in the mid-span other complex flow features such as turbulent intensity, pressure gradient, surface curvature, laminar-turbulent transition are still important. To correctly predict heat transfer it is important to accurately predict flow field and then obtain correct prediction of heat transfer coefficients and Stanton and Nusselt numbers.

1.2 Experimental Studies on Effects of Free-stream Turbulence

Sugawara et al [8] conducted one of the first studies on turbulence. They used a hot wire anemometer to measure grid generated turbulence in the boundary layer and free-stream. They concluded that the transition Reynolds number decreases and transition occurs earlier when the FST intensity increases. The turbulent boundary layer thickens but the viscous sub-layer gets thinner which enhances the heat transfer. They found that the heat transfer in the laminar region is unaffected. Nusselt number (Nu_0 was found to increase sharply with increasing FST at low intensity but the increase of Nu becomes less for higher turbulence intensity. Heat transfer coefficient was reported to increase approximately 55% at 7-8% turbulent intensity compared to the low FST level.

Simonich & Bradshaw [9] reported 5% increase of Stanton number for every 1% increase in grid generated turbulent intensity for high $Re_\theta = 6500$. Reynolds analogy factor $\frac{2St}{C_{f,x}}$ increased significantly implying that heat transfer rate increases more than skin friction coefficient due to freestream turbulence. Dissipation length scale was found to be about 1.1 times of the integral length scale for grid generated turbulence. Stanton number decreases with the increase in the length scale, though they did not rule out the possibility of this decrease due to low Reynolds number effect since large length scale to boundary layer thickness ratios were obtained at low Reynolds number.

Hancock & Bradshaw [6] measured the effect of grid-generated nearly isotropic turbulent free-stream on two-dimensional incompressible content-pressure turbulent boundary layer. They observed that the skin friction coefficient varies non-linearly with the freestream turbulence intensity. They also reported the effect of length scale of the boundary layer in addition to freestream turbulence intensity. The effect of freestream turbulence decreases with increasing length scale due to reduction of normal component of velocity by the solid wall. Length scale obtained from the decay equation was given by:

$$L_e^2 = \left(\frac{\bar{u^2}}{U^2}\right)^{-0.3}(0.8A)\Big(\frac{1}{M} - B\Big) \tag{1.2}$$

A and B are constants based on grid spacing M.

Hancock & Bradshaw [10] conducted experiments to examine effects of free stream turbulence for wide range of $\frac{\Lambda_f}{\delta}$ and developed a correlation for dissipation length scale as

$$L_u^\infty = \frac{-(\bar{u_\infty^2})^{\frac{3}{2}}}{U_\infty\big(\frac{d\bar{u_\infty^2}}{dx}\big)} \tag{1.3}$$

For grid generated turbulence, dissipation and longitudinal integral length scale were related as

$$L_u^\infty \approx 1.5\Lambda_f \tag{1.4}$$

Blair and Werle [7] reported length scales of the order $0.2 \leq \frac{\Lambda_f}{\delta} \geq 1$. They conducted baseline test at $Tu = 0.25\%$. They found that for zero pressure gradient turbulent boundary layer, skin friction increases with increasing FST level. The increase is about 14% for Tu increase of 5% at the same Re_θ. For Stanton number, the increase was reported as 18% for same flow conditions.

Blair and Ewards [11] continued the work and measured grid generated turbulent intensities ranging from 0.5 % to 7 %. They found that Stanton number increased at a higher rate as compared to the skin friction coefficient. They modified Hancock's and Bradshaw's correlation to account for low Re_θ effects that provided good prediction of effects of FST in skin friction. They further added comprehensive measurements for turbulent kinetic energy profiles at several downstream locations.

Macmullin et al [12] studied the effect of longitudinal turbulence intensity (7% to 18%) parameter on heat transfer of a circular tangential wall jet over a constant heat flux surface. They found skin friction coefficient and Stanton number to increase with turbulent intensity. Reynolds analogy factor also increased upto 12% intensity, remained constant and then decreased after 15% intensity. Their length scale study was found to be inconclusive.

Yavuzkurt & Batchelder [13] developed new heat transfer coefficient correlation as a function

of free stream turbulence number, α, which is product of length scales of fluctuating velocity in three directions and the mean square velocity in that direction for $10\% < \text{Tu} < 18\%$. The length scale in the flow direction was measured using auto-correlation based on Taylor′s hypothesis. They found Stanton number to be a function of Reynolds number characterized by convection and diffusion. At higher FST, convection is still based on freestream velocity but diffusion is governed by freestream turbulence. Stanton number was found to be $St = C\alpha^n$ and

$$\alpha = \frac{V_i \bar{u_i^2}}{\delta^3 U^2} \tag{1.5}$$

where V_i are quantities relative to the volume (or mass) in the three directions.

Dullenkopf & Mayle [14] studied the effect of free-stream turbulent length scale on heat transfer in laminar boundary layer. They proposed a model that accounted for effective turbulent intensity based on dominant frequency of laminar boundary layer. They used both turbulence level and length scale to correlate heat transfer data for laminar stagnation flows and found heat transfer to be linearly dependent on effective freestream turbulence intensity.

Thole and Bogard [15] measured boundary layer statistics (mean and rms velocities, velocity correlation coefficients, length scales and power spectra) for the effect of 10 to 20 % freestream turbulence on a turbulent boundary layer. They found that large scale turbulent eddies penetrate to within $y^+ = 15$ of the wall but the mean velocity profile still exhibited log-linear region. Though the outer region was significantly altered since the defect velocities in the wake region was significantly reduced.

Barrett & Hoffmann [16] studied the effects length scale ($\frac{L_e}{\theta}$ from 4 to 32) and intensity (Tu = 0.1% to 8%) on skin friction coefficient for momentum Reynolds number varying from 225 to 2700. They found friction velocity correlation that uses only freestream information. They used only few available input parameters eliminating the need of momentum thickness Reynolds number. They used modified Hancock-Bradshaw (HB) parameter developed by Blair [11] to represent their analysis. They obtained C_f enhancement of 6 to 16%. They developed a new model based correlation for skin friction coefficient that did not require knowledge of momentum-thickness Reynolds number. Correlation was evaluated using data with Re_θ up to 6300, FST upto 29% and length scale to momentum boundary layer thickness upto 190.Their model was based on the assumption that for small streamwise pressure gradients u+ profile in the viscous sub-layer and the buffer layer remained unchanged. Final expression for skin friction coefficient was obtained as

$$\sqrt{C_{f,x}/2} = 0.117 L_e^{+(-1/18)} Tu_\infty^{0.18} \tag{1.6}$$

The authors stated that freestream velocity, length scale and turbulent intensity were sufficient to define skin friction coefficient. But the length scale was non-denationalized using friction velocity. In all past studies, friction velocity was correlated based on the streamwise Reynolds number or momentum Reynolds number. This seems that the authors had hard-wired this correlation and

it would fail to provide sufficient results for other studies.

Gibson et al [17] measured mean flow and turbulence that developed on a flat plate followed by a curved surface. Convex curvature reduced turbulent intensity, shear stress and wall frition by approximately 10% as compared to the flat plate.

Schultz & Volino [18] investigated flow over concave surface with high FST ($Tu_\infty = 9\%$) and high acceleration ($K = \frac{v}{U_w^2}\frac{dU_w}{dx} = 9 \times 10^{-6}$) and found that curvature plays an important role in increasing skin friction coefficient by 40% as compared to the baseline case. This was due to the movement of transition location upstream of the flow. They reproduced this case on a flat test wall and strong concave curvature ($\left(\frac{r}{\theta_s}\right)^{0.5} = 27$).

Stefes & Fernholz [19] studied the influence of turbulent intensity (less than 13%) and ratio of integral length scale to boundary layer thickness less than two on flat plate boundary layer with zero pressure gradient. Increase in skin friction was found to be 34% and the data correlated satisfactorily with Hancock-Bradshaw parameter.

Nix et al [20] generated high intensity (10-12%) and large length scale (integral scale = 2 cm) freestream turbulence at the entrance of the cascade. They found increase in heat transfer coefficient by 8% on suction side and approximately 17% on pressure side. Their study was conclusive in terms of determining the effect of turbulence on curved surface but had few data location and boundary conditions were not fully described.

Hylton et al [21] measured external convective heat transfer coefficient on a C3X and MarkII 2-D linear airfoil cascades. They employed both analytical and experimental strategy. Experiments were first performed to determine internal and external boundary conditions at steady state and then a finite element solver was used for calculating the blade conduction to obtain normal gradients of temperature on the surface of the blade, using which the external convective heat transfer coefficients were predicted. Their study formed basis for several researchers to analyze effects on curved surface.

Ames [22] conducted experimental research on four vane linear cascade at chord based exit Reynolds number of 500,000 and 800,000 to examine the influence of large scale high intensity turbulence on vane heat transfer corresponding to initial turbulence levels of 1 %, 7.5 %, 8 % and 12 %. He found heat transfer to be affected by length scale and found good match at the stagnation region with Ames and Moffat [23] correlation.

Dees et al [1] performed experimental and computational study of conjugate heat transfer effects on a large scale model of C3X turbine blade of Hylton [21]. Vane geometry was scaled 3.9 times which resulted in chord length of 56.2 cm, span of 54.9 cm, and vane cascade pitch of 45.7 cm. Inlet velocity of $U_\infty = 5.8m/s$ was selected based on chord Reynolds number of

$Re_c = 750,000$ with turbulence intensity of 0.5% and integral length scale $\Lambda \approx 3cm$. There was no clear edge due to variation of velocity outside the boundary layer. So, velocity was measured at each location and a linear curve fit to extrapolate velocity distribution outside the boundary layer to the wall was done to determine the velocity if viscous affects of the wall were absent. Thickness was 99% value of the extrapolated velocity. They used $k-\omega$ model in CFX v11.0, and used temperature dependent viscosity using Sutherland's model. They found boundary layer to be laminar for most of the region on the pressure side due favorable pressure gradients.

1.3 Numerical Studies on Effects of Free-stream Turbulence

Chen et al [24] modified standard $k-\epsilon$ model using Myong and Kasagi's low-Reynolds number model that used damping function f_μ based on Reynolds number, $R_y = (\sqrt{k}y)/\nu$ and $R_t = k^2/\nu\epsilon$, and does not require wall distance $y+$. They modified pressure velocity correlation and included in their model. Pressure-velocity term showed improvement in near wall prediction of turbulence kinetic energy for flow over flat plate. The formulation is based on Frost and Moulden (1977) idea:

$$\Pi = C_\Pi \rho k (\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}) \tag{1.7}$$

This term is significant in the near-wall region since both the gradients in TKE and velocity are high but at far wall distance both TKE gradient and velocity gradient are small so this term becomes negligible.

The Lam-Bremhorst model solves for ε itself and therefore the term D is prescribed as zero. The f_μ function for this model shows the correct variation in the near wall region but tends to unity somewhat slowly as compared to the Launder-Sharma model. This model also employs the function f1 to model the appropriate growth of ε in the region very close to the wall. The f2 function in this model is modified in order to yield a zero value on the wall by simply omitting the factor 0.3 from the Launder-Sharma model. Thus the sink term in the ε equation is damped leading to the expected rapid increase of viscous dissipation as the wall is reached. [25]

Philip et al [26] showed that the ε boundary condition for Lam-Bremhorst model is not unique and derived an equivalent boundary condition to be enforced at the wall as

$$\frac{\partial \varepsilon}{\partial y} = 0 \tag{1.8}$$

Henkes and Hoogendorn [27] compared turbulence models for natural convection boundary layer along a vertical heated plate and found that setting $\varepsilon = 0$ at wall leads to only small changes for Lam-Bremhorst low-Reynolds number model.

Kwon and Ames [28] formulated eddy viscosity based on normal component of turbulence and length scale since the effect of freestream turbulence on boundary layer development was found to be significantly reduced due to strong attenuation of normal component of turbulence

by the wall. The model accounted for anisotropy of the dissipation and the reduced length of mixing due to high strain rates in the near wall region.

Iyer and Yavuzkurt [29] compared four well known low Reynolds number model for the effect of high freestream turbulence on Stanton number and skin friction coefficient. They showed that the predictions were good for FST $< 5\%$ but became poorer for higher value of FST. They found that for FST value of 26%, Stanton number was over-predicted more than 50% whereas TKE was under-predicted more than 50%.

Iyer and Yavuzkurt [3] introduced a new TKE diffusion based model to include the effect of free stream turbulence and length scale. Results for Stanton number and skin friction coefficient were found to be within 3% of experimental data for Tu=6.53% and the model worked better than the existing low Reynolds number model for Tu=25.7%. Turbulent initial profiles of mean velocity, temperature, TKE and dissipation rate were derived from mixing length arguments since all initial profiles are not reported. They recalculated value of c_μ based on the experimental data sets for $\frac{\overline{u'v'}}{k}$ of Blair & Edwards [11] and Yavuzkurt & Batchelder [13].

Table 1.1: Diffusion coefficient for the high FST TKE model of Iyer and Yavuzkurt [3]

Tu (%)	C_{FST}
1	–
6.53	0.015
25.7	0.01

Aldemir and Yavuzkurt [30] implemented a modified turbulent kinetic energy transport equation in ANSYS FLUENT code using user defined functions (UDF). Launder and Spalding high Reynolds number $k-\epsilon$ model was modified by turbulent diffusion term, YI-diffn, developed by Iyer and Yavuzkurt [3]. They curve fitted the experimental data for c_μ to obtain a polynomial expression.

$$c_\mu = 31.383Tu^4 - 24.36Tu^3 + 7.6112Tu^2 - 1.1942Tu + 0.084 \tag{1.9}$$

They were able to correctly predict TKE profile for moderate FST (6.53%) but under-predicted it under high FST conditions. One of the reasons may be the use of high Reynolds number Launder and Spalding's model instead of using a low-Reynolds number model. This can be a good starting point to implement a low Reynolds number model in FLUENT using UDF. This can be done by including a damping function f_μ near the wall. Heat transfer characteristics can be further improved by including the effects of turbulent Prandtl number (Pr_t) on the calculation of Stanton number.

Foroutan and Yavuzkurt [31] developed a general 3-D model based on YI-diffn model and implemented it in OpenFOAM CFD code such that it is independent of coordinate rotation. The model is based on low-Reynolds number $k-\epsilon$ model of Launder & Sharma. The additional term for redistribution of TKE under high FST conditions models the diffusion of high level turbulence fluctuations in the free stream flow into the boundary layer. The results for skin friction

coefficient and turbulent kinetic energy were improved significantly for moderate FST (6.53 %) and high FST (25.7 %) as compared to the standard LS model.

Mathur and He [32] implemented Launder & Sharma low Reynolds number turbulence model in Fluent using UDF for a pipe flow. They found that the Fluent UDF LS model behaves differently than the in-built Fluent LS model. The latter gives inconsistent results with the available experimental data and in-house CFD code data but the former agrees well with the literature. The authors used the UDF to demonstrate that Launder-Sharma model is sensitive to interpretation of model formulation but is not sensitive to numerical method.

Dyson et al [2] compared 4 different RANS model (realizable $k-\epsilon$, RNG $k-\epsilon$, $k-\omega$ SST, Transition SST) for prediction of hydrodynamics and thermal boundary layer with the experimental measurements of Dees et al [1] on the pressure side and suction side of model C3X vane. Ideal gas law was used to compute density but the variation was negligible over the range of temperature. They used prismatic layer near the walls and unstructured grid away from the wall. Temperature monitor was created at $s/d = 9$ to check convergence criteria. They found the models to over-predict the thermal boundary layer thickness even for low turbulence with the best results deviating more than 15% from the experimental data. They studied the effect of turbulent Prandtl number variation but found it to be inadequate to improve thermal boundary layer predictions. $U_\infty = 5.8m/s$ was used for low turbulence (Tu = 0.5% and $\Lambda_f = 320mm$) and high turbulence (Tu = 20% and $\Lambda_f = 37mm$) at the inlet. They started the calculation at x/c = -0.26 with Tu = 47% and integral length scale 3.1 mm for high turbulence case to match with the experimental condition at x = 0. $T_{wall} = 330K$ and $T_\infty = 305K$ were used and periodic boundary conditions were used to produce the cascade. Final discretization scheme was second order for all parameters except pressure where it was 1st order. $k-\varepsilon$ was found inadequate to because it lacked transition.

Luo et al [33] assessed v2-f, SST, k-ε and realizable k-ε RANS models to predict gas side heat transfer coefficients on airfoil and end-walls. 3D model was able to capture the secondary flow vortexes and thus provided more accurate local values of adiabatic temperature. They concluded that SST model performed better than other models.

Balogh et al [34] modified realizable k-ε model in FLUENT to first convert it to standard k-ε and then implement the model modifications since realizable k-ε uses variable c_μ. They mentioned that standard k-ε model in FLUENT makes it impossible to modify c_μ.

Cotas [35] tested and modified low Reynolds number models in FLUENT using User Defined Function to simulate the flow of pulp inside a pipe. Launder-Sharma inbuilt model in FLUENT showed largest deviations of pressure drop as compared to other four LRN models tested which further highlight the findings by the work of Mathur and He [32].

Furbo [36] compared flow separation at sharp and smooth surfaces using several RANS models in OpenFOAM with LES results and experimental data. Models were found to have problem predicting complex dynamics of flow separation. In addition, wall function implementation for k-ε model was studied in detail its comparison was provided.

Garcia [37] studied two-dimensional compressor cascade flow in OpenFOAM for the effects of various solver parameters. He found that unstructured grids gave quicker solution but structured grid gave more accurate results when compared with the data.

Jones et al [38] studied RANS models in OpenFOAM using semi-implicit pressure linked equations (SIMPLE) algorithm for a two dimensional flow along a curved ramp and 3-D flow around a hull of a generic submarine. Detailed study of discretization schemes, solver parameters and boundary conditions were presented. It was found that the results matched well with that of Fluent when first order upwind scheme was used for all variables except velocity for which second order upwind scheme was used. Anderson and Bonhaus [39] performed validation study for compressible flow cases in OpenFOAM 'extend'. They found run time of OpenFOAM solver to be much higher (approximately 10 times) than of commercial solvers and suggested several acceleration techniques.

Mangani et al [40] developed solver based on SIMPLE-C All-Mach algorithm to treat pressure corrector for highly compressible flow implemented in OpenFOAM. They used the solver to predict conjugate heat transfer from first stage turbine vane cascade based on 1988 C3X NASA setup. They found SST turbulence model to give reasonable prediction for heat transfer.Mangani and Bianchini [41] developed a steady state solver to solve several flow regimes including incompressible, high Mach number to simulate turbine like behavior and implemented it in OpenFOAM. They tested low Reynolds number model and kwSST model for various test cases that included impinging jets, film cooling and effusion cooling. They presented OpenFOAM as an effective substitute to commercial softwares for such complex flows.Laskowski et al [42] performed coupled simulations of solid and fluid domain for NASA C3X vane and VKI rotor with film cooling. They used SST model with near wall resolved grid. They found numerical simulations to constantly under-predict surface heat transfer coefficients under the effect of film cooling as compared to the experiments.

Erney [43] validated single phase solutions on a flat plate, hemispherical head-form and NACA0012 airfoil using OpenFOAM and expanded it to study cavitation flow. LS model with resolved boundary layer was found fairly accurate to correctly predict skin friction at low turbulence level of 1 %. He concluded that y+ between 0.1 and 1 gives accurate results in OpenFOAM using LRN model. Jones [44] conducted CFD study of flow around submarines and compared the results from Fluent and OpenFOAM. He found difference of upto 15 % for drag coefficient between the two softwares. He compared the effect of different divergence schemes in OpenFOAM on the results and found k and ε to be independent of the scheme and suggested first

order *upwind* scheme for these whereas *linearUpwind* was found sufficient for velocity to provide good results. All the simulations were performed using standard k-ε model.

1.4 Conclusions from Literature Review

Extensive experimental and computational research conducted over the past few decades on the effect of incoming turbulence in the free-stream give evidence that both the free-stream turbulence intensity as well as the length scale of turbulence has significant effect on heat transfer from a solid wall. Free-stream Turbulence (FST) increases the skin friction coefficient and Stanton number as compared to the baseline case of low FST levels. It is also important to predict the transition from laminar to turbulent flow. Boundary layer transition itself is dependent on several factors such as FST, surface roughness, pressure-gradient and compressibility. Results from the flat plate experiments and computation can be directly used with slight modification if the curvature is not significant when the radius of curvature is very large compared to boundary layer thickness.

Computational Fluid Dynamic (CFD) analysis acts as an inexpensive and time saving alternative to experiments with the capability of simulating complex systems. It is able to provide satisfactory results for the low FST case but the difference in the experimental and numerical results increases for high FST. This is primarily due to the type of turbulence model selected for the study which may not be applicable over all flow conditions. The main problems encountered by the models are the poor prediction of turbulent kinetic energy profiles near the wall and inability to correctly predict transition from laminar to turbulent flow. Most models also fail when the flow separates as in case of curved surfaces and adverse pressure gradients. Also, they do not correctly predict diffusion of FST towards the wall.

Fluent and OpenFOAM have been extensively used for flow simulation and these softwares have been extensively validated. Fluent is a commercial software with robust solvers for several flow simulations. OpenFOAM is an open source software with an active community support that continuously extend the capabilities of this software.

1.5 Objectives

Near wall predictions of the standard models can and should be improved by better prediction of turbulent kinetic energy (TKE) since TKE is used to calculate turbulent transport coefficient μ_t or ν_t . The main objective of the present research is to implement TKE diffusion model to predict heat transfer coefficients under FST on a gas turbine blade which can be listed into sub-topics as:

- Study of popular turbulence models used in CFD analysis
- Sensitivity of solution to starting profiles of k and ε
- Implementation of variable c_μ in Low Reynolds Number (LRN) models

- Implementation of additional diffusion term to TKE transport equation for better prediction of aerodynamics and heat transfer
- Study of flow over curved surface using inbuilt models
- Study of flow over curved surfaces using the high FST TKE diffusion model for the purpose of improving heat transfer predictions

Chapter 2
Governing Equations & Turbulence Models

The present study utilizes two-equation based Reynolds Average Navier Stokes (RANS) models. Standard k-ε model based on Launder and Spalding [45] with near wall corrections, realizable k-ε model proposed by Shih et al. [46] and standard k-ω model of Wilcox [47] are used for preliminary analysis. Further studies are based on modification of Launder and Sharma [45] low-Reynolds number k-ε model with modifications suggested by Iyer and Yavuzkurt [48] which incorporates effect of free-stream turbulence at the near wall flow properties. This model was initially developed for 2-D boundary layers implemented in TEXSTAN CFD code and was converted into tensorial form by Foroutan and Yavuzkurt [31] and implemented in OPENFOAM [49] CFD code. There is no universal model and individual models are suitable under different flow conditions. So, it is important to know what factors impact the flow physics before using them for Computational Fluid Dynamics (CFD) analysis. Some of the important things to keep in mind while performing CFD analysis of complex flow systems are:

i Adverse pressure gradients

ii Flow separation

iii Laminar to Turbulent transition

iv Effect of walls (ex. turbine vanes) or free shear flows (ex. jets)

v Swirl

Flow physics is just one aspect of model selection but there are certain practical limitations which should be kept in mind:

i Computational cost

ii Convergence and robustness

iii Near wall treatment

iv Accuracy

For example, Sparlart-Allarmas (SA) one-equation model [50] may be computationally inexpensive but on the other hand Reynolds Stress Model (RSM), a five equation model, may yield superior answer but is highly computationally demanding and has convergence issues. So, there is always a compromise in the selection of the model based on our requirements and how much one is willing to pay.
So, before selecting a model it is basic requirement to understand the underlying equations which define flow physics.

2.1 Governing Equations

For a compressible Newtonian fluid, governing equations are:

Conservation of mass

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u_i)}{\partial x_i} = 0 \tag{2.1}$$

Conservation of momentum

$$\frac{\partial \rho u_i}{\partial t} + \frac{\partial(\rho u_i u_j)}{\partial x_j} = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho g_i \tag{2.2}$$

Conservation of energy

$$\frac{\partial \rho e}{\partial t} + \frac{\partial(\rho e u_j)}{\partial x_j} = -p\frac{\partial u_j}{\partial x_j} + \frac{\partial}{\partial x_j}\Big(k\frac{T}{\partial x_j}\Big) + \phi \tag{2.3}$$

In equation 2.2 , τ_{ij} is viscous stress and for Newtonian fluid assuming that Stokes Law is applicable, it becomes

$$\tau_{ij} = 2\mu S_{ij} \tag{2.4}$$

S_{ij} is the viscous strain-rate given by

$$S_{ij} = \frac{1}{2}\Big(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\Big) - \frac{1}{3}\frac{\partial u_k}{\partial u_k}\delta_{ij} \tag{2.5}$$

e is the internal energy given by $e = h - \frac{p}{\rho} + \frac{u_i u_i}{2}$, where h is sensible enthalpy. ϕ is known as viscous dissipation and is defined as thermal energy created by viscous shear flow. It is given as

$$\begin{aligned}\phi =& \mu\Bigg[2\Big\{\Big(\frac{\partial u}{\partial x}\Big)^2 + \Big(\frac{\partial u}{\partial x}\Big)^2 + \Big(\frac{\partial u}{\partial x}\Big)^2\Big\} \\ &+ \Big(\Big(\frac{\partial u}{\partial x}\Big) + \Big(\frac{\partial u}{\partial x}\Big)\Big)^2 + \Big(\Big(\frac{\partial u}{\partial x}\Big) + \Big(\frac{\partial u}{\partial x}\Big)\Big)^2 + \Big(\Big(\frac{\partial u}{\partial x}\Big) + \Big(\frac{\partial u}{\partial x}\Big)\Big)^2 + \lambda\frac{\partial u_i}{\partial x_i}\Bigg]\end{aligned} \tag{2.6}$$

ϕ is generally negligible for incompressible flows and should be considered in CFD calculations when Brinkman number $Br_n > 1$.

The above mentioned flow equations 2.1 and 2.2 are coupled since the unknowns u_i appear in each of them and must be solved simultaneously. Furthermore, the presence of non- linear viscous term like $\frac{\partial \tau_{ij}}{\partial x_j}$ and ϕ causes an analytical solution of these equations impossible, except for a few simplified cases. Therefore, CFD is the only way of obtaining solution of these equations when dealing with complex flow situation.

2.2 Basics of Turbulence

Fluid flows which are unsteady, irregular and in which transported quantities fluctuate in time and space are termed as a turbulent flows. Turbulent flow contains a wide range of length scales in form of energy carrying rotational flow structures called eddies.
Energy is transferred from larger eddies to smaller eddies via the process of vortex stretching. Presence of mean velocity gradient in a turbulent flow stretches and distorts the large eddies. Larger eddies are essentially inviscid in nature and their angular momentum is conserved during the process of vortex stretching. Kinetic energy is transferred from large eddies to smaller eddies due to more vortex stretching of smaller eddies thereby forming an energy cascade. Smaller eddies are viscous dominant whereas inertial forces dominate the larger eddies. The larger eddies are highly anisotropic while the smallest eddies in a turbulent flow are isotropic.
Ratio of the length scale of turbulence is dependent on the flow Reynolds number as $Re^{3/4}$. The computational cost to fully resolve all the eddies using Direct Numerical Simulation (DNS) is huge even for very simple flows. So, to solve complex flows it becomes necessary to model some of the length scales and eddies.

2.3 Reynolds Average Navier-Stokes (RANS)

Instantaneous turbulence flow property can be denoted as

$$\phi = \bar{\phi} + \phi' \tag{2.7}$$

The mean flow property can be defined as the ensemble average of the instantaneous flow property,

$$\bar{\phi} = \frac{1}{\Delta t} \int_{t}^{t+\delta t} \phi(t) dt \tag{2.8}$$

Now taking the ensemble average of the velocity, pressure and temperature, and denoting the mean flow parameters by capitals and fluctuating parameters by small caps, equation (2.1), (2.2) and (2.3) can be re-written as RANS equations for: Continuity,

$$\frac{\partial \rho}{\partial t} + \frac{\partial (\rho u_i)}{\partial x_i} = 0 \tag{2.9}$$

Momentum,

$$\frac{\partial \rho U_i}{\partial t} + \frac{\partial}{\partial x_j}(U_i U_j) = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}(\mu \frac{\partial U_i}{\partial x_j}) - \frac{\partial (\rho \overline{u_i u_j})}{\partial x_j} \tag{2.10}$$

Here, the last term $-\rho\overline{u_i u_j}$ is called Reynolds stress tensor and is denoted by $\tau_{ij}^{'}$. These add six additional unknowns to the system with only four equations and lead to the problem of turbulence closure. Thus Reynolds stress need to be modeled to solve the RANS equations.

2.4 Turbulence Models

One of the most common approach to tackle the turbulence closure problem is through Bousinesq's hypothesis which states that the Reynold stress can be linked to mean strain rate using eddy viscosity or turbulent viscosity, μ_t.

$$-\rho\overline{u_i u_j} = \mu_t\Big(\frac{\partial U_i}{\partial x_j} + \frac{\partial U_j}{\partial x_i}\Big) - \frac{2}{3}k\delta_{ij} \tag{2.11}$$

This reduces the effort to now find only one unknown which is eddy viscosity. Bousinesq's hypothesis is reasonable for many practical application such as boundary layer flows, jets, mixing layers, and offer reduced computational cost but the main disadvantage is that it assumes eddy viscosity to be isotropic which is not completely true. There can be one-equation (Spalart-Allarmas), two-equation (k-ε, k-ω), 4-equation (Durbin v2f) models based on the number of equations solved to obtain the eddy viscosity. Low Reynolds number k-ε, realizable k-ε and k-ω SST models are used in the present study and are discussed in brief.

2.4.1 Standard k-ε Model

This model was developed by Launder and Spalding [51] and is the most widely used RANS turbulence model. The original model is two equation semi-empirical high Reynolds number model based on transport of turbulent kinetic energy, k, and dissipation rate, ε. k transport equation,

$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho k U_j}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + G_k + G_b - \rho\varepsilon - Y_M + S_k \tag{2.12}$$

ε transport equation,

$$\frac{\partial \rho \varepsilon}{\partial t} + \frac{\partial \rho \varepsilon U_j}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial \varepsilon}{\partial x_j}\right] + \frac{C_{1\varepsilon}\varepsilon}{k}\Big(G_k + C_{3\varepsilon}G_b\Big) - \frac{C_{2\varepsilon}\rho\varepsilon^2}{k} + S_\varepsilon \tag{2.13}$$

where G_k is production of k due to mean velocity gradient, G_b is production due to buoyancy and Y_M is dissipation due to compressibility.
Eddy viscosity is modeled as,

$$\mu_t = \frac{\rho c_\mu k^2}{\varepsilon} \tag{2.14}$$

The constants are: $C_{1\varepsilon} = 1.44$, $C_{2\varepsilon} = 1.92$, $c_\mu = 0.09$, $\sigma_k = 1.0$, $\sigma_\varepsilon = 1.3$.

It is robust and computationally economic, applicable to wide range of flows and widely validated. Some of the disadvantages are that it assumes isotropic eddy viscosity, overly diffusive for many situations, dissipation rate has be solved using wall functions near the wall, and performs poorly in strong separation, streamline curvatures and low Reynolds number flows.

2.4.2 Realizable k-ε

This was developed by Shih et al. [46] to meet mathematical conditions of positive normal Reynolds stress ($\overline{u_i u_i} > 0$) and Schwarz's inequality $\overline{u_i^2 u_j^2} \geq \overline{u_i u_j}^2$. It is based on standard k-ε model with some changes :

- c_μ is a variable, $c_\mu = \frac{1}{A_o + \frac{A_s k U}{\varepsilon}}$
- Transport equation for ε is based on mean square vorticity fluctuations

ε transport equation is modified as,

$$\frac{\partial \rho \varepsilon}{\partial t} + \frac{\partial \rho \varepsilon U_i}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial \varepsilon}{\partial x_j}\right] + \frac{C_{1\varepsilon}\varepsilon}{k} C_{3\varepsilon} G_b + \rho C_1 S_\varepsilon - \frac{\rho S_\varepsilon \varepsilon^2}{k + \sqrt{\nu\varepsilon}} + S_\varepsilon \tag{2.15}$$

where, $C_1 = max\left[0.43, \frac{\eta}{\eta+5}\right]$, $\eta = S\frac{k}{\varepsilon}$, $S = \sqrt{2S_{ij}S_{ij}}$.
The constants in realizable k-ε models are: $C_{1\varepsilon} = 1.44$, $C_{1\varepsilon} = 1.9$, $\sigma_k = 1.0$, $\sigma_\varepsilon = 1.0$. Due to realizability this model predicts better spreading rates for both planar and round jet, performs better in separation and adverse pressure gradients, and is effects for flow with large strain rates. The main disadvantage is that it produces non-physical turbulent viscosity in cases of sliding mesh where there are both rotating and stationary fluids [52].

2.4.3 Low Reynolds Number Model

Turbulent viscosity formulation of the standard k-ε is modified to by adding a damping function along with some changes in the k and ε transport equation to make it applicable for wall bounded shear flows. This forms Low Reynolds Number (LRN) turbulence models. General transport equation for these models are given as:

$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho k U_j}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + G_k - \rho\varepsilon \tag{2.16}$$

$$\frac{\partial \rho \varepsilon}{\partial t} + \frac{\partial \rho \varepsilon U_i}{\partial x_j} = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial \varepsilon}{\partial x_j}\right] + \frac{C_{1\varepsilon} f_1 \rho\varepsilon}{k} G_k - \frac{C_{2\varepsilon} f_2 \rho\varepsilon^2}{k} + E \tag{2.17}$$

Here, modified dissipation rate is given as

$$\tilde{\varepsilon} = \varepsilon - D \tag{2.18}$$

and turbulent viscosity as

$$\mu_t = \rho C_\mu f_\mu \frac{k^2}{\tilde{\varepsilon}} \tag{2.19}$$

Here, D is generally a function of TKE and defined such that the boundary condition for ε can be defined as zero at the wall. Different LRN models can be defined based on the different formulations of D, E, f_1, f_2, f_μ, and model constants.

Low-Reynolds number models are a good choice to study and resolve near wall phenomena and can tackle wide variety of problems as compared to standard models. But on the other hand

Table 2.1: LRN model constants

Model	c_μ	$C_{1\varepsilon}$	$C_{2\varepsilon}$	σ_k	σ_ε
LS	0.09	1.44	1.92	1.0	1.3
LB	0.09	1.44	1.92	1.0	1.3

Table 2.2: LRN model formulations for Launder-Sharma (LS) and Lam-Bremhorst (LB) model

Model	f_1	f_2	f_μ	D	E	$\tilde{\varepsilon}_{wall}$
LS	1	$1-0.3e^{-Re_t^2}$	$e^{\frac{-3.4}{(1+Re_t/50)^2}}$	$2\nu(\frac{\partial\sqrt{k}}{\partial y})^2$	$2\nu\nu_t(\frac{\partial^2 u}{\partial y^2})^2$	0
LB	$1-(\frac{0.05}{f_\mu})^2$	$1-e^{-Re_t^2}$	$[1-e^{0.0165Re_y}]^2(1+\frac{20.5}{Re_t})$	0		$\frac{\partial\varepsilon}{\partial y}=0$

the computational demand increases due to requirement of fine mesh near the wall where viscous sub-layer has to be resolved [53].

2.4.4 k-ω SST Model

It was developed to take advantage of both k$-\omega$ near the wall and k-ε model away from the wall. Standard k$-\omega$ is overly sensitive to freestream value of ω but k-ε is not. Most two equation models over-predict turbulent stresses in the wake region which leads to poor performance of boundary layers under separation or adverse pressure gradients. This model includes transport of shear stress and a blending function to ensure that turbulent stresses are correctly predicted in near wall region and behave properly in the outer region [54]. Turbulent viscosity is modified as

$$\mu_t = \rho\frac{k}{\omega}\frac{1}{max[\frac{1}{\alpha},\frac{SF_2}{\alpha_1\omega}]} \tag{2.20}$$

where S is the strain rate magnitude. Blending function F_1 and F_2 are given as:

$$F_1 = tanh(\phi_1^4) \tag{2.21}$$

$$\phi_1 = min\Big[max(\frac{\sqrt{k}}{0.09\omega y},\frac{500\mu}{\rho y^2\omega}),\frac{4\rho k}{\sigma_{\omega,2}D_\omega^+ y^2}\Big] \tag{2.22}$$

$$D_\omega^+ = max\Big[\frac{2\rho}{\sigma_{\omega,2}\omega}\frac{\partial k}{\partial x_j}\frac{\partial\omega}{\partial x_j},10^{-10}\Big] \tag{2.23}$$

$$F_2 = tanh(\phi_2^4) \tag{2.24}$$

$$\phi_2 = max\Big[2\frac{\sqrt{k}}{0.09\omega y},\frac{500\mu}{\rho y^2\omega}\Big] \tag{2.25}$$

D_ω^+ is the positive part of the cross diffusion term which it used to blend together k$-\omega$ and k-ε.

$$D_\omega = \frac{2(1-F_1)\rho\sigma_{\omega,2}}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j} \tag{2.26}$$

This model has the advantage of better performance under adverse pressure gradient but generally predicts early and excessive separation.

2.5 SIMPLE Algorithm

Semi-Implicit Method for Pressure Linked Equations (SIMPLE) [55] provides an iterative method to solve steady-state flow problems. It can be summarized as:

i Set initial values for all fields

ii Begin the outer iteration loop

iii Assemble and solve under-relaxed momentum predictor

iv Solve pressure equation and calculate conservative fluxes. Update the pressure field and explicitly calculate velocity correction

v Solve other equations (TKE, TDR, Energy) and update pressure and velocity fields.

vi Check convergence criteria or iterate

This is used for all the simulations in the current study.

2.6 CFD Softwares Used for the Current Study

2.6.1 ANSYS Fluent

FLUENT is a state-of-the-art computer program for modeling fluid flow and heat transfer in complex geometries from ANSYS Inc. [56]. The code is available from national fluent vendors for both academic and public institutions by acquiring an annual renewable license. It solves the governing conservation equations of fluid dynamics by a finite-volume formulation on a structured, non-orthogonal, curvilinear coordinate grid system using a collocated variable arrangement. Three different spatial discretization schemes may be used, that is power-law, second-order upwind, and QUICK, a bounded third-order accurate method. Temporal discretization is achieved by a second-order, implicit Euler scheme. Pressure/Velocity coupling is achieved by the SIMPLE algorithm resulting in a set of algebraic equations which are solved using a line-by-line tridiagonal matrix algorithm, accelerated by an additive-correction type of multi-grid method and block-correction. Additional equation solvers are also available to the user. FLUENT models turbulent flows with the standard k-ε model, an RNG model, and a second-moment closure or Reynolds-stress model (RSM).

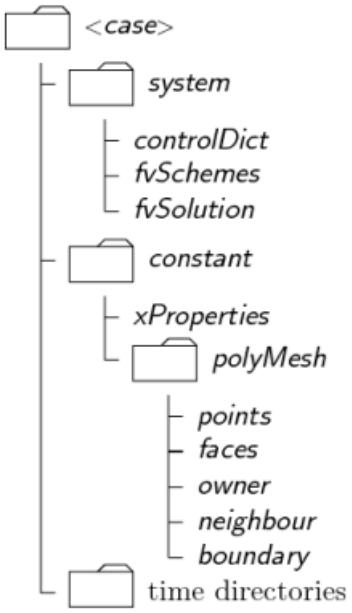


Figure 2.1: Structure of OpenFOAM case

2.6.2 OpenFOAM Code

Open source Field Operation And Manipulation is C++ toolbox for numerical solvers and pre-processing for problems of continuum mechanics [49]. Being an open source software it provides the source code to the user and gives the flexibility to modify the source code. Main portion of the present study is performed in OpenFOAM.

Basic directory of OpenFOAM case is shown in Figure 2.1 and the minimum set of files required to run a simulation are:

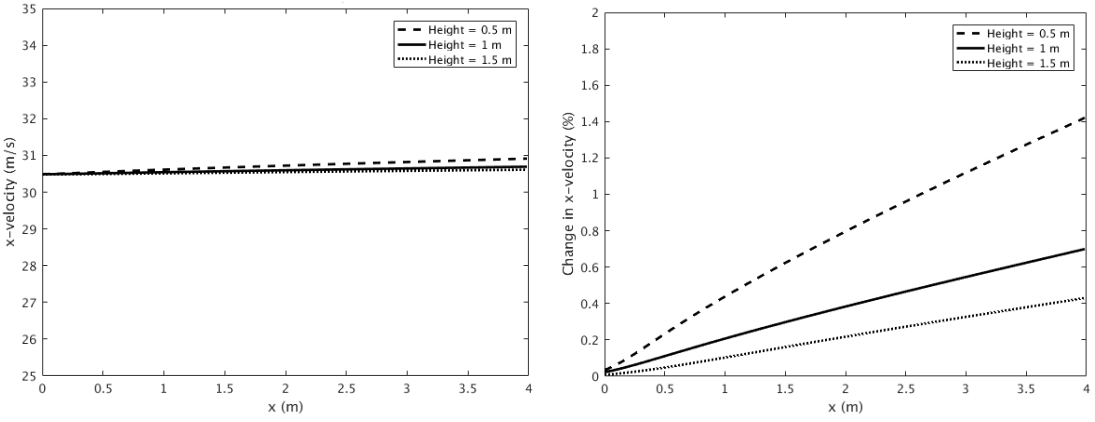
- *constant*: It contains full details of the geometry, mesh, turbulence model used for the simulation, transport and thermo-physical properties needed for the simulation.
- *system*: It contains the parameters to run and control the simulations. It should have minimum of 3 files - *controlDict* to control running paramters such as startTime, stopTime, run time printing; *fvSolution* contains solver details for all the parameters used in the simulation; and *fvSchemes* that contains the details of gradient schemes, divergence schemes, interpolation for the variables used in the simulation. This directory can have additional files to sample, decompose, or post-process the simulation
- *time*: It contains the details of the initial and boundary conditions for velocity, pressure, turbulence. More time directories are created as the solution proceeds containing the results at the iteration number.

Chapter 3
FLUENT Code Validation

3.1 Selection of Domain & Boundary Conditions

Hirsch [57] suggested that it is common to find boundaries at 50 chord lengths for an airfoil and many researchers [58] have used this scaling factor of domain height for flat plate results. But in the previous studies [30, 59] of the research group it has been found that a shorter domain height with the symmetric boundary condition at the top can provide non-accelerated velocity field and eliminate the effect of boundary layer development on flow acceleration. So, it is important to find minimum domain height that would provide non-accelerated flow with velocity change limited to less than 1% as suggested by Vaughan [58].

Plate length, L = 4 m, was taken to allow sufficient high momentum thickness be available at low velocity of 6 m/s. Three test cases were setup with domain heights of 0.2 m, 1 m and 1.5 m respectively. Inlet condition was taken as fully developed turbulent boundary layer profile with δ=0.00615 m and freestream velocity of 30.48 m/s.



(a) Streamwise velocity

(b) Change in the velocity

Figure 3.1: Effect of domain height on flow acceleration over a flat plate test case

It is observed that the flow accelerates by 1% for the domain height of 0.5 m. For the 1

m and 1.5 m this increase is limited to 0.6% and 0.4% respectively. So, it is decided to keep domain height as 0.2 m since it fulfills the requirement of flow acceleration limited to 1% and is computationally less demanding than the larger domain height for same level of mesh refinement. It was observed from Figure 3.2 that no change is observed for shear stress while increasing the mesh points even 4 times. Current study is concerned with the profiles within the boundary layer so at a point two-third downstream law of the wall as shown in Figure 3.3 was plotted for different meshes. No change was observed and number of grid points with $y^+ < 5$ were found to be 12 for the coarsest grid. So, it can be said with confidence that Mesh 4 (Table 3.3) with 180 mesh points in the plate normal direction and 300 grid points along the plate direction can be used for current study.

Table 3.1: Mesh details after grid independence study for flow over flat plate

Mesh no.	*x-cells*	*y-nodes*	*Distance of first cell (m)*	*No. of cells*
1	150	75	1×10^{-5}	10,656
2	200	100	7×10^{-6}	20,158
3	300	150	5×10^{-6}	44,104
4	300	180	3.5×10^{-6}	53,044
5	400	200	3.5×10^{-6}	78,804

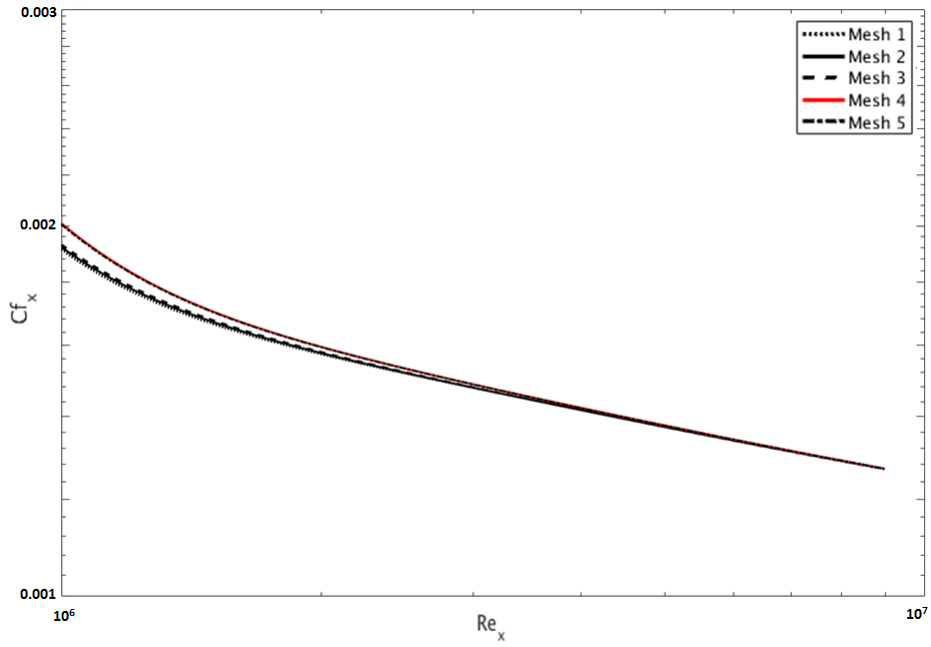


Figure 3.2: Effect of mesh density on skin friction coefficient over a flat plate

There are six low Reynolds number available in FLUENT.

- Abid et al. (0) - *Abid*
- Lam-Bremhorst (1) - *LB*

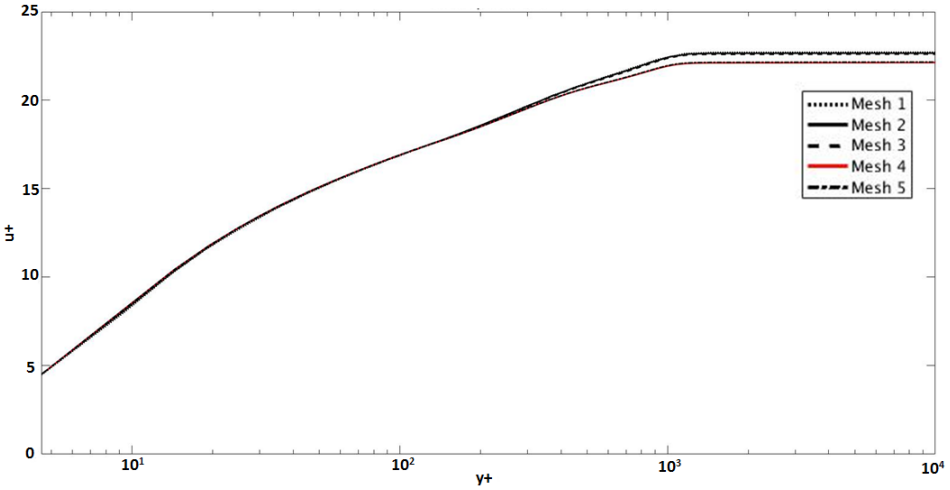


Figure 3.3: Law of the wall comparison for grid convergence test over a flat plate

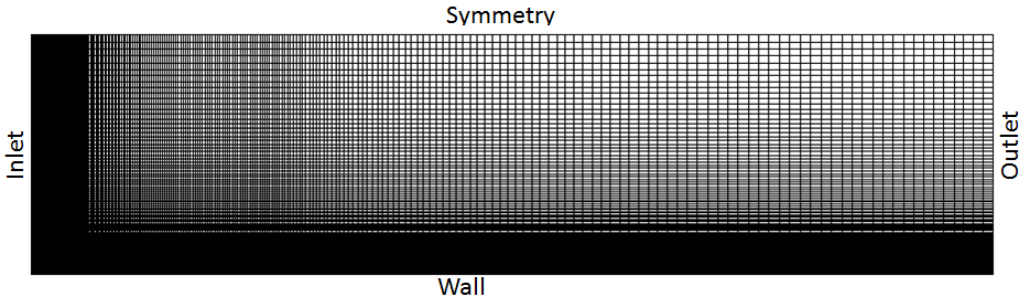


Figure 3.4: Final mesh for flat plate after grid convergence study

- Launder-Sharma (2) - *LS*
- Yang-Shih (3) - *YS*
- Abe-Kondoh-Nagano (4) - *AKN*
- Chang-Hsieh-Chen (5) - *CHC*

These are hidden and can be accessed only thorough text commands.
define/models/viscous/turbulence-expert/low-re-ke
enable [yes]
low-re-ke-index
Here, index is number 0 to 5 mentioned above along with the models, and the abbreviations for the models are mentioned along with the names which shall be used throughout this document.

3.2 Effect of Initial Profiles

It may look straightforward to use uniform profiles at the inlet but their choice may impact the solutions. k and $\bar{\varepsilon}$ decay and vary from the wall value of zero to free-stream value in non-monotonic fashion. Their profile impacts the mixing and thus the velocity profiles. Which in turn is responsible for variation in near wall flux such as shear stress and heat transfer. For the present case of flat plate, it is important to start with the initial values given in the experimental data and reports. But the problem one faces is that in most cases only the freestream values are defined.

3.2.1 Uniform Profiles of Turbulent Variables

The first study focuses on comparing the models with using one-seventh power law profile for velocity and temperature at the inlet but TKE and ε as taken as uniform constant values.
After the baseline analysis for the low Reynolds number model present in FLUENT, the models are compared at higher turbulence intensities. For the comparison two cases are considered. First one is at FST 6.53 % and compared with the data set of Blair and Werle [7] and Blair and Edwards [11]. The second case is taken at higher intensity of 25.7 % which is compared with data set of Ames and Moffat [23] and Yavuzkurt and Batchelder [13]. To be consistent with the previous studies of research group, the case with $Tu_\infty = 6.53\%$ is called moderate FST and with $Tu_\infty = 25.7\%$ is called high FST.

For the baseline case of 1% turbulence intensity all the models except LS seems to work well. CHC works best in predicting skin friction coefficient with the average error being only 2%. YS works reasonably with the average deviation from the data and correlation limited to 4 %. LB model shows an error of 9-10 % which is greater than what Iyer [48] (4-8%) and Foroutan & Yavuzkurt [31] (4-8 %) predicted. LS model over-predicts skin friction by as much as 64 % even at low turbulence intensity.

Prediction of Stanton number follows similar trend as that of the skin friction coefficient with CHC performing the best followed by YS and LB. CHC shows an error of 2-4 % from data, YS overpredicts by 7 % where as LB overpredicts by 12 %. LS models performs poorly with over-prediction of upto 55 %.

All the models correctly predicts the freestream value of TKE but they under-predict in the near wall region and slightly over-predict away from the wall. CHC under-predicts peak TKE by 18 % but all other models over-predict peak TKE value. YS model has peak TKE value only 2 % higher than the data while LB and LS model over-predict by 5 % and 11 % respectively. Iyer reported 10% over-prediction by LB model as compared to other models with gave results in ±4%. The reason for LB model not doing well was due to its sensitivity towards the functions f_1 and f_μ. The over-prediction on TKE can be attributed to the fact that the data was taken at 0.4 % turbulence intensity but all the cases were run at 1 % to prevent numeri-

cal instability. The baseline cases for LS and LB are compared to the predictions by Iyer [29] and Foroutan [31]. LB model give similar results but LS model gives completely incorrect results.

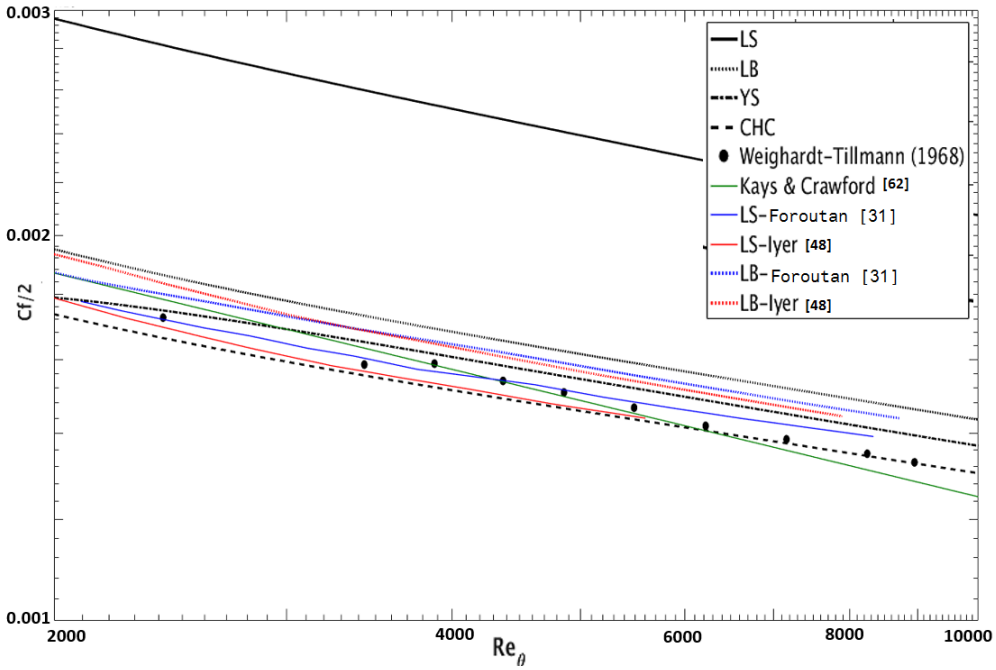


Figure 3.5: Skin Friction Coefficient over a flat plate at Tu = 1% - baseline results

The first thing to observe from the moderate FST in Figure 3.8 case is increase in skin friction as compared to the baseline case. CHC under-predicts skin friction coefficient while all other models over-predict skin friction coefficient. LB model over-predicts on an average by 9 % whereas YS model performs slightly better with average error of 7 %. LB model in FLUENT perform very closely to the TEXSTAN prediction by Iyer [48]. Here also, LS model deviates from the data by up to 84 %.

From Figure 3.9 it can be observed that Stanton number increases with increase in FST and this increase is greater than the increase of skin friction coefficient at the same FST level. CHC matches very well with the data. YS model overpredicts by 2-6 % whereas LB model shows an error of 9 %. LB model of FLUENT matches with the LB model prediction by Iyer [48] for higher Reynolds number. LS model perform poorly with maximum error of 43 %.

All the models predict TKE values poorly in the near wall region as compared to the data in Figure 3.10a and Figure 3.10b. Freestream value for all the models is same. All the models in FLUENT overpredicts peak TKE value as compared to the previous studies based on OpenFOAM and TEXSTAN. YS over-predicts peak TKE by 14 % and LB over-predicts by 16 %. Although LS model over-predicts peak TKE by only 16% but its profile is different from all other models and also the previous studies based on LS models. Higher TKE implies more mixing. In the near wall region the value of TKE is higher as compared to the data which causes the skin friction coefficient and the Stanton number prediction to be higher than the data.

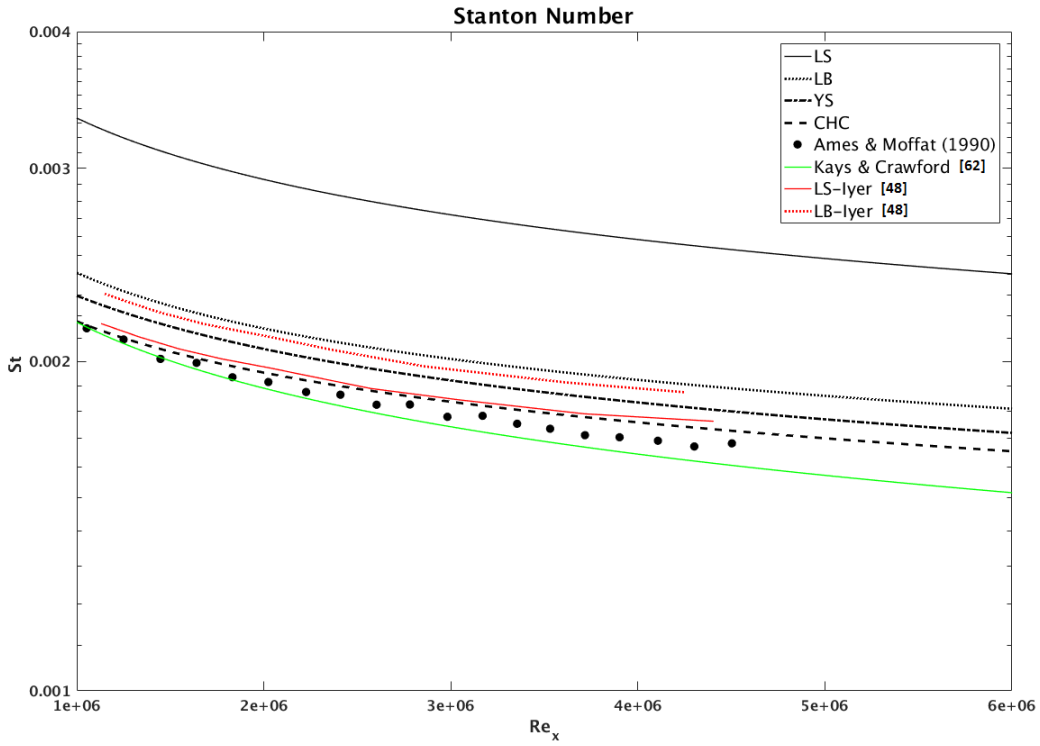


Figure 3.6: Stanton Number over a flat plate at Tu = 1% - baseline results

Significant increase in skin friction coefficient is observed at high FST for all the models. YS deviates from the data by 27 % and LB model deviates from the data by 32 %. LS model over-predicts by more than 100 %. It can be seen in figure 3.11 that LB model of FLUENT does not match with the previous studies and same is with the LS model. But it can also be observed that neither do the two previous studies match with each other. This behavior can be attributed to the fact that TEXSTAN is a boundary layer code whereas OpenFOAM and FLUENT solves the complete equations, and also the inability of models to predict at high FST. Another problem may be the effect of the uniform values of TKE and ε at the inlet as compared to the fully developed profiles.

Near wall TKE value for all the models is under-predicted by almost than 50 % from the data. It was concluded in the previous studies that the correct prediction of TKE should lead correct prediction of skin friction coefficient and Stanton number since TKE is used to calculate turbulent viscosity which affects the momentum and the energy equation. This was done by Aldemir and Yavuzkurt [30] but this led to further increase in Stanton number and skin friction coefficient.

Before moving forward it is important to observe the effect of correct initial profiles of TKE and ε.

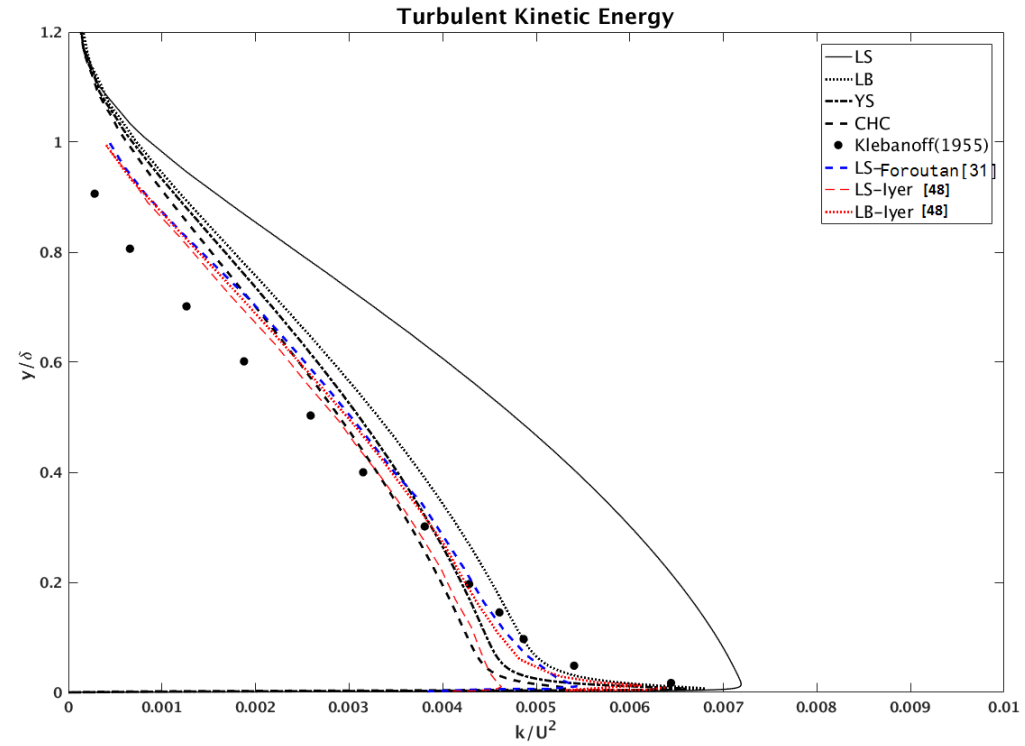


Figure 3.7: Turbulent Kinetic Energy profile over a flat plate for Tu = 1% at Re_θ = 7700 - baseline results

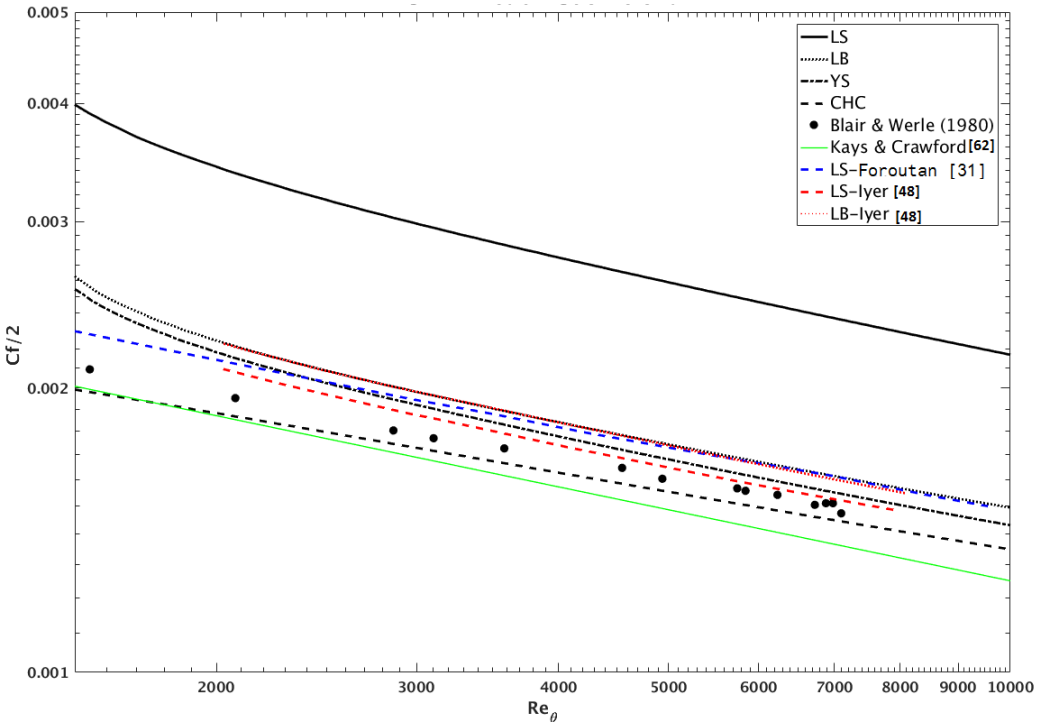


Figure 3.8: Skin Friction Coefficient over a flat plate at Tu = 6.53% - baseline results

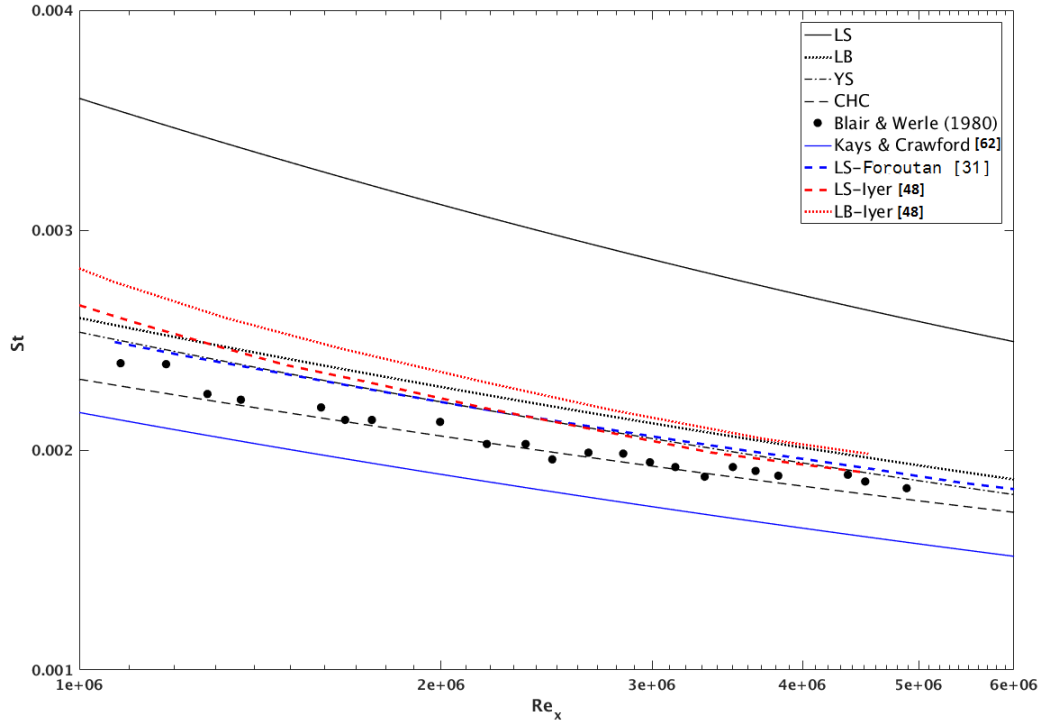


Figure 3.9: Stanton number over a flat plate at Tu = 6.53% - baseline results

3.2.2 Exact Profiles Based on Experimental Data

Fully developed profiles for velocity, temperature, TKE and ε are developed by writing MATLAB code and then importing the points using *Boundary Profiles* in FLUENT. MATLAB code to generate the profiles is given in Appendix-C.

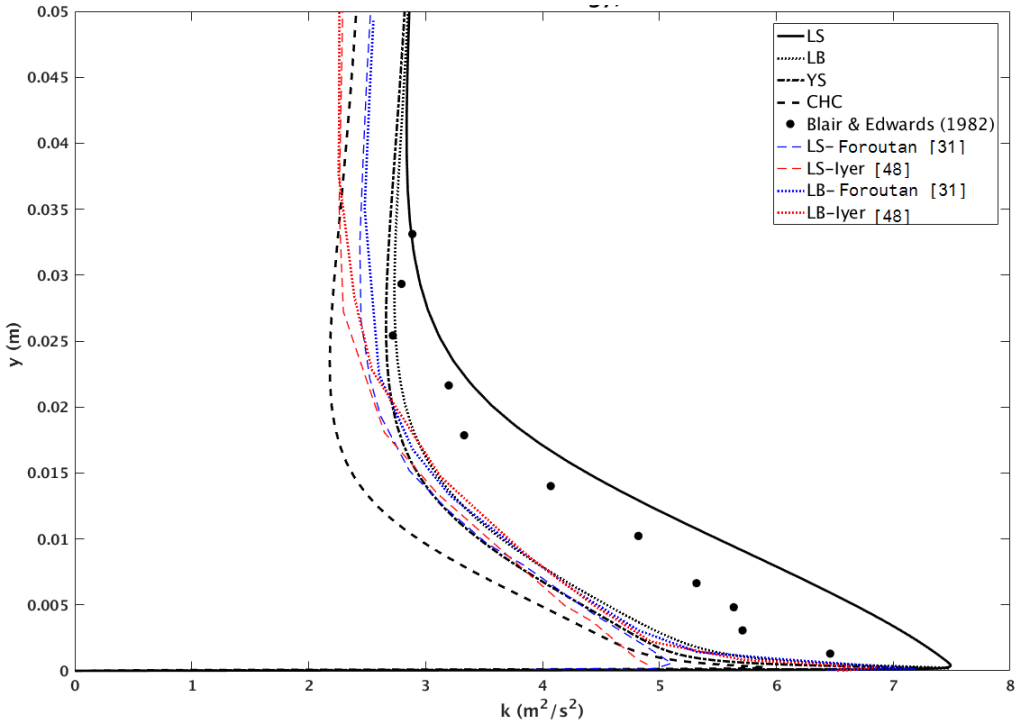
Starting profiles of TKE and ε has some effect at the lower momentum thickness based Reynolds number by increasing the skin friction coefficient. This effect dampens out as the boundary layer becomes thicker and skin friction values collapses to that of the uniform TKE and ε values. This is true for all the models. But in process the over-prediction from the data increases.

3.3 Effect of Variable Prandtl Number

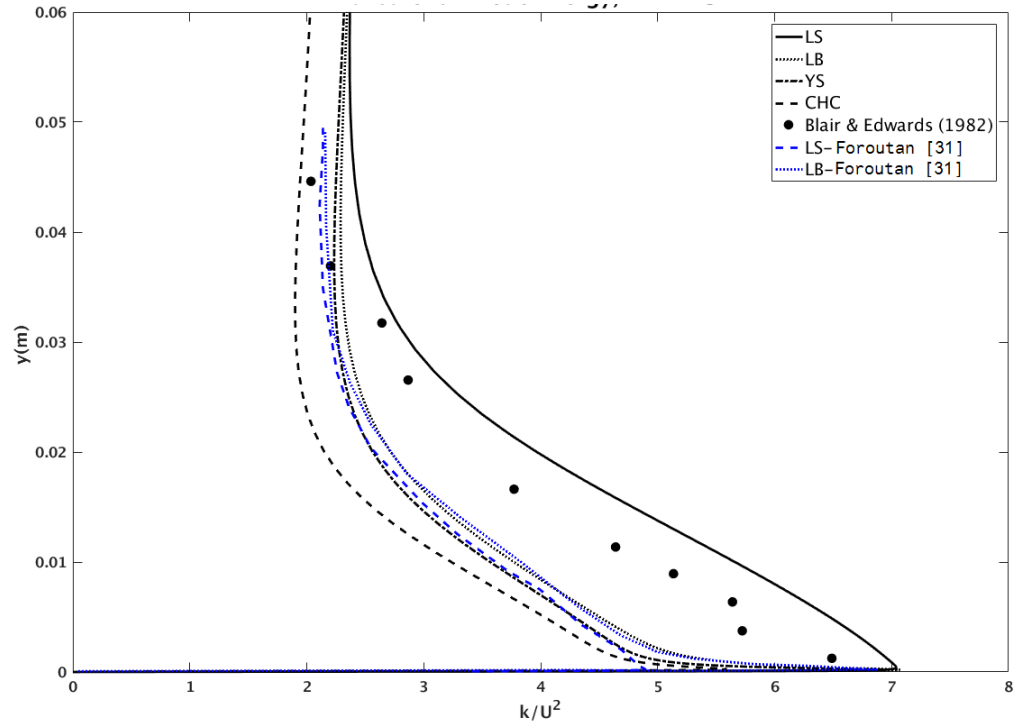
Turbulent heat flux is given by the following equation

$$\overline{u_i T'} = -\frac{\nu_t}{Pr_t}\frac{\partial T}{\partial y} \tag{3.1}$$

where Pr_t is the turbulent Prandtl number and is equal to 0.85 away from the wall. This value is suitable for most flow conditions but a more general form given by Kays and Crawford [60] can be used. No change is observed for skin friction coefficient and TKE profiles by changing Pr_t. Stanton number improved at low TU. For moderate FST, predictions were better at higher



(a) x = 1.32 m



(b) Tx = 1.73 m

Figure 3.10: Turbulent Kinetic Energy profiles over a flat plate for Tu = 6.53% at x=1.32 m and x = 1.73 m for uniform starting profiles - baseline results

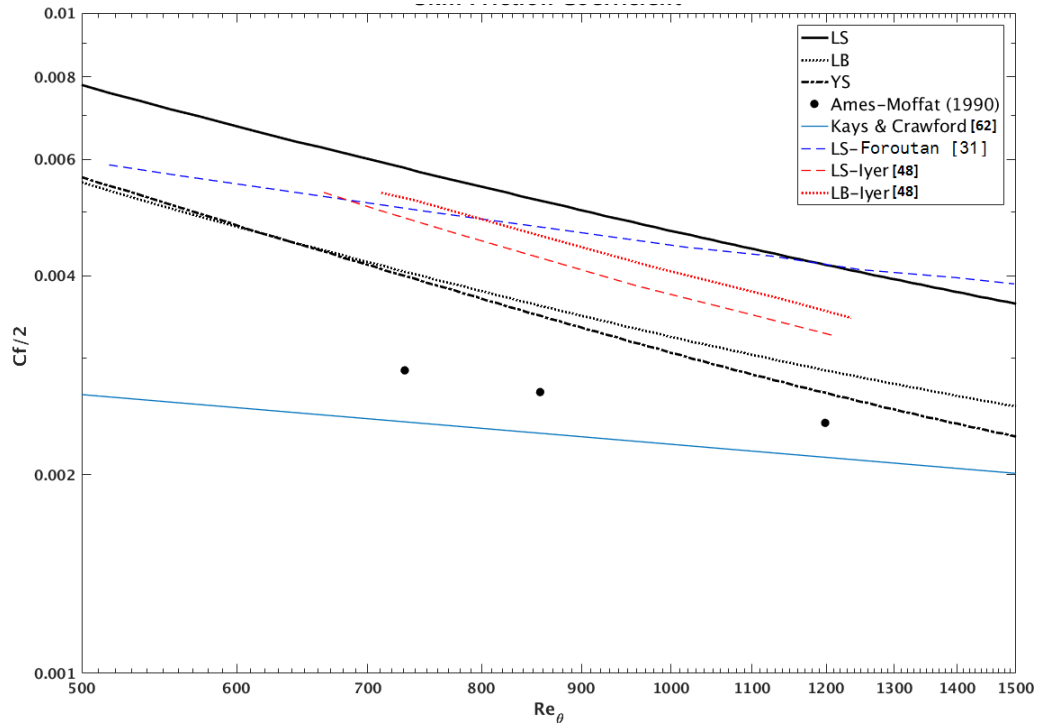


Figure 3.11: Skin Friction Coefficient over a flat plate for Tu = 25.7% - baseline results

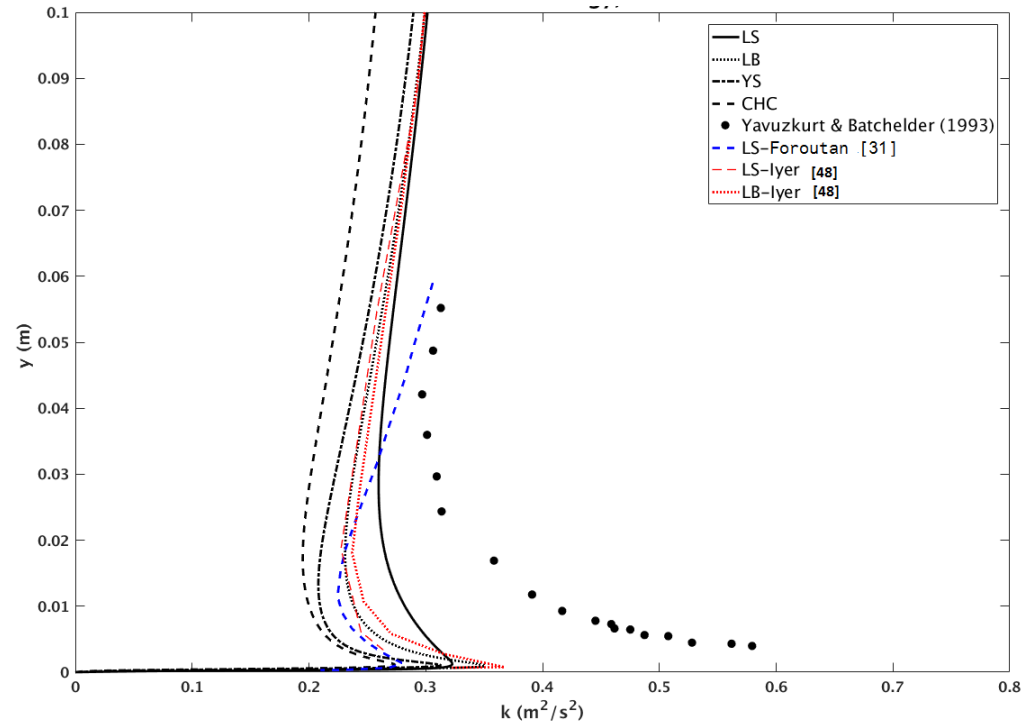
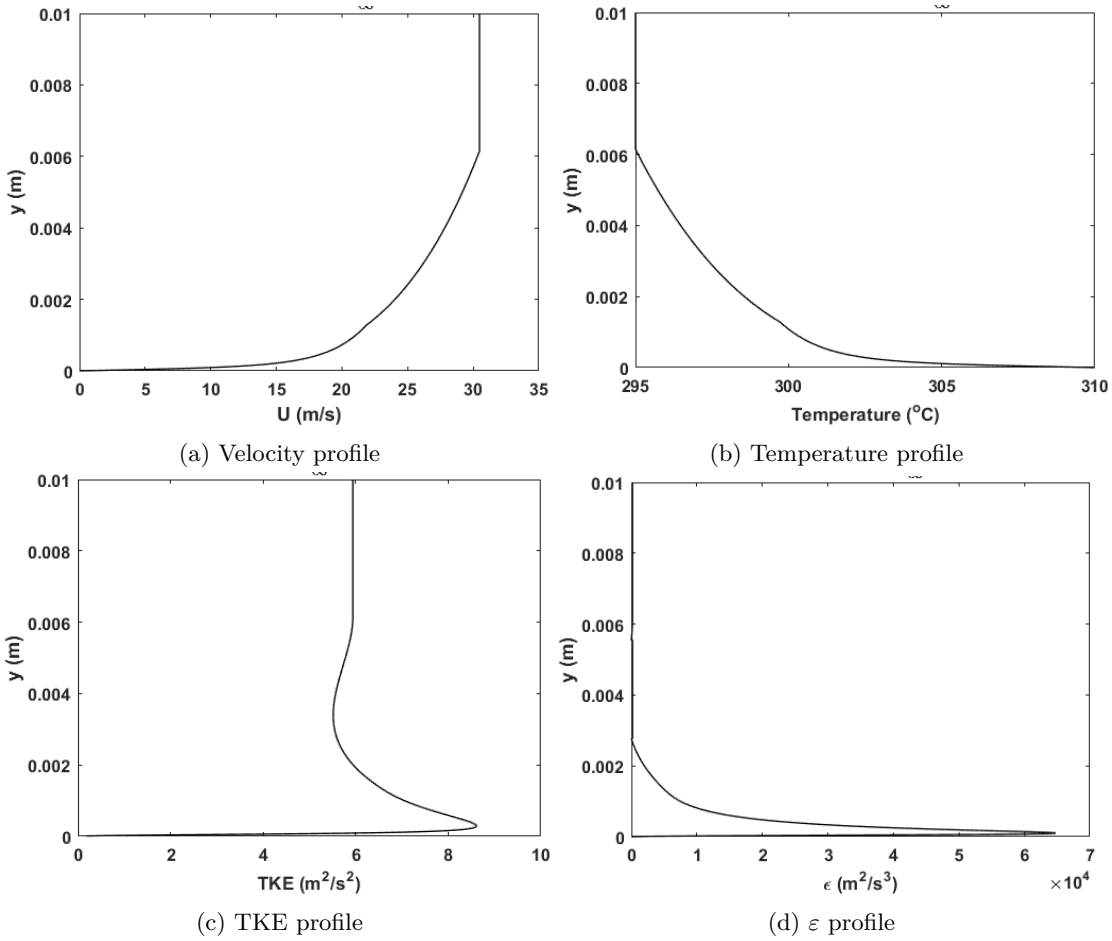


Figure 3.12: Turbulent Kinetic Energy profile over a flat plate for Tu = 25.7% at x = 2.08 m - baseline results



(a) Velocity profile

(b) Temperature profile

(c) TKE profile

(d) ε profile

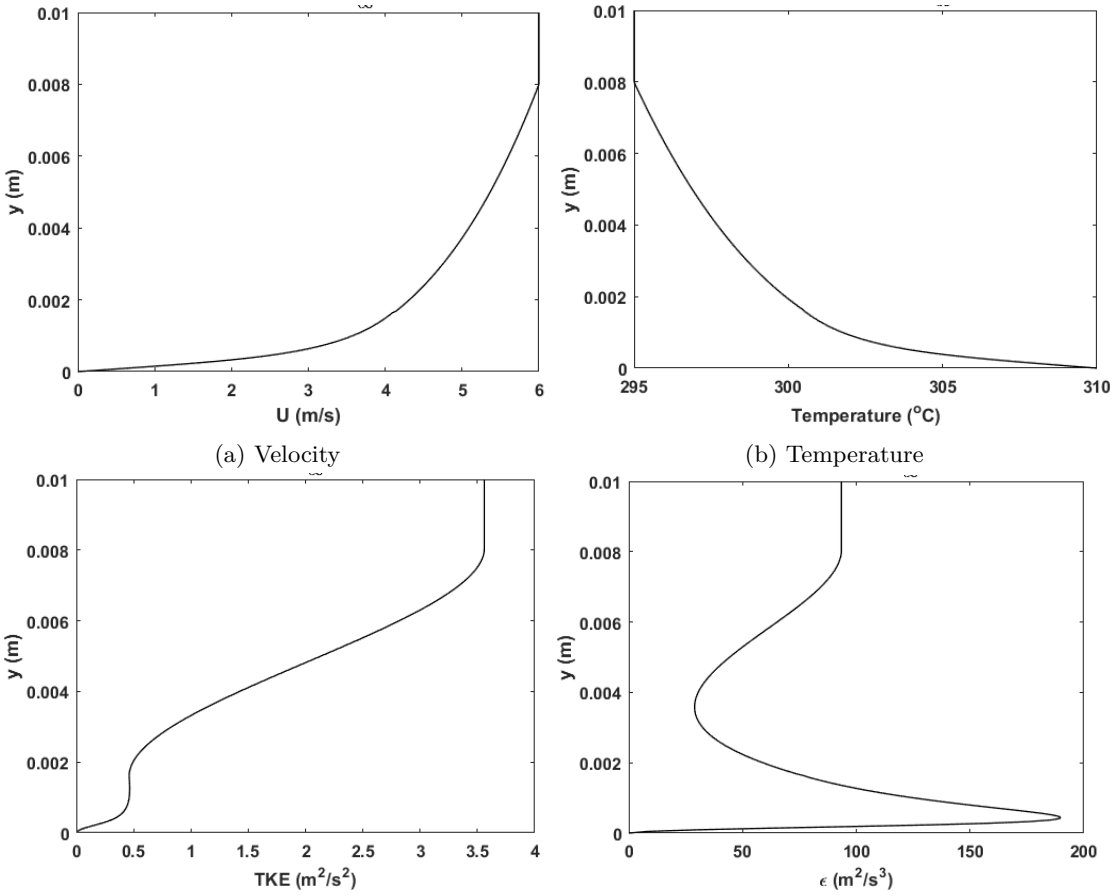
Figure 3.13: Initial fully developed turbulent boundary layer profiles for simulation over a flat plate at $Tu_\infty = 6.53$ %

Reynolds number. For variable Pr_t LB model gave worst results with error of 11 % whereas other models were in 4% of the data for Stanton number. St and Cfx were good but TKE was bad. For high TU, there is 35-100 % over-prediction and the slopes do not match. There is up to 80 % error for Stanton number but this may be due to the wrong slope. Results improve for variable Pr_t but were still poor.

$$Pr_t = \frac{1}{\frac{1}{2Pr_\infty} + CPr_t\sqrt{\frac{1}{Pr_\infty}} - (CPe_t)^2\Big[1 - exp\big(\frac{-1}{CPe_t\sqrt{Pr_{t\infty}}}\big)\Big]} \tag{3.2}$$

Here, $Pr_{t\infty}$ and C are experimentally determined values equal to 0.85 and 0.2 respectively. Pe_t is the Peclet number given by

$$Pe_t = -\frac{\nu_t}{\nu}Pr \tag{3.3}$$



(a) Velocity (b) Temperature

(c) TKE (d) ε

Figure 3.14: Initial fully developed turbulent boundary layer profiles for simulation over a flat plate at $Tu_\infty = 25.7$ %

3.4 Launder-Sharma Model in Fluent

It is observed that the predictions from LS model in Fluent consistently differs from other LRN models and also from the previous studies using LS model. Mathur and He [32] modification is applied to Fluent. They did not change the energy equation, so that must be changed to get correct results as was in for momentum equation. Energy equation is implemented using UDF in the same manner as momentum transport equation is written.

It is clearly observed from the results obtained in Figures 3.24 - 3.31 that the modified implementation of the Launder-Sharma model predicts skin friction coefficient, Stanton number and TKE profiles similar to other LRN models. These predictions are very different from Fluent's Launder-Sharma model. Thus it can be concluded that the Fluent's LS model cannot be used for further study and there is something wrong in that implementation.

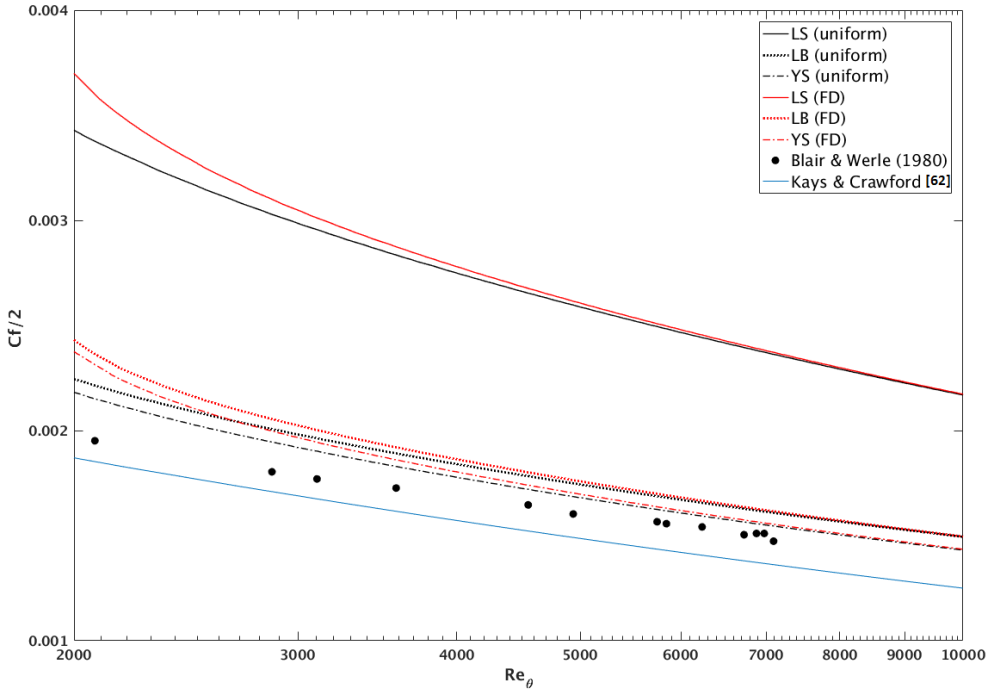


Figure 3.15: Skin Friction Coefficient over a flat plate at Tu = 6.53% for fully developed initial turbulent boundary layer profiles - baseline results

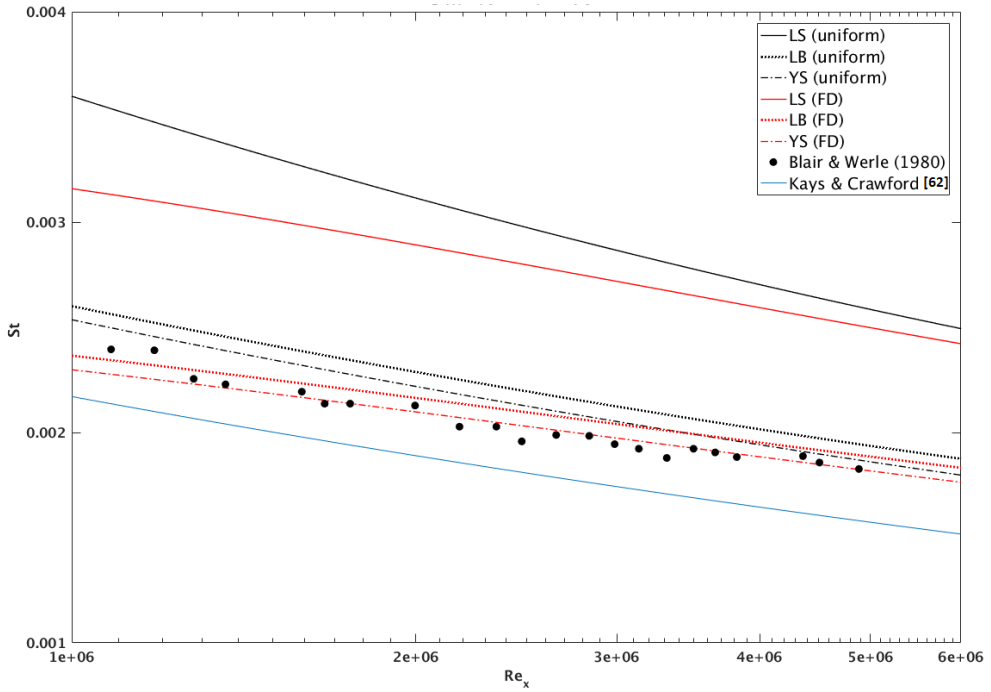
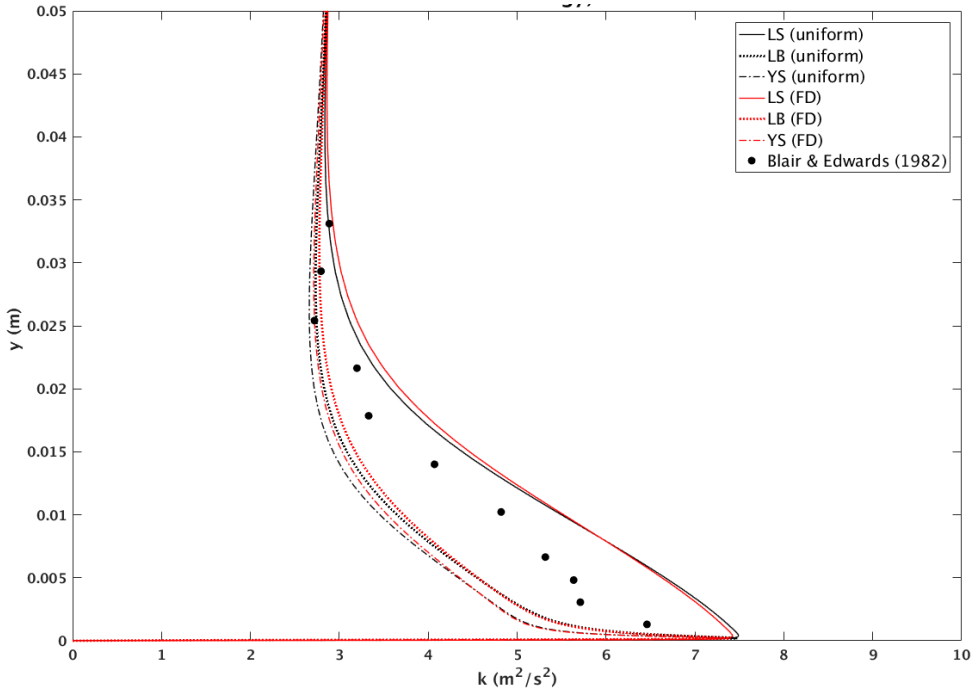
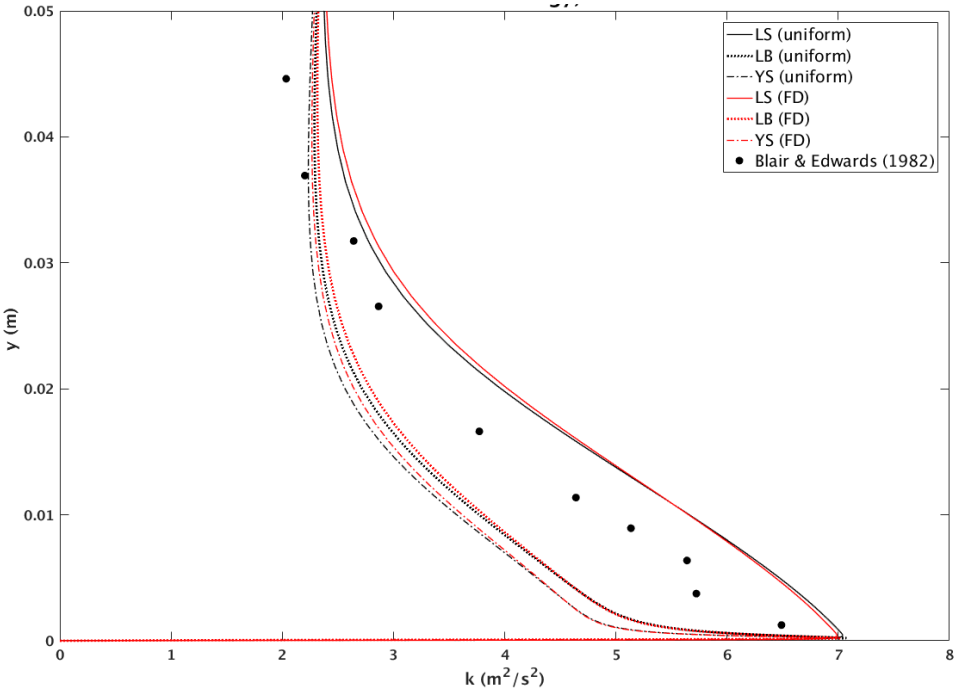


Figure 3.16: Stanton number over a flat plate at Tu = 6.53% for initial turbulent boundary layer profiles - baseline results



(a) x = 1.32 m



(b) Tx = 1.73 m

Figure 3.17: Turbulent Kinetic Energy profiles over a flat plate for Tu = 6.53% at x=1.32 m and x = 1.73 m for initial turbulent boundary layer profiles - baseline results

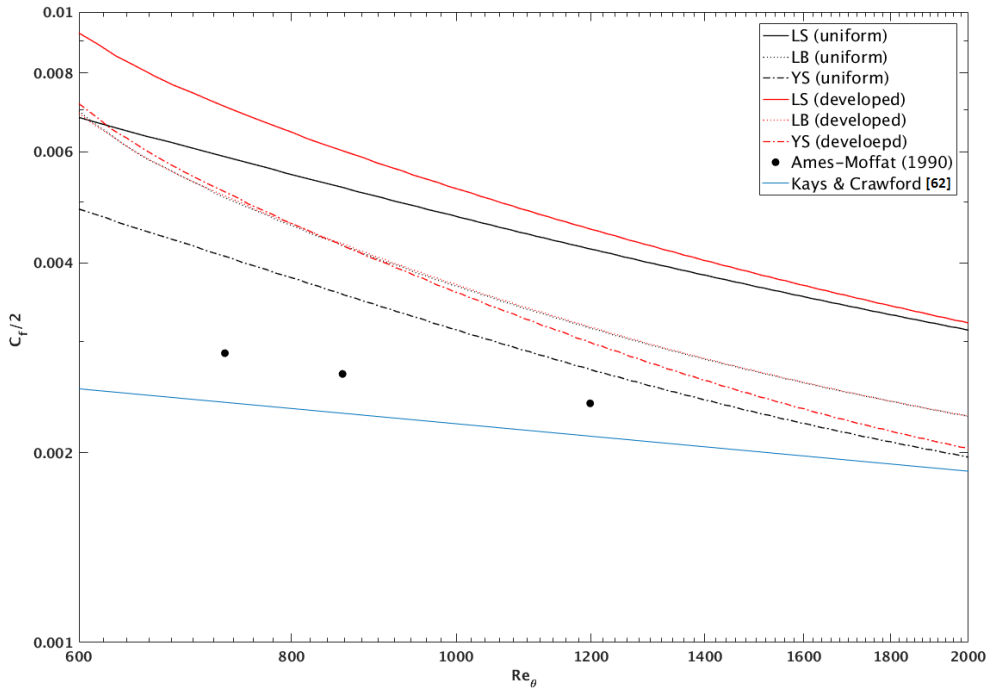


Figure 3.18: Skin Friction Coefficient over a flat plate for Tu = 25.7% for initial turbulent boundary layer profiles - baseline results

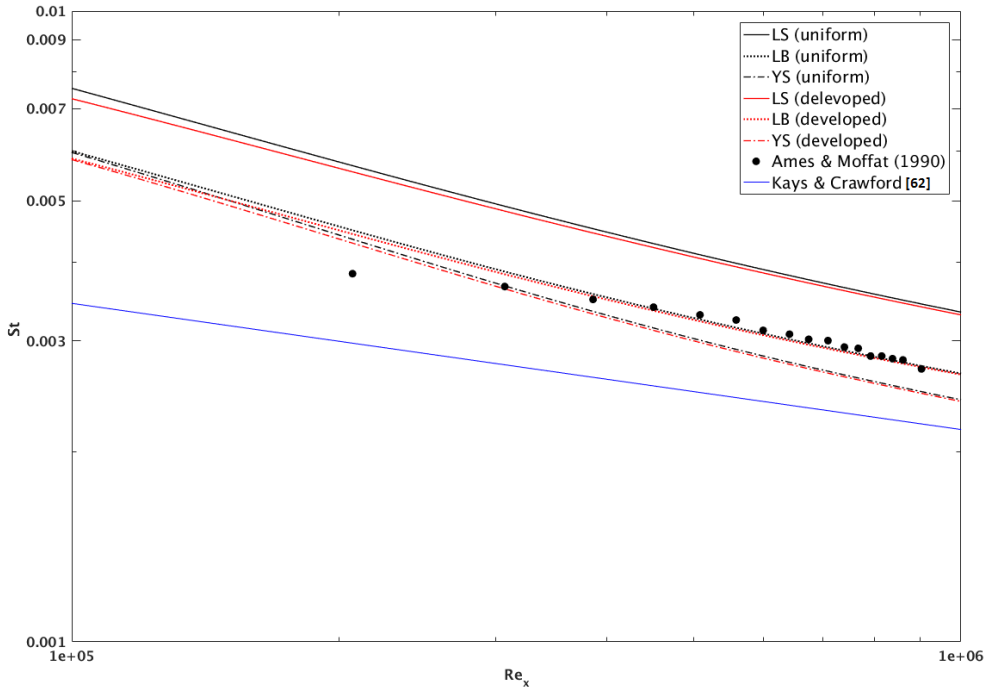


Figure 3.19: Stanton Number over a flat plate for Tu = 25.7% for initial turbulent boundary layer profiles - baseline results

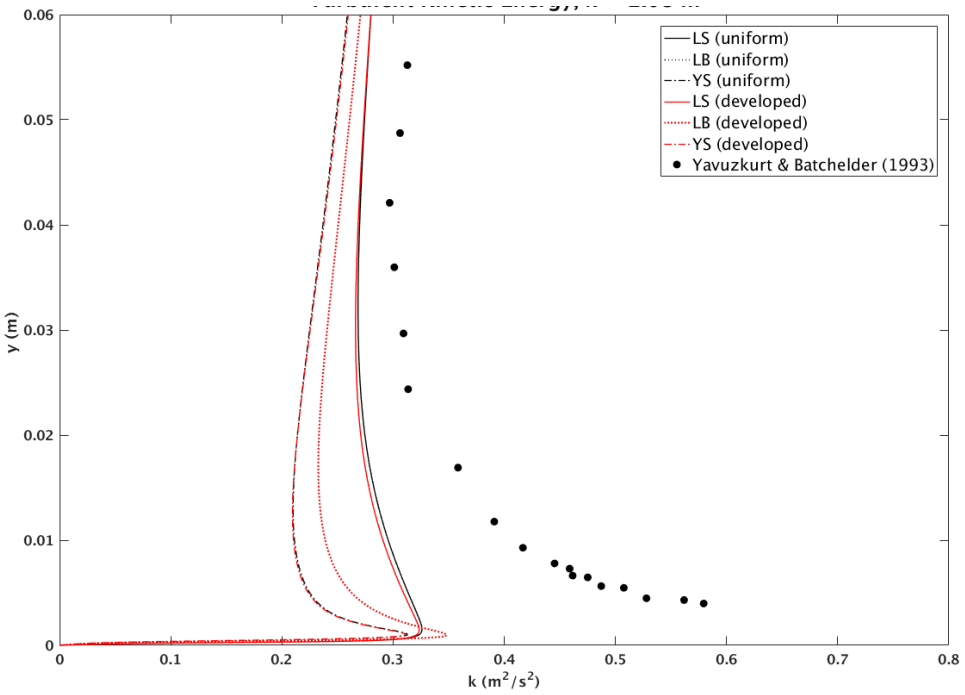


Figure 3.20: Turbulent Kinetic Energy profiles over a flat plate for Tu = 25.7% at x = 2.08 m for initial turbulent boundary layer profiles - baseline results

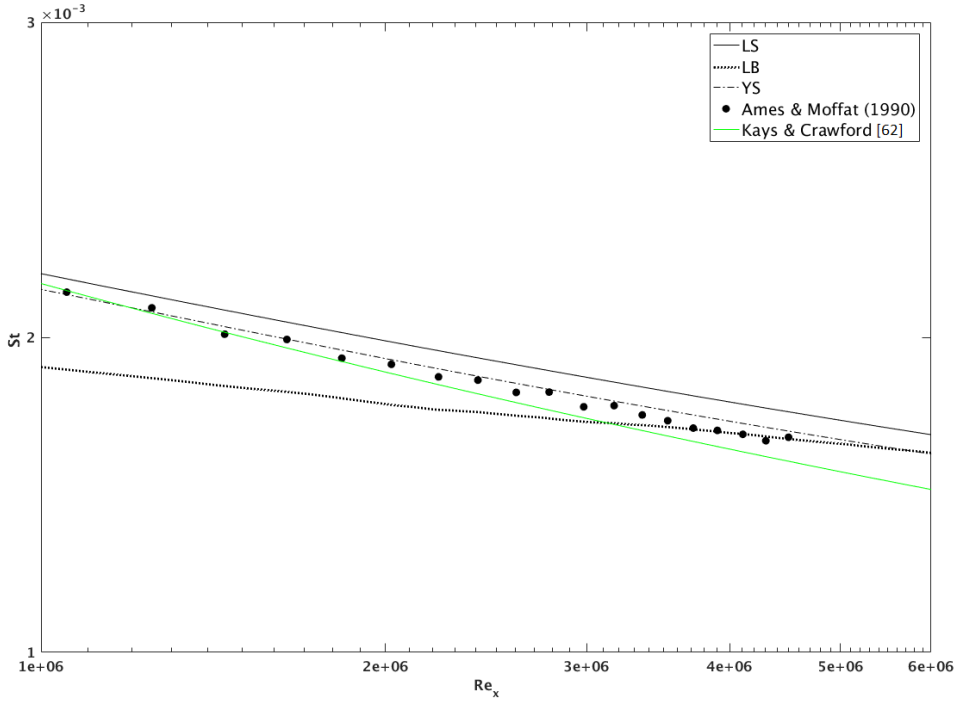


Figure 3.21: Stanton Number over a flat plate at Tu = 1% for variable Prandtl number - baseline results

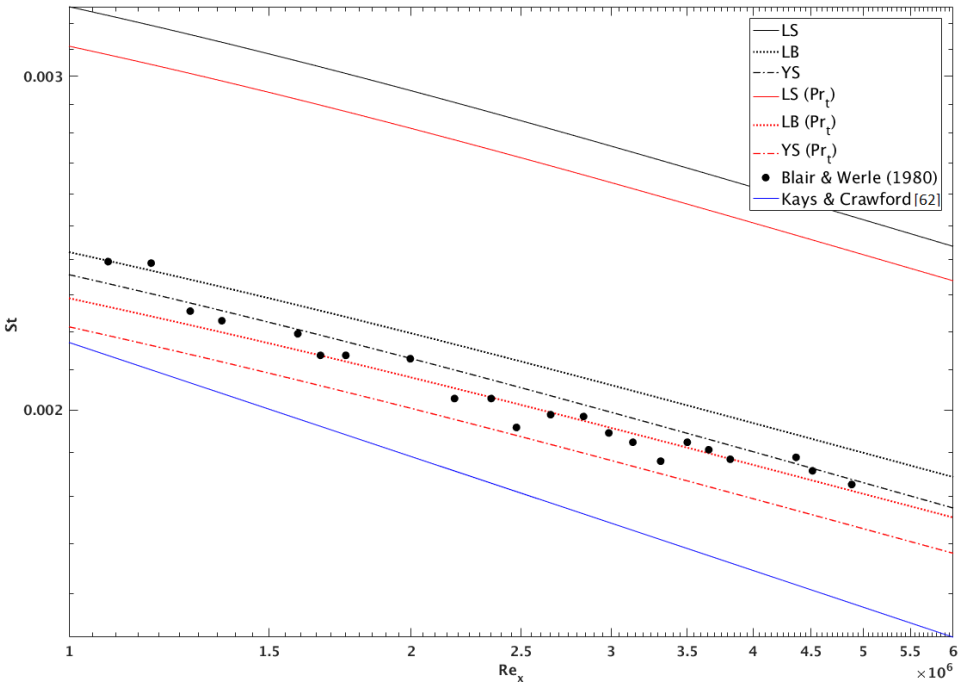


Figure 3.22: Stanton Number over a flat plate at Tu = 6.53% for variable Prandtl number - baseline results

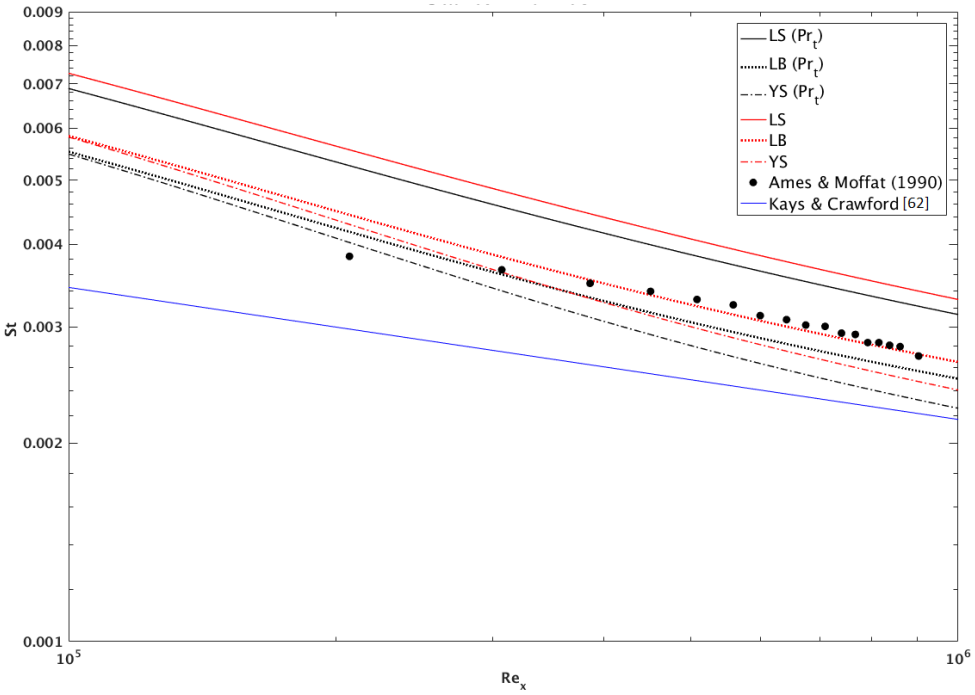


Figure 3.23: Stanton Number over a flat plate at Tu = 25.7% for variable Prandtl number - baseline results

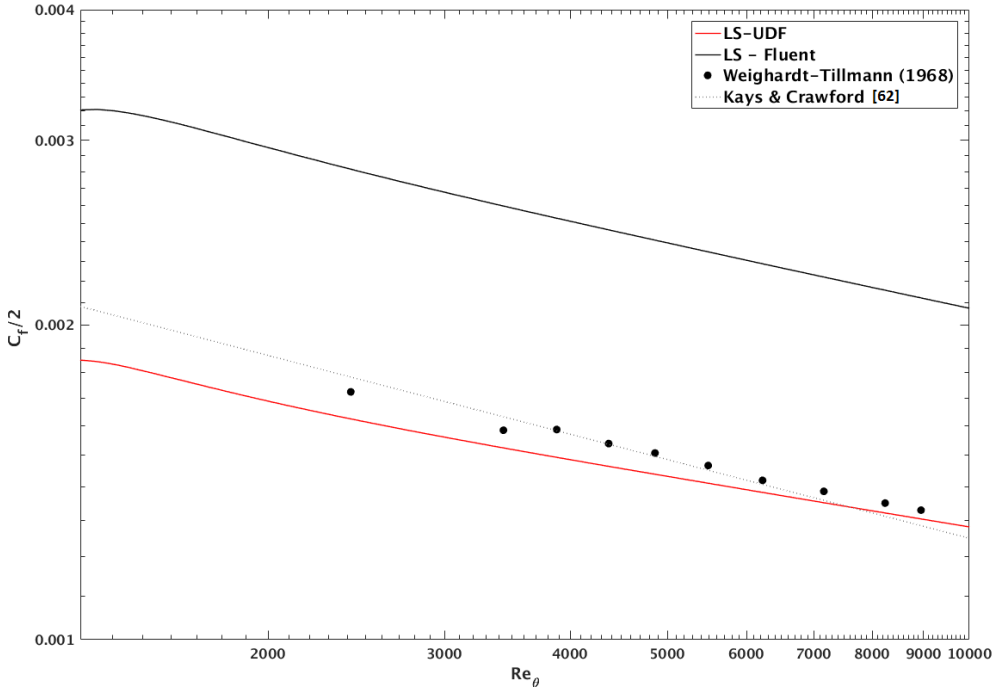


Figure 3.24: Skin Friction Coefficient over a flat plate at Tu = 1% for UDF implemented LS model - baseline results

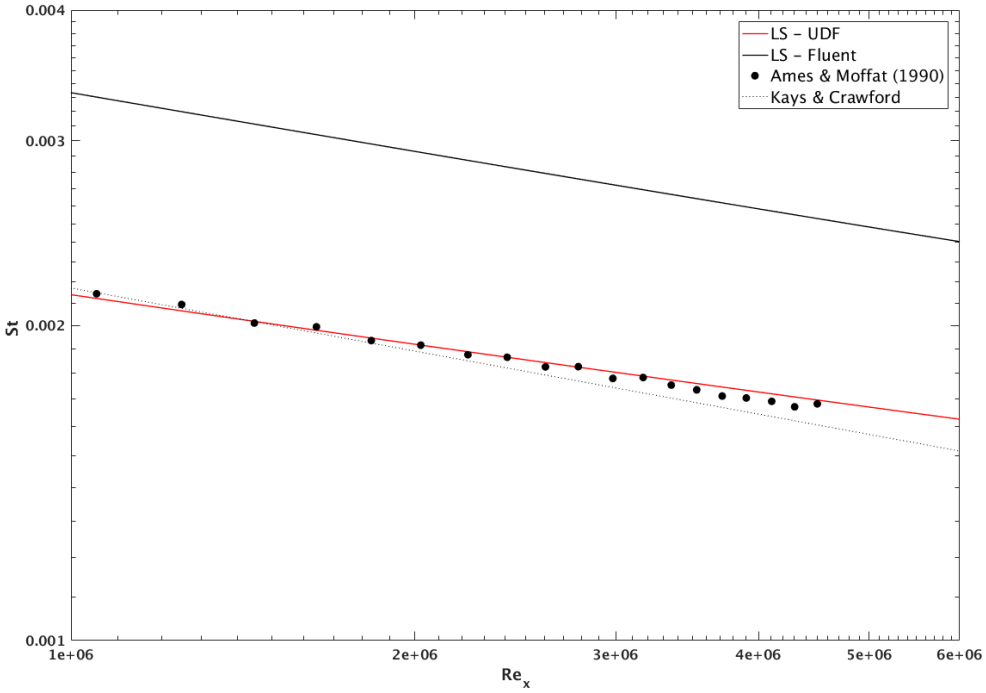


Figure 3.25: Stanton Number over a flat plate at Tu = 1% for UDF implemented LS model - baseline results

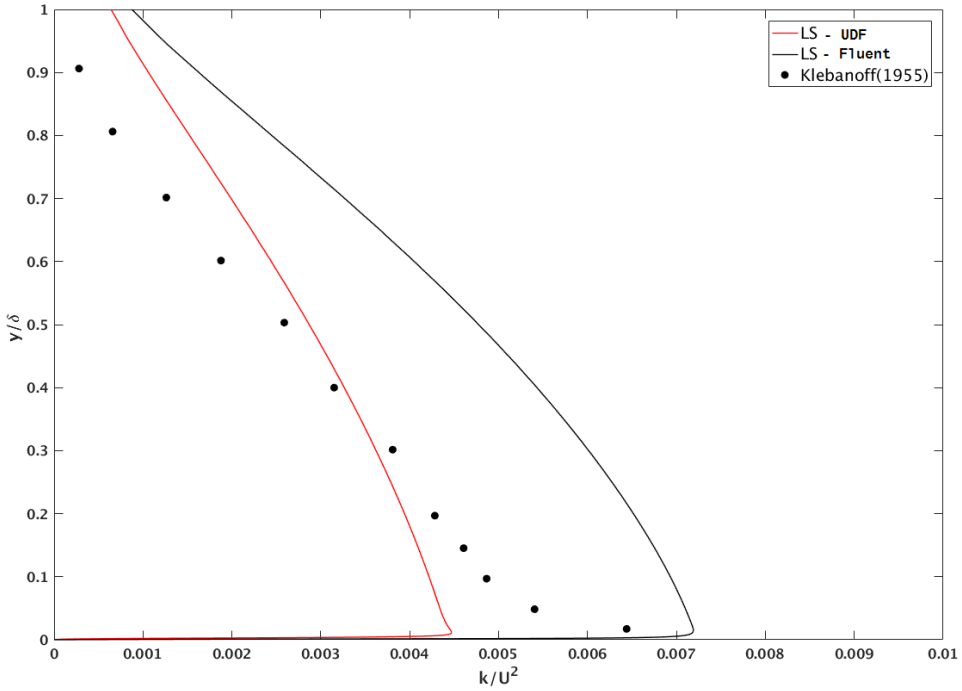


Figure 3.26: Turbulent Kinetic Energy profile over a flat plate for Tu = 1% at $Re_\theta = 7700$ for UDF implemented LS model - baseline results

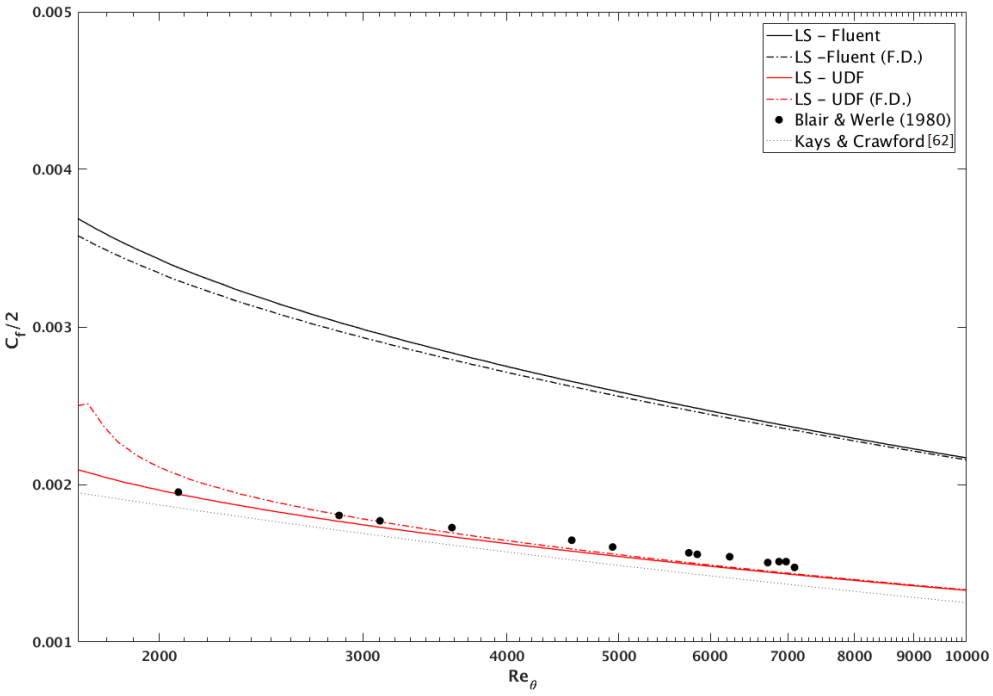


Figure 3.27: Skin Friction Coefficient over a flat plate at Tu = 6.53% for UDF implemented LS model - baseline results

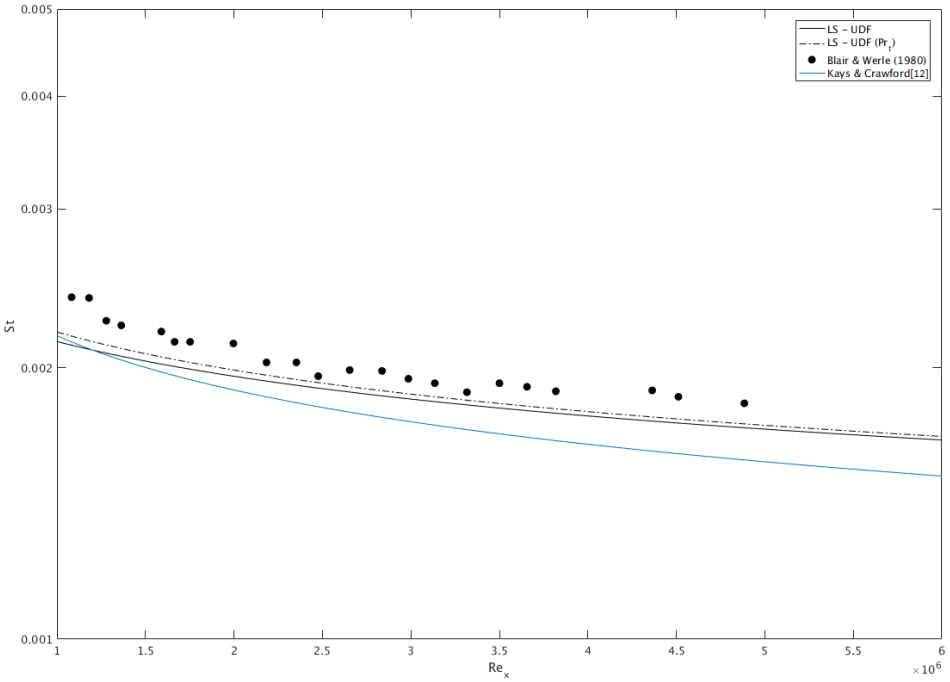


Figure 3.28: Stanton Number over a flat plate at Tu = 6.53% for UDF implemented LS model - baseline results

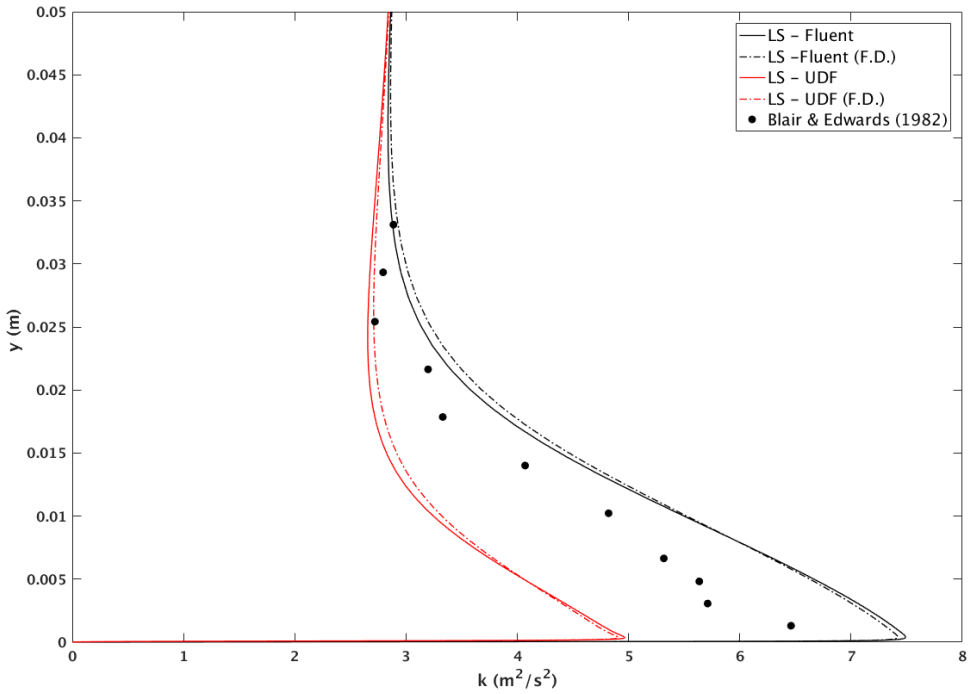


Figure 3.29: Turbulent Kinetic Energy profile over a flat plate at Tu = 6.53% at x = 1.32 m for UDF implemented LS model - baseline results

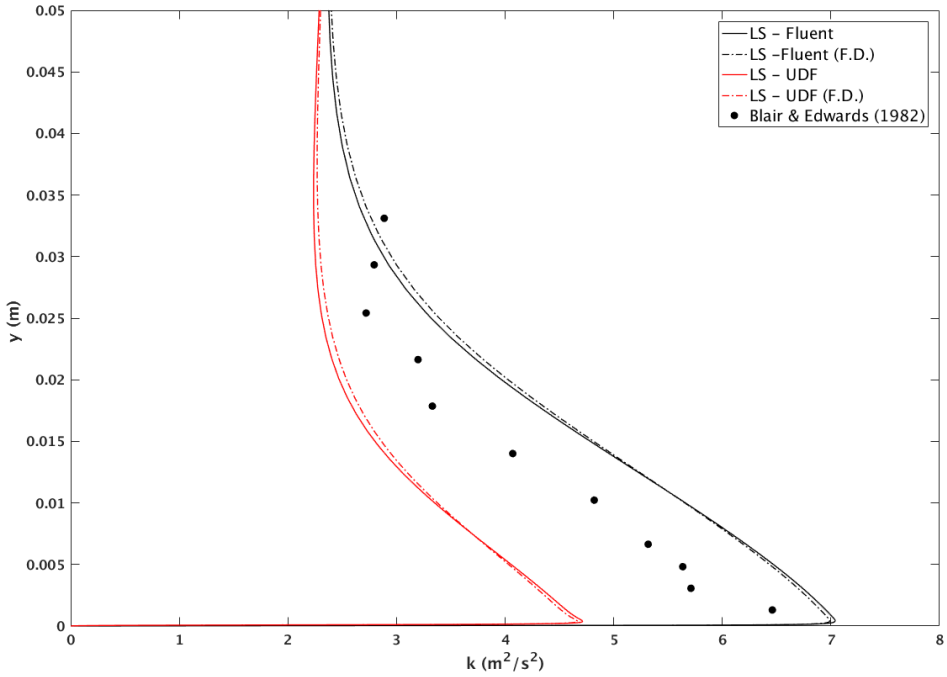


Figure 3.30: Turbulent Kinetic Energy over a flat plate at Tu = 6.53% at x = 1.73 m for UDF implemented LS model - baseline results

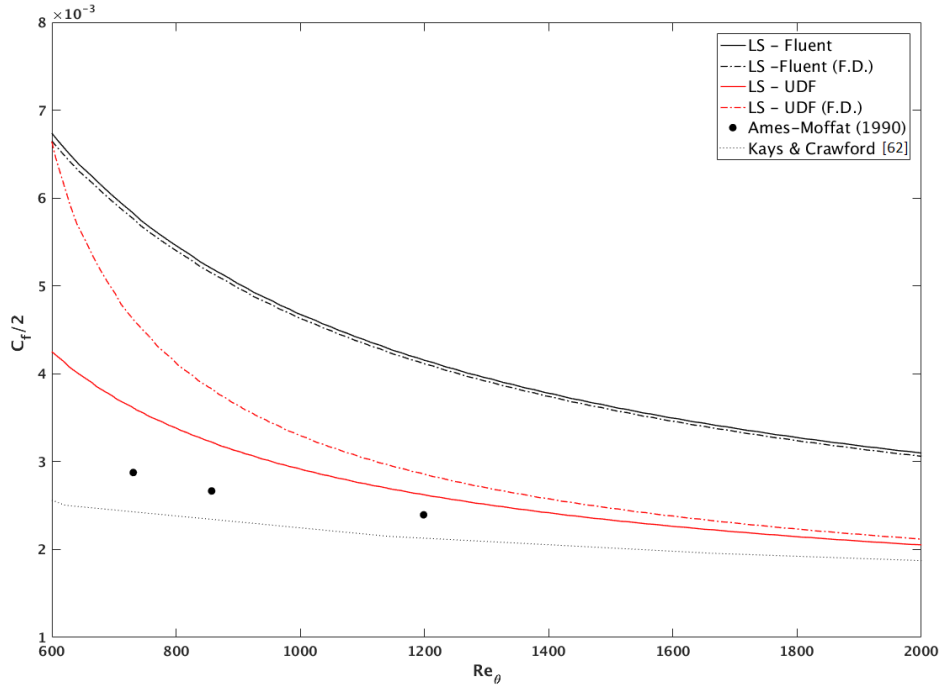


Figure 3.31: Skin Friction Coefficient over a flat plate at Tu = 25.7% for UDF implemented LS model - baseline results

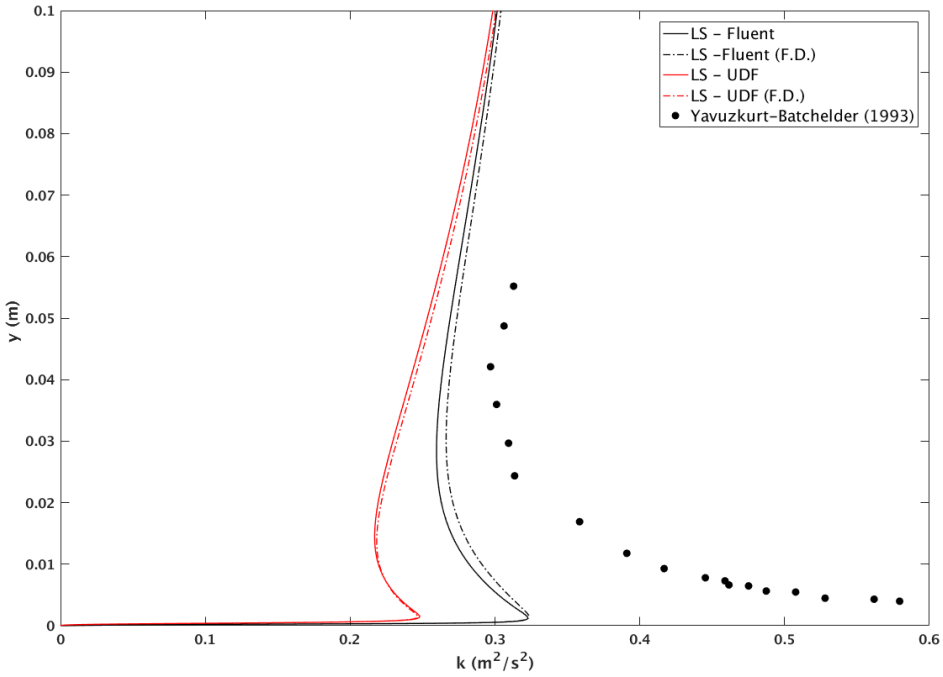


Figure 3.32: Turbulent Kinetic Energy profile over a flat plate for Tu = 25.7% at x = 2.08 m for UDF implemented LS model - baseline results

3.5 New High FST TKE Diffusion Model Implementation in Fluent

In the first analysis to implement the model in FLUENT only c_μ is changed as a function of FST intensity [48].

Table 3.2: User Defined Scalars and User Defined Memory for UDF in Fluent

textit	Variable name	Definition
UDS 0	KFST	$C_{FST}\mu_t\frac{\partial U}{\partial y}$
UDM 0	TKE	Free-stream Turbulent kinetic energy
UDM 1	TDR	Free-stream Dissipation rate
UDM 2	RK	$\sqrt{(k)}$
UDM 3	RET	$\frac{\rho k^2}{\mu\varepsilon}$
UDM 4	REY	$\frac{\rho\sqrt{k}y}{\mu}$
UDM 5	CFST	C_{FST}
UDM 6	CMU	c_μ
UDM 7	MUT	turbulent viscosity μ_t
UDM 8	L	Free-stream length scale

The additional term in the TKE equation given in Equation 3.4 accounts for the interaction between Reynolds stress and mean flow. Here the first term represents additional production

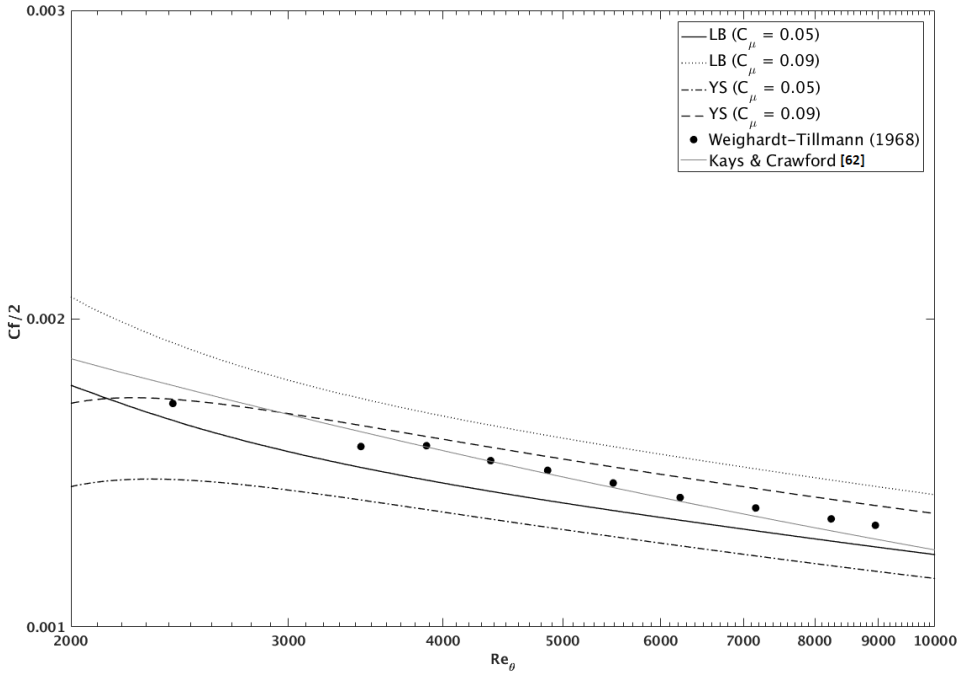


Figure 3.33: Skin friction coefficient over a flat plate for constant c_μ comparison at low Tu_i

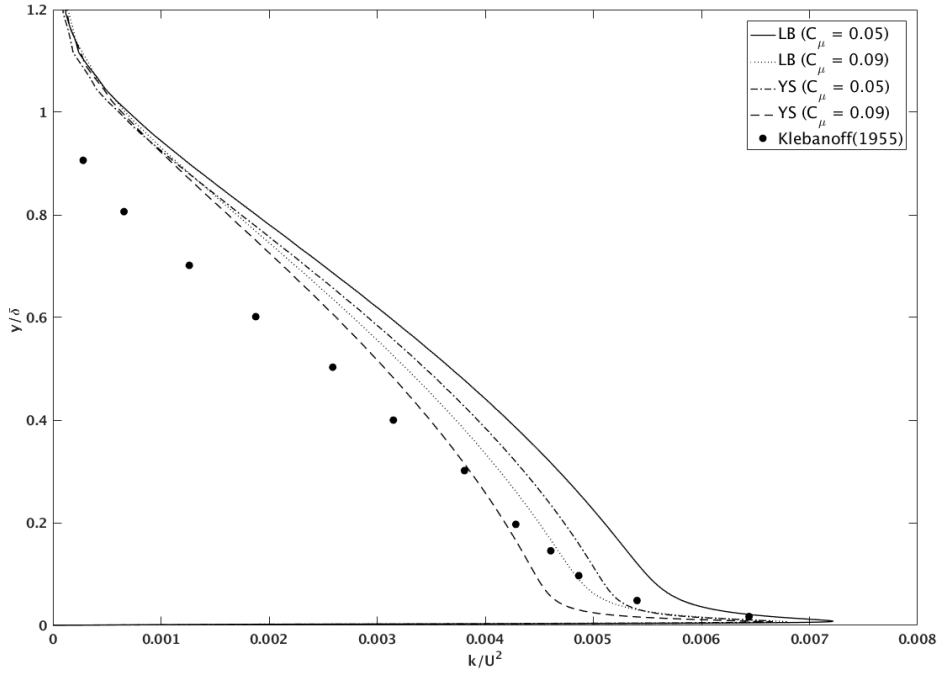


Figure 3.34: Turbulent Kinetic Energy profile over a flat plate for constant c_μ comparison at low Tu_i

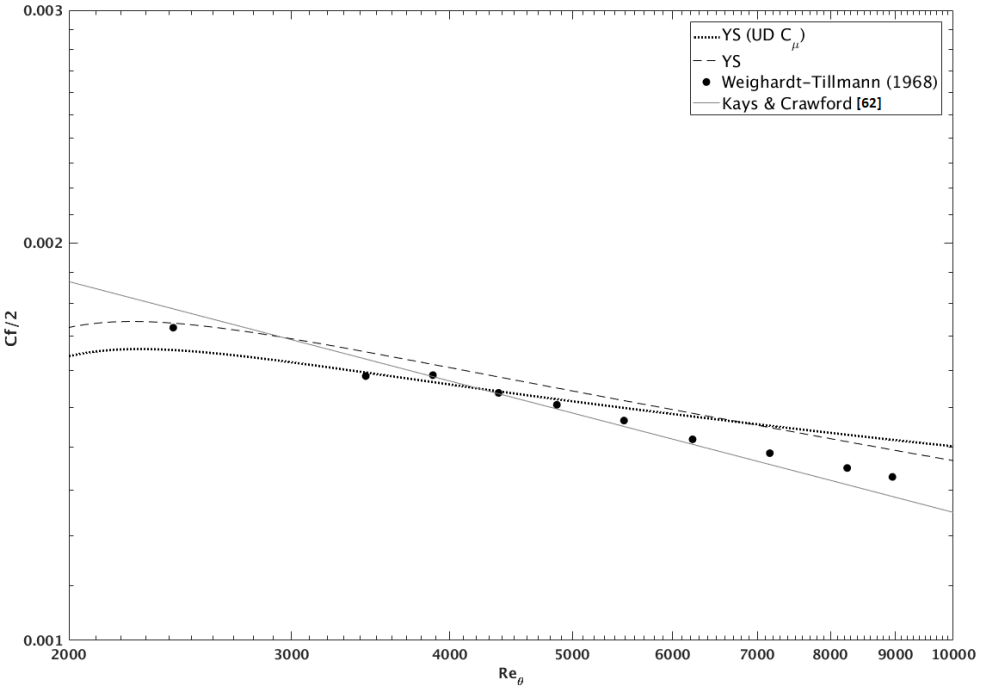


Figure 3.35: Skin friction coefficient over a flat platefor constant c_μ comparison from UDF

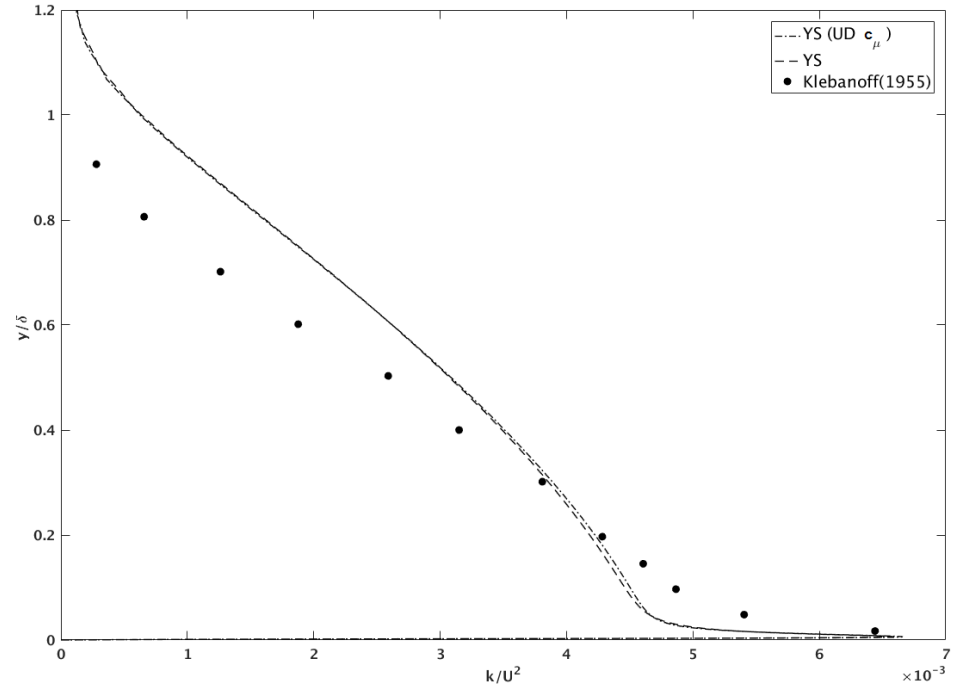


Figure 3.36: Turbulent Kinetic Energy profile over a flat plate for constant c_μ comparison from UDF

due to FST and the second term represents the increase in TKE due to diffusion [31].

$$D = C_{FST}\mu_t\Big(\frac{\partial U}{\partial y}\Big)^2 + U\frac{\partial}{\partial y}\Big(C_{FST}\mu_t\frac{\partial U}{\partial y}\Big) \tag{3.4}$$

Turbulent viscosity in Equation 3.5 is function of k and ε

$$\mu_t = \frac{\rho f_\mu c_\mu k^2}{\varepsilon} \tag{3.5}$$

So, the derivative in Equation 3.4 with respect to k can be written as shown in 3.6

$$\frac{\partial D}{\partial k} = \frac{2C_{FST}\mu_t}{k}\Big(\frac{\partial U}{\partial y}\Big)^2 + 2U\frac{\partial}{\partial y}\Big(\frac{C_{FST}\mu_t}{k}\frac{\partial U}{\partial y}\Big) \tag{3.6}$$

After several trials diffusion term is added in a step by step manner by first varying only c_μ thereafter including this effect into turbulent viscosity, and finally addition of diffusion term.

Simulations were run at low FST to study the effect of c_μ considering two constant value of 0.09 (original) and 0.05. Figure prove that the c_μ has the effect of lowering the skin friction coefficient and Stanton number which is as expected from theory. The next step should be to see the UDF implementation of c_μ for these two models considering only the constant values which can be expanded to include the effect of variable c_μ based on FST.

The only way by which the effect of c_μ can be implemented in Fluent is by changing the turbulent viscosity to reflect this change. Turbulent viscosity for LRN models has a damping function, f_μ which depends on two Reynolds number - based on wall distance, turbulent kinetic energy and dissipation rate.

$$Re_y = \frac{\rho\sqrt{k}y}{\mu} \tag{3.7}$$

$$Re_t = \frac{\rho k^2}{\mu\varepsilon} \tag{3.8}$$

For LB model,

$$f_\mu = \Big(1 - exp(-0.0165Re_y)\Big)^2\Big(1 + \frac{20.5}{Re_T}\Big) \tag{3.9}$$

UDF for $c_\mu = 0.09$ gives same results as compared to the inbuilt functions. But when the c_μ value is changed to 0.05, inbuilt model gives under-prediction which is expected. But the UDF deviates unexpectedly and mostly over-predicts. It was decided to test the variation of c_μ on the model. To perform this task - c_μ was changed from the GUI panel in FLUENT and by defining new eddy viscosity using user defined functions. Reducing c_μ from GUI showed decrease in skin friction as is evident from figure 3.37. In the same figure, the effect of variation of the constant using $DEFINE\_TURBULENT\_VISCOSITY()$ is seen with the results making no sense. This created possibility of mistakes either in the UDF or in the way FLUENT interprets the macros. It was assumed that the diffusivity of the transport equations for k and ε did not

change by changing on the eddy viscosity. So, the complete transport equations for k and ε were written in C language and hooked to the FLUENT solver. Inbuilt turbulence equations were turned off and the new transport equations were solved. It was also important to correctly give the wall boundary conditions for both k and ε. Value of k as zero was given at wall and and value of ε was given.

$$\varepsilon = 2\nu \frac{k_1}{\Delta y^2} \tag{3.10}$$

It was observed in the same figure that the default value of $c\_mu = 0.09$ gave result close to the inbuilt model but lower value varied completely. Though this variation respected the TKE profiles but this raised doubts over the implementation. The error may be that the diffusivity values are not being changed by changing the turbulent viscosity. To produce that effect it was decided to change the value of σ_k and σ_ε that will take into account the value of c_μ.

In Figure 3.39, total of six cases were simulated corresponding to two viscosity implementation - Inbuilt and User Defined, and three implementation of σ_i - FLUENT values from GUI panel, values of same constants as FLUENT using macros and values of constants used in the literature using macros. Macros used here are given in Appendix-A. It is observed that the effect of diffusivity values given from GUI panel or from UDFs is the same. Modifying the constants to match that of the paper gives slightly higher value of shear stress as compared to inbuilt FLUENT constants. But the effect of viscosity implementation is noticeable. It can be inferred that the Prandtl number (diffusivity constants) change from GUI panel and from UDF has the same effect. So, the problem lies in the correct definition of μ_t. The only variable that can be different in the FLUENT implementation and the literature would be the damping function f_μ. Thus the focus now shifts to identify the correct damping functions. On further trials and flooring the values of k and ε it was found that the Fluent has the same implementation of the damping function as is given in the literature. It was observed that the only change that is observed is the one due to the change of model constants of diffusivity.

Since the turbulent viscosity of the UDF matches that of FLUENT , we move on to the next step - to study effect of the value of c_μ. Only two values of c_μ are considered, 0.09 (default) and 0.05. Corresponding to these values the model constants and viscosity is modified using the GUI panel and UDF for both Inbuilt FLUENT values and AKN values. It can be observed from Figure 3.38 that for a particular value of c_μ both UDF and GUI implementation matches. Changing c_μ to 0.05 reduces skin friction coefficient with the other trends remaining the same. This change is exactly what would be expected by reducing c_μ at low FST. Thus the UDF implementation of turbulent viscosity along with σ_k and σ_ε is validated.

A study to check if c_μ in GUI panel is suppressed by UDF or it takes precedence was conducted. The model constant values from AKN paper [61] is used and total of 6 cases are run-

- GUI $c_\mu = 0.05$
- GUI $c_\mu = 0.09$

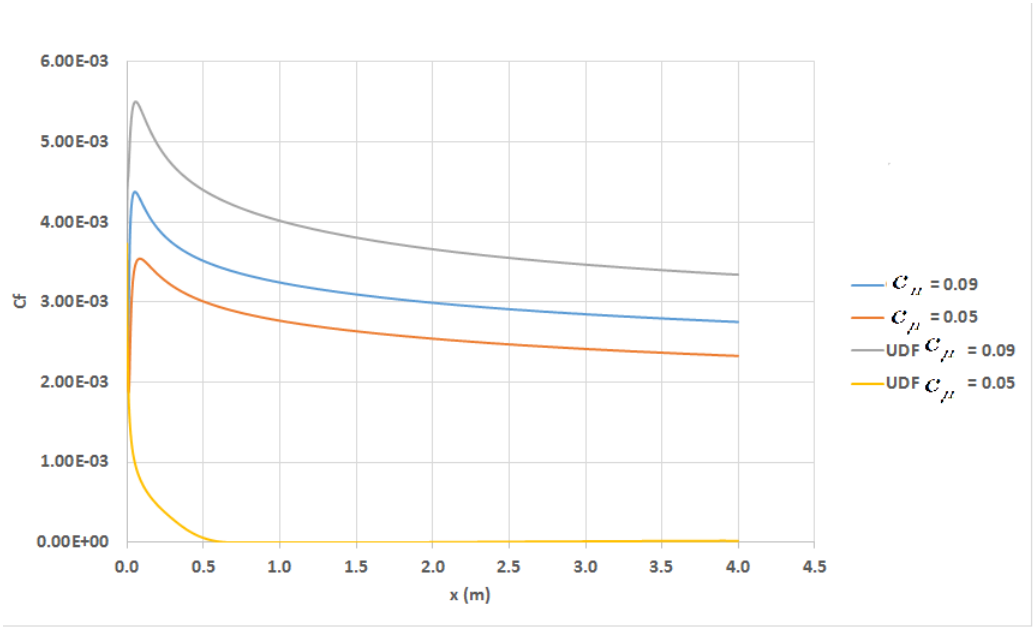


Figure 3.37: Skin friction over a flat plate for AKN model using macro for turbulent viscosity in UDF

- GUI $c_\mu = 0.05$ and UDF $c_\mu = 0.05$
- GUI $c_\mu = 0.09$ and UDF $c_\mu = 0.09$
- GUI $c_\mu = 0.05$ and UDF $c_\mu = 0.09$
- GUI $c_\mu = 0.09$ and UDF $c_\mu = 0.05$

It is also observed in the same study that UDF value exactly matches the inbuilt model when c_μ is defined same both using GUI panel and using turbulent viscosity macro. $c_\mu = 0.09$ gives results very close to Kays and Crawford [60] empirical correlation whereas $c_\mu = 0.05$ predicts lower shear stress which is as expected. When GUI c_μ is lower than UDF c_μ (due to viscosity), this means that turbulent production term becomes larger than the transport term, and vice versa. Thus the former gives very high value of shear stress than the latter case. It can be seen that value of c_μ from GUI is not suppressed while modifying μ_t from UDF. This means that the diffusivity remains independent from the change in μ_t.
A study to see the change in c_μ by modifying Schmidt number or σ_k and σ_ε proved inconclusive.

$$\sigma_{modified} = \sigma_{actual} \frac{C_{\mu,actual}}{C_{\mu,new}} \tag{3.11}$$

Figures 3.40 - 3.42 show that variable c_μ can be implemented in Fluent only by over-writing the existing equations for velocity, temperature, k and ε using UDFs. This also creates stability issues and since the source code is not available it is better to continue the work on a different

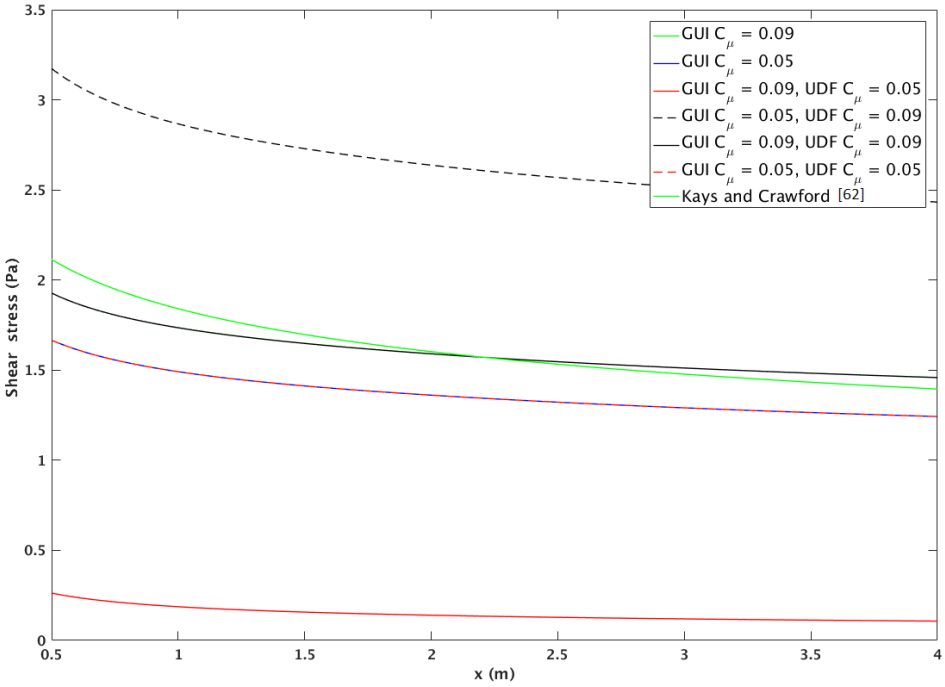


Figure 3.38: Effect of c_μ variation from GUI panel and User Defined viscosity for flow over a flat plate

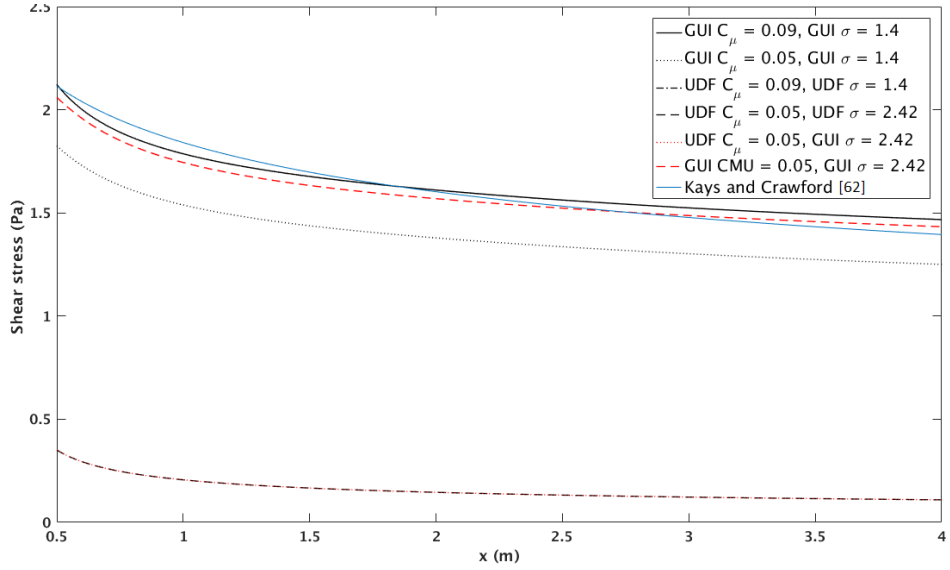


Figure 3.39: Effect of c_μ and σ variation from GUI panel and UDF for flow over a flat plate

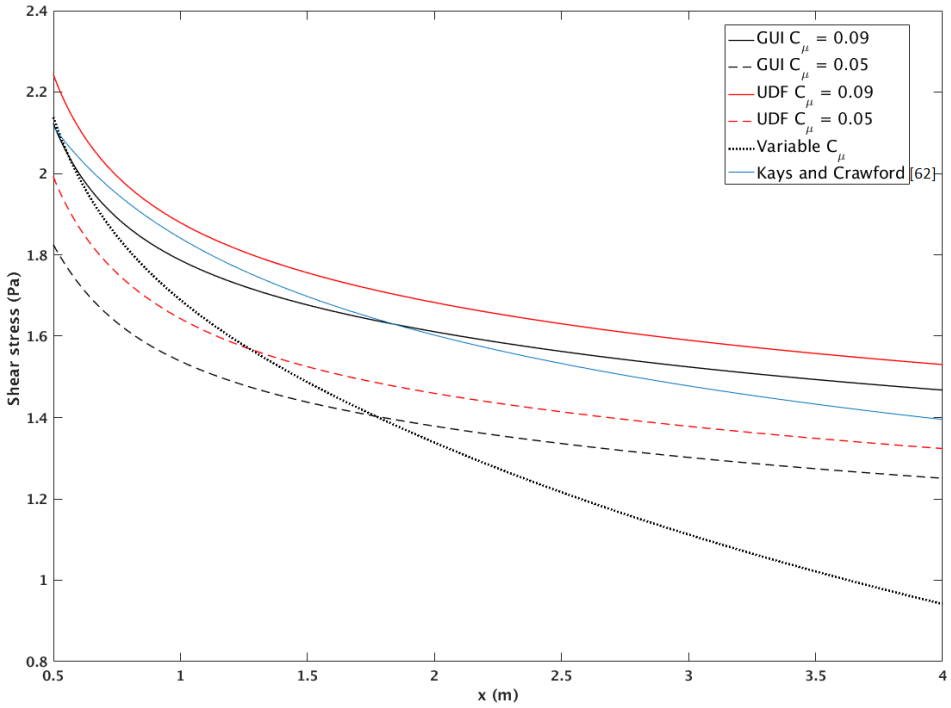


Figure 3.40: Comparison on full model implementation with variable c_μ for flow over a flat plate

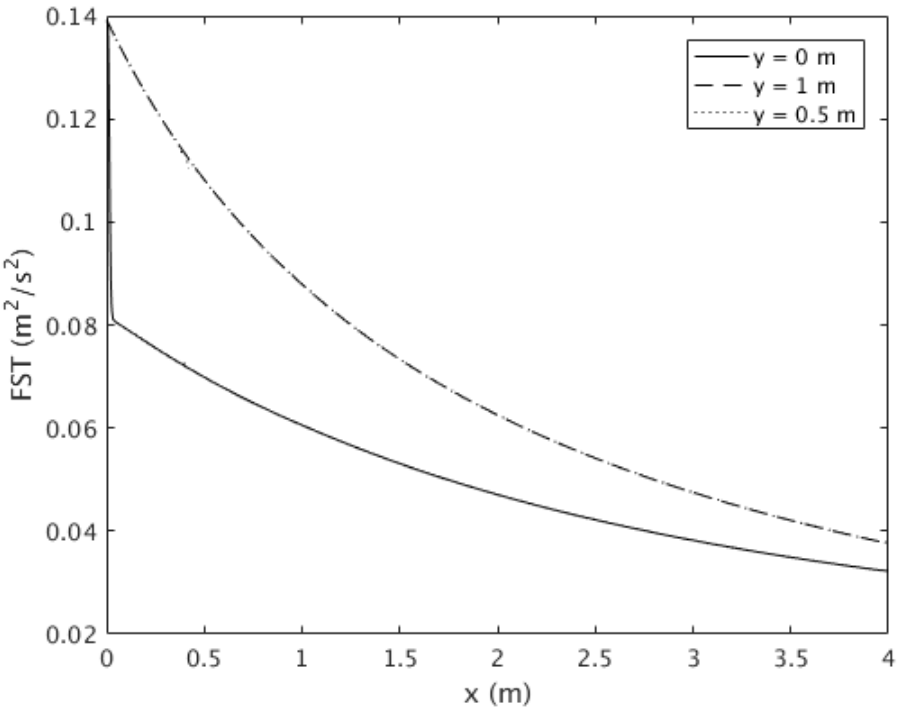


Figure 3.41: FST capturing from UDF for flow over a flat plate

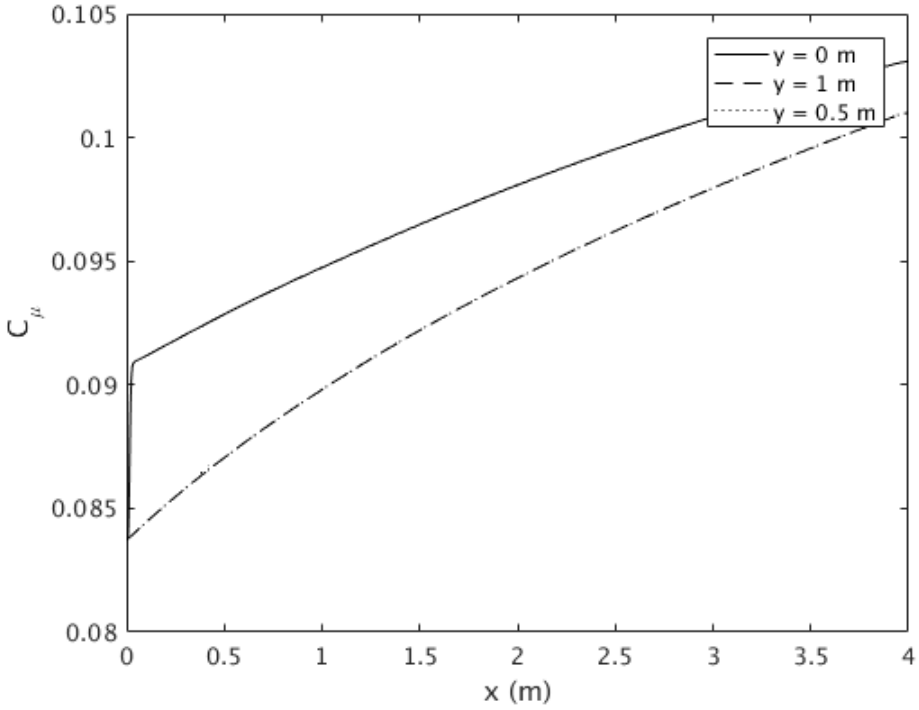


Figure 3.42: Variable C_μ capturing from UDF for flow over a flat plate

software that can provide the source code. Nonetheless, a final test case was run at high intensity to show that c_μ can be modified in Fluent by changing the value in GUI panel. The whole domain is divided into small parts and the profiles at the end of one part are used are initial profiles for the successive part. The results can be further improved by increasing the number of division or parts but such an implementation is just like solving the equations manually and will not help with the purpose of the research. Therefore, switching to another code such as OpenFOAM is considered.

Table 3.3: Variation of c_μ along the plate based on FST values

x(m)	Tu (%)	c_μ
0	24.05	0.0053
0.3	17.29	0.0087
0.6	14.06	0.012
0.9	12.03	0.0153
1.2	10.62	0.0185
1.5	9.6	0.0216
1.8	8.84	0.0245
2.1	8.11	0.0280
2.4	7.62	0.0299

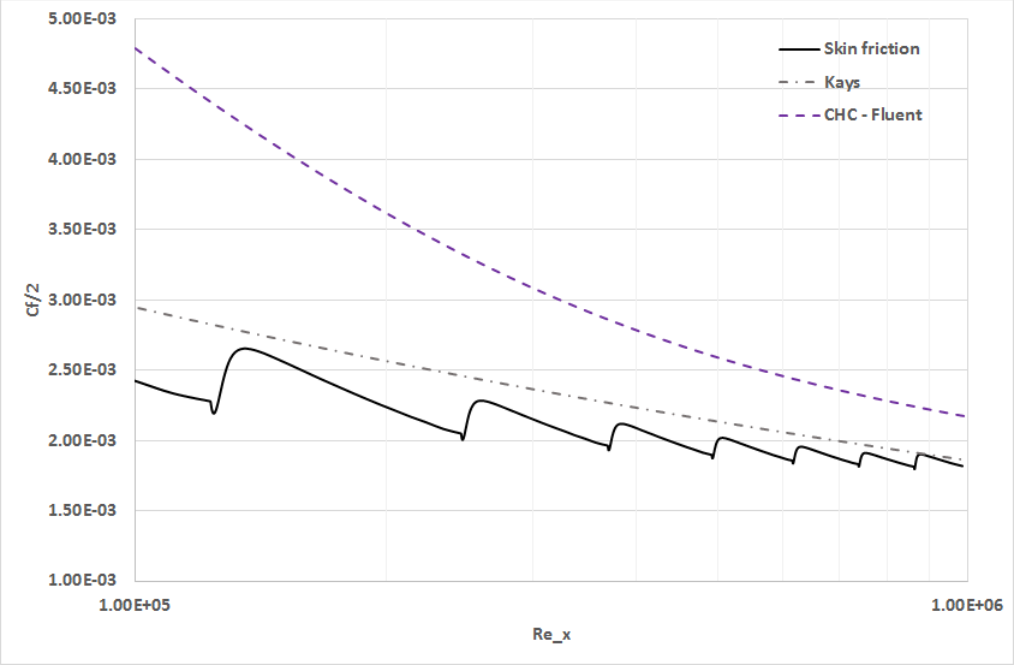


Figure 3.43: Skin friction coefficient over a flat plate for high FST by dividing the domain in streamwise direction and manually providing the corresponding value of c_μ through GUI

Chapter 4
Validation Study in OpenFOAM

OpenFOAM is considered because ANSYS-FLUENT did not provide any method to implement variable c_μ to the existing models. Preliminary study is conducted on a basic case of a two-dimensional, smooth flat plate turbulent boundary layer. A basic study can eliminate all other effects and focuses only on the effects of FST. To obtain consistent results and for comparison purposes, a geometry matching the previous CFD studies [31] is used . The geometry as shown in Figure 4.1 is a 3 m long flat plate with the domain height of 0.2 m. Computation begins ahead of the leading edge because it enables uniform profiles of k and ε to be assigned. This reduced inter-modal spacing in the vicinity of the leading edge thereby saving computational resources.

4.1 Mesh and Boundary Conditions

High quality structured mesh was generated in Pointwise [62]. To use low Reynolds number model so that the calculations are resolved throughout the whole domain without the use of wall-functions very fine mesh near the wall is required. A non-dimensional distance, y^+, is defined whose value should be of order 1 near the wall.

$$y^+ = \frac{yu_*}{\nu} \tag{4.1}$$

u_* is called friction velocity which is defined as $u_* = \sqrt{\frac{\tau_w}{\rho}}$.

To estimate the nearest grid point, shear stress at the wall is required. In the preliminary case, shear stress is obtained from the empirical correlation for skin friction coefficient at the end of the plate. To approximate the first grid location,

$$y_1^+ = \frac{y_1 u_*}{\nu} \approx 0.3 \tag{4.2}$$

$$y_1 = \frac{\nu}{u_*} = \frac{\nu}{\sqrt{\frac{C_f}{2}U_\infty^2}} \tag{4.3}$$

For the case of $U_\infty = 30.48$ m/s, first grid point is estimated as 6.6×10^{-06} m. Next grid locations in the boundary layer region are calculated based on the uniformly stretched grid $\Delta y_{i+1} = \Delta y_i(1 + r)$. Rate r is calculated by using the domain height of 0.2 m and number of mesh points as 180. In OpenFOAM, the ratio of final cell height to initial cell height, R_y, is used to generate the mesh which in this case is 13000. For the stream-wise direction of 3 m length, a similar method is followed but ensuring that the initial aspect ratio is less that 10. R_x is selected as 4700 for this case.

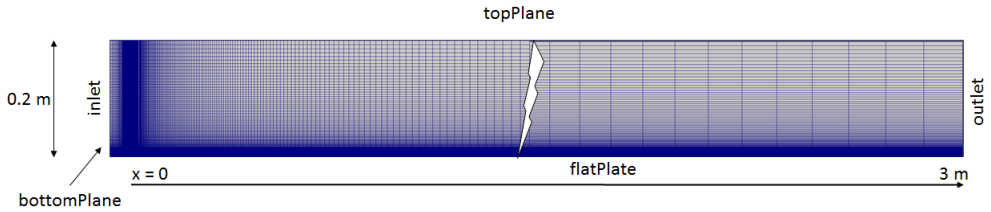


Figure 4.1: Final mesh used for computation study on flat plate in OpenFOAM code

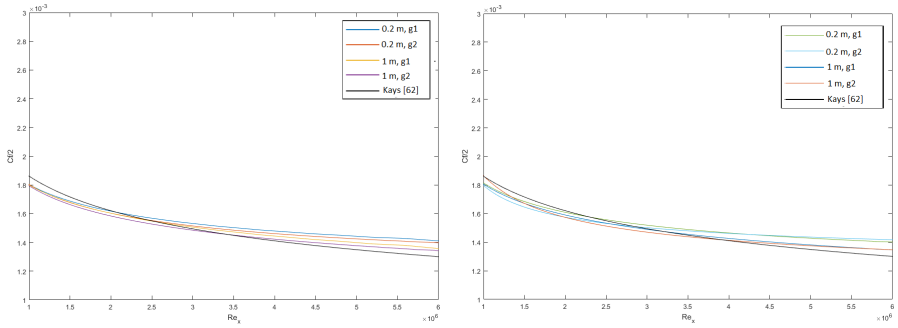
Inlet is selected velocity inlet where uniform values of U, k and ε are provided, outlet as uniform pressure - *zeroGradient* for all variables, top plane as *symmetry*, developing section as *slip* and plate as *no-slip* with velocity as zero. k and ε values should be theoretically 0 at the wall for Launder-Sharma model but to avoid division by zero a small value of 1e-14 is selected. Details of the boundary conditions can be found in Appendix-B.

Sufficient information regarding the boundary conditions for k-*epsilon* model is not present in the literature. Correct initial value of ε is very important to get the decay rate of k which matches the experiment. Initial value of k was calculated based on the initial turbulent intensity as defined by Iyer [48] and [31]. Initial turbulent intensities of 1%, 6.53% and 25.7% corresponded to value of k as 0.1393 m^2/s^2, 5.92 m^2/s^2 and 3.56 m^2/s^2 respectively. Several simulations were run with different values of ε_i to match the experimental decay rates. It was observed that $\varepsilon_i = 2.56$, $\varepsilon_i = 125$ and $\varepsilon_i = 75$ produced decay profiles matching the that of the experiments for low, moderate and high turbulence intensities respectively.

4.2 Grid Independence Test

The test plate was 3 m long and the computations began 0.03 m ahead of the leading edge to enable uniform profiles of k and ε at the inlet. To study the effect of domain height, near wall y+ and grid density, several cases were formed by combination of y+ (0.1, 0.3, 1), domain height of 0.2 m, and grid densities of (140x90, 280x180, 560x360, 1120x720) in x-y direction as shown in Figure 4.1. Skin friction coefficient values along the plate was selected as the grid independence parameter. It was found that 280 stream-wise nodes and 180 stream-normal nodes gave the grid independent solution for both the domain heights as can been seen in Figure 4.3 since results did not changes by doubling the grid in both the directions.

Next the effect of y+ was considered. It was found that the results changes by changing y+ from 1 to 0.3 but no observable change was noticed when changing y+ from 0.3 to 0.1. Results were also compared for the domain height as shown in Figure 4.3 and it was found to affect the skin friction results. The flow acceleration was limited to 1% of the initial velocity. Thus, further analysis shall be carried out for y+ = 0.3 corresponding to domain height of 0.2 m with 280 nodes in x-direction and 180 nodes in the y-direction.



(a) $y^+ = 0.1$ (b) $y^+ = 0.3$

Figure 4.2: Grid independence test for OpenFOAM for near wall y^+ values for flow over flat plate. (g1: grid 1, g2: grid 2. 0.2 m and 1 m are the domain heights)

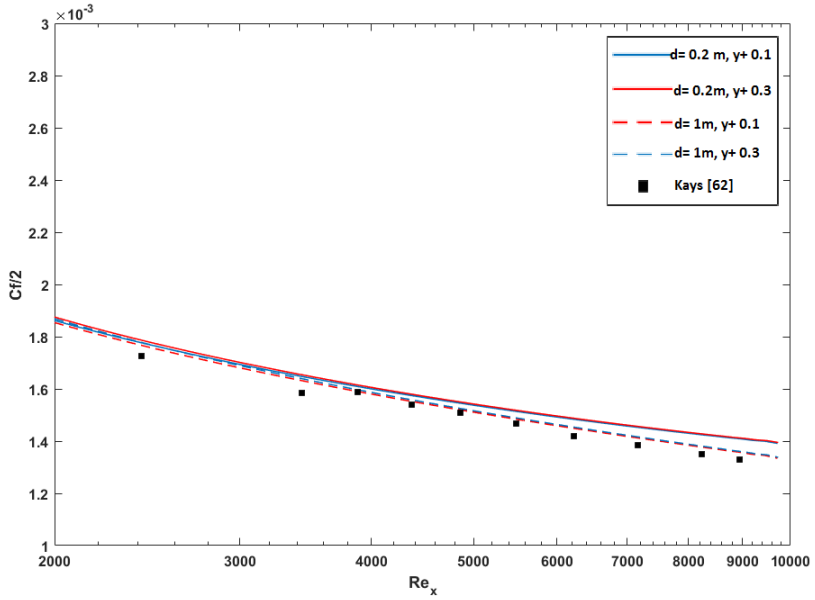


Figure 4.3: Selection of domain height and near wall y^+ values based on the skin friction coefficient over flat plate (d: domain height of 0.2 m and 1 m, y+: 0.1 and 0.2, grid 2)

4.3 Addition of Energy Equation

Steady state solver based on SIMPLE (Semi-Implicit Method for Pressure Linked Equations) algorithm does not include energy equation as such by default. But OpenFOAM provides flexibility to add any transport equation to the existing code. One way is to modify *simpleFoam.C* and add the temperature equation as defined in Chapter -2.

Upon further investigation of the inbuilt solvers in OpenFOAM it was found that *buoyantBousinesqSimpleFoam* uses SIMPLE algorithm but with the Bousinesq approximation. It can be converted to incompressible *simpleFoam* solver by defining the gravity as 0 and the coefficient of expansion (or density variation due to temperature change) as 0 in the *constant* folder. This would also eliminate any uncertainty or possibility of error in modifying the existing solvers and was chosen as the default for further simulations of steady state.

4.4 Comparison of LRN Models

Results from the Launder-Sharma model in OpenFOAM were compared to the results of Chien et al [63] and Abe et al [61] models in Fluent and the experimental data. The experimental data of Weighardt and Tillmann [64], and TKE profile data from Klebanoff [65] was selected for comparison at Tu = 1 %; Blair and Werle [7], and Blair and Edwards [11] data was used at Tu = 6.53 %; and Ames and Moffat [23] was used at Tu = 25.7 %. Due to absence of TKE profiles for the highest turbulent intensity data from Yavuzkurt and Batchelder [13] was used.

It is observed in Figure 4.4 that skin friction coefficient increases with turbulent intensity at a given Reynolds number. CFD data for highest turbulent intensity was limited to Reynolds number of 10^6 due to the lower inlet velocity of 6 m/s

4.4.1 Low $Tu_i = 1\%$

Figure 4.5 shows variation of skin friction coefficient in streamwise direction plotted against Reynolds number based on momentum thickness. It indicates that the results from all the models are in good agreement (within 3%) with both the experimental data and the empirical correlation 4.4 of Kays and Crawford [60]

$$Cf/2 = 0.0125 Re_\theta^{-0.25} \tag{4.4}$$

Dimensionless kinetic energy profiles at $x = 2m$ or $Re_\theta = 5000$ are plotted in figure 4.7 The results obtained from LS model in OpenFOAM are compared with LRN models of CHC and AKN in Fluent and the data. All the models gives poor predictions very near the wall but compare well in the free-stream. Peak value of dimensionless TKE for LS model is 0.005 m^2/s^2 , AKN is 0.007 m^2/s^2 and CHC is 0.006 m^2/s^2 which is under-prediction of 22%, -11% and 4% compared to the data. Away from the wall all the models over-predict the value which can be

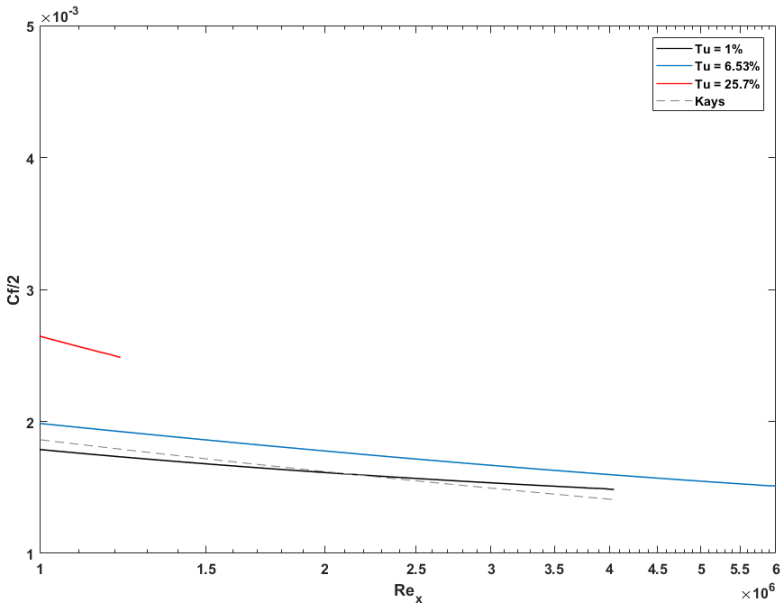


Figure 4.4: Effect of freestream turbulence on skin friction coefficient for flow over flat plate using LS model in OpenFOAM. The curve for high FST is short since the velocity used for this case in 6 m/s as compared to 30.48 m/s for other cases

explained due to the fact that the experiment was carried at much lower freestream turbulence than the CFD study.

Thus it can be concluded that all the models show good prediction at low turbulence intensities since they were empirically adjusted at these turbulence levels.

4.4.2 Moderate Tu_i = 6.53%

Free-stream velocity was set at 30.48 m/s for this case. Figure 4.8 depicts the results for skin friction coefficient. It is observed that the agreement between LRN models and experimental data decreases in this case as compared to the low FST case. Over-prediction of 4%, 5% and 4% is observed for LS, AKN and CHC models respectively. This can be attributed to the incorrect prediction of TKE near the wall and also to coefficient c_μ used in the models as explained in [31]. Similar observations are made for Stanton number where the over-prediction of 4%, 5% and 8% are respectively seen for LS, AKN and CHC models.

TKE profiles at x = 1.32 m and x = 1.73 m are shown in figure 4.10. All the models show under-prediction of up to 22% inside the boundary layer except CHC which shows an over-prediction of 4%. This show that these models are unable to capture effects of FST in enhancing TKE value inside the boundary layer through diffusion from free-stream.

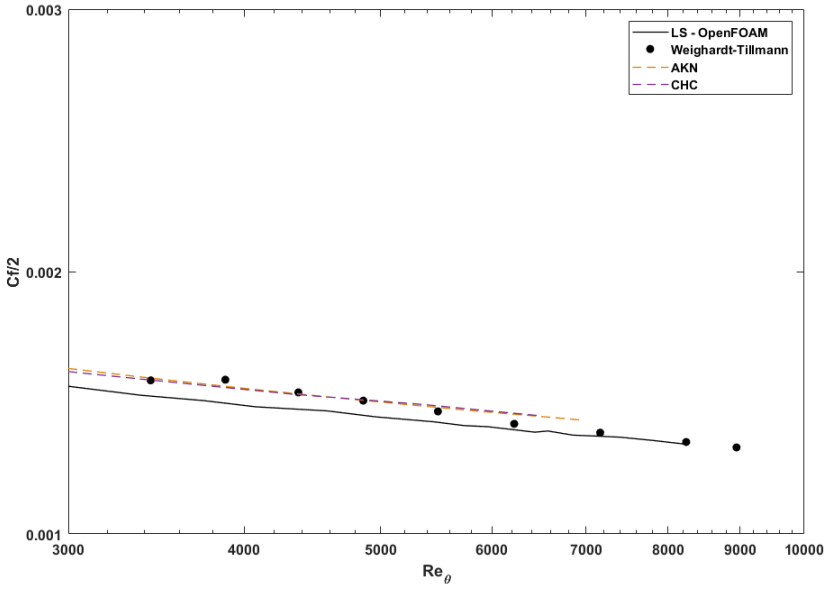


Figure 4.5: Skin friction coefficient for flow over flat plate for low turbulent intensity of Tu = 1% in OpenFOAM - baseline results

4.4.3 High Tu_i = 25.7%

The freestream value was set at 6 m/s and the data uncertainty was reported as ±4.5% for Cf/2 and ±4% for St.

Comparison of skin friction coefficient and Reynolds number based on momentum thickness can be seen in Figure 4.11. It is observed that the deviation of CFD results and the data increases on increasing turbulent intensity. LS model over-predicts Cf/2 by 20% on average while AKN and CHC over-predicts by 24% and 26% respectively. Results for Stanton number are similar where over-prediction of 32%, 40% and 28% is observed for LS, AKN and CHC models respectively.

Profiles of TKE are shown in Figure 4.13. Results from all the models highly under-predicts TKE level both inside and outside the boundary layer by upto 48%. It was explained by Iyer and Yavuzkurt [29] that diffusion of eddies with high turbulence intensity into the boundary layer from the free-stream will increase as FST increases. Changes in boundary layer structure due to diffusion of high fluctuating free-stream flow and also due to velocity-pressure gradient interactions can cause extra turbulence production which will results in increasing TKE levels inside the boundary layer. It is assumed that failure of the present models to capture this phenomena results in under-prediction of TKE inside the boundary layer.

It is validated by comparison from the Fluent and OpenFOAM results that the LS model is correctly setup in OpenFOAM. The results for moderate and high FST were also compared with Foroutan & Yavuzkurt [31]. It was found that skin friction and Stanton number for the present study were lower though the TKE decay and TKE profiles matched very well. This observation leads to one of the two conclusion - First, the present setup has some calculation mistake or, second, previous study had some difference in the boundary conditions. The first conclusion

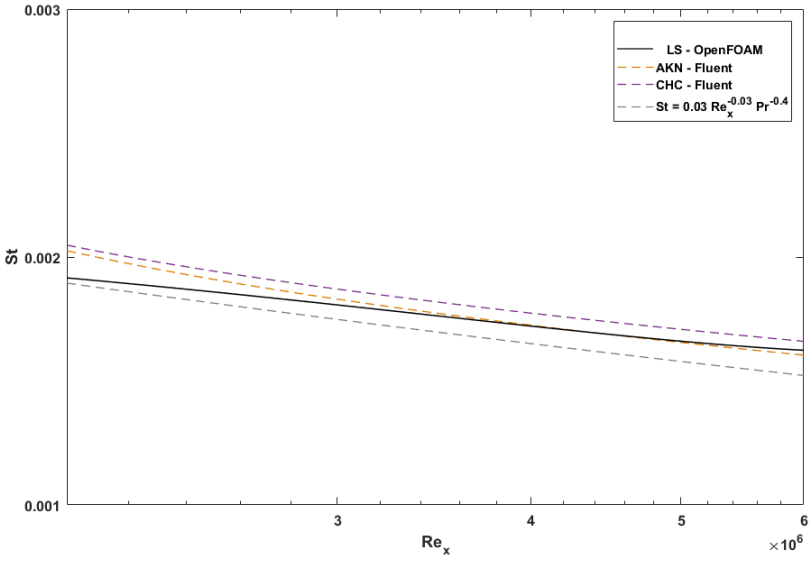


Figure 4.6: Stanton number for flow over flat plate for low turbulent intensity of $Tu_i = 1\%$ in OpenFOAM - baseline results

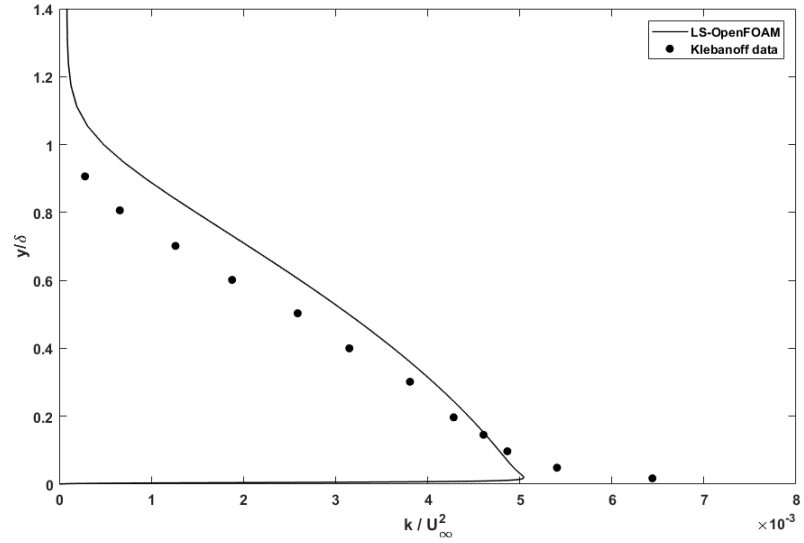


Figure 4.7: Turbulent kinetic energy profile for flow over flat plate at x = 2 m for low turbulent intensity of $Tu_i = 1\%$ in OpenFOAM - baseline results

does not hold well since accurate results were obtained for low turbulence intensities and the skin friction calculation was compared by three different method - using the utility *wallShearStress*, using utility *wallGradU* and by calculating velocity gradient from the obtained velocity profiles. Thus in all likelihood and the absence of the case files and all the boundary conditions in the literature, it may be not be possible to get the exact results with different boundary conditions.

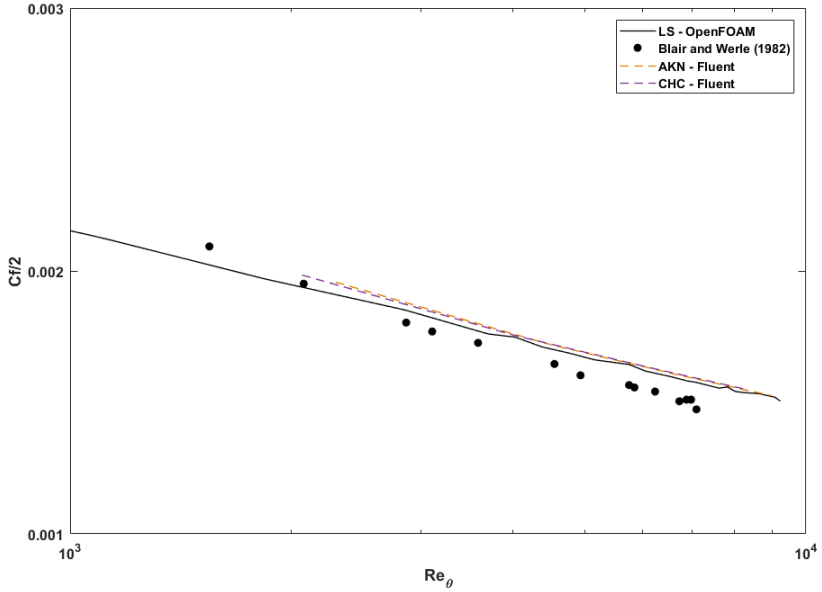


Figure 4.8: Skin friction coefficient for flow over flat plate for moderate initial turbulent intensity of $Tu_i = 6.53\%$ in OpenFOAM - baseline results

4.5 Implementation of High FST TKE Model for Flow Over a Flat Plate in OpenFOAM Code

Modification in the model is performed by reformulation of c_μ and addition of an extra diffusion term to capture the effect of FST. Iyer [48] stated that the constant c_μ was obtained by the ratio of $\frac{u'v'}{k}$ near the wall at negligible free-stream turbulence. He recalculated the value of c_μ through the value of the ratio of turbulent shear stress to near wall turbulent kinetic energy. Other models used $c_\mu = 0.09$ which results in high value of turbulent viscosity and thus near wall data under high FST [45] [61] [63]. The new formulated c_μ was found by curve-fitting this ratio Yavuzkurt [13] as:

$$\begin{aligned} C\mu &= 0.0837 - 0.061 log_{10}(Tu_\infty) \qquad Tu_\infty \leq 8\% \\ C_\mu &= 0.7 Tu_\infty^{-1.538} \qquad Tu_\infty > 8\% \end{aligned} \tag{4.5}$$

where Tu_∞ is the free-stream turbulence intensity. Thus first step in modeling the new term is to make the constant c_μ into a local variable. It is defined as the *volScalarField* in OpenFOAM. Similarly a new variable is created to get the FST values from the free-stream. OpenFOAM changes the node numbering from global to local to get a compact matrix (or make the matrix equation less sparse) for faster calculations. So traversing through the entire domain is required to obtain the free-stream values of TKE and ε. This means that for the present mesh of $280x180 = 50400$ nodes, traversing through the entire domain would be required for all x-locations. This would amount to huge calculations per iteration of the range of $50400 \times 280 \approx 15$M. This makes the

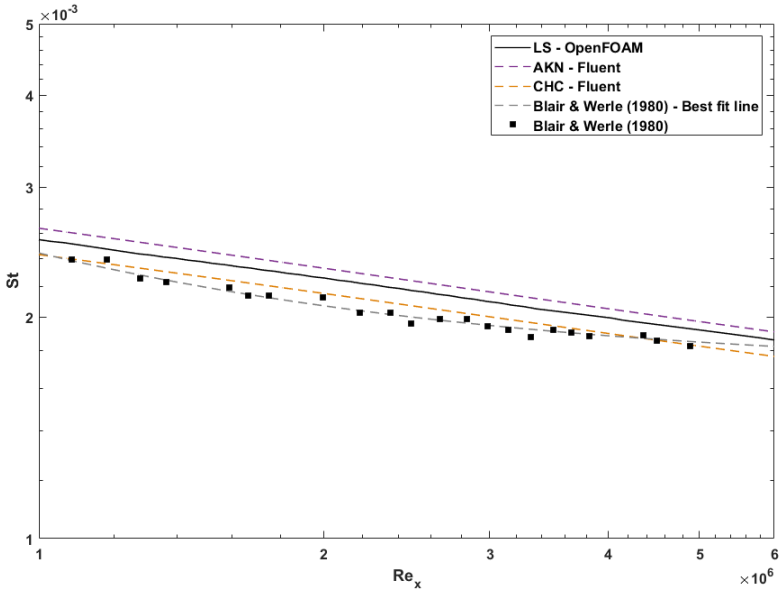


Figure 4.9: Stanton number for flow over flat plate for moderate initial turbulent intensity of $Tu_i = 6.53\%$ in OpenFOAM - baseline results

model computationally demanding offsetting its effectiveness. The other way around is to use the TKE decay and ε decay rates, and insert the curve fit equations. This would lead to quicker calculations but does not make the model independent of the geometry. In the present implementation the free-stream values are obtained by traversing through the top boundary patch named *topPlane* since OpenFOAM provide the name of the boundary. This can be directly called to extract free-stream values thereby reducing the calculations by at least 300 times. Detailed code for the implementation can be found in Appendix-B.

Near wall TKE values have significant impact on the wall fluxes (shear stress, heat transfer). Thus to correct the near wall TKE an additional diffusion term was introduced in equation (2.16).

$$D_{FST} = -\frac{\partial}{\partial x_i}\left[C_{FST}\overline{u_i u_j}U_j\right] \tag{4.6}$$

Equations in OpenFOAM are written in tensorial form using the *div* and *grad* keywords for divergence and gradient respectively. Also, the ratio of Reynolds stress to density is predefined as volume scalar function *R()*, which returns a tensor.

Thus the 4.6 is written as

$$D_{FST} = \nabla.(C_{FST}\bar{\bar{R}}.\bar{U}) \tag{4.7}$$

It is converted to OpenFOAM keyword as $fvc :: div(CFST * (R()\&U))$.

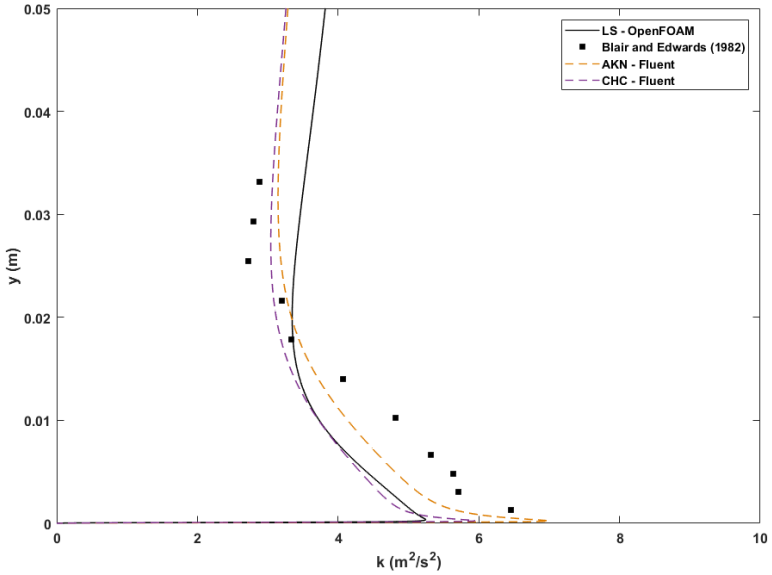
C_{FST} was found by Iyer [48] to include the effect of both the turbulent kinetic energy, length scale and the boundary layer thickness.

$$C_{FST} = 0.076\Big(\frac{L_u^{\infty}}{\delta}\Big)^{0.208} = \quad 0.076\Big(\frac{k_{\infty}^{1.5}}{\varepsilon_{\infty}\delta}\Big)^{0.208} \tag{4.8}$$

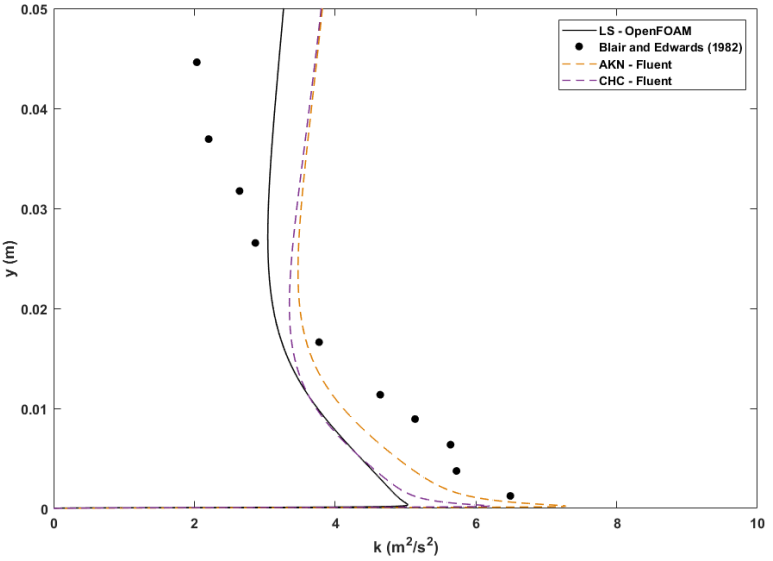
Here boundary layer thickness was inserted as an analytical function to save computational time. Approximate value of thickness should be enough since one-fifth power would reduce any error to a small value. This also ensures that C_{FST} is limited to a small range throughout the whole domain and affect the regions only with high velocity gradients that is near the wall since free-stream decay of TKE of the original model has to be preserved.

Figure 4.14 shows the prediction of skin friction coefficient for moderate turbulent intensity compared to the data set of Blair and Werle [7] and the LS model in OpenFOAM. It is observed that the new model shows better predictions when compared to the data than the LS model. Similar improvement in the Stanton number prediction can be observed in Figure 4.15 where the new model shows better capability. The data set in this case was sparsely distributed so a best fit line is also plotted to represent the experimental values. The error in the peak TKE value at x = 1.73 m is limited to 11 % for the new model when compared to LS model that under-predicted peak TKE value by 22 %. Better fit for the remaining portion of the profile is also observed in figure .
The actual capability of the model is shown when the results at high FST are compared. Figure 4.16 shows the skin friction results where a significant improvement is shown as compared to the standard LS model. The error in skin friction coefficient is limited to just 5 % as compared to 20 % over-prediction by the standard model. This behavior is observed for skin friction coefficient as well where significant improvement is observed to limit the error to 11 % as compared to 33 % by the standard model. This improvement can be explained by observing the TKE profiles it figure 4.18. New model shows better predictions of near wall TKE and also overall profile match with the data as compared to the standard model which highly under-predicted peak TKE values. Error in peak TKE prediction is limited to just 10 % which was as high as 48 % for the standard LS model. Though the overall predictions are much better but the new model cannot exactly capture the correct slope of the graph and need further improvement which is not in the scope of the current study.

Implementation of the new model along with reformulation of c_μ improves the result for all the cases. It can now be said with confidence that the new model has been correctly implemented in OpenFOAM and can now be used for further study on curved surfaces which is the main focus of this thesis.



(a) x = 1.32 m



(b) x = 1.73 m

Figure 4.10: Turbulent kinetic energy profiles for flow over flat plate at two different streamwise locations for moderate initial turbulent intensity of $Tu_i = 6.53$ % in OpenFOAM - baseline results

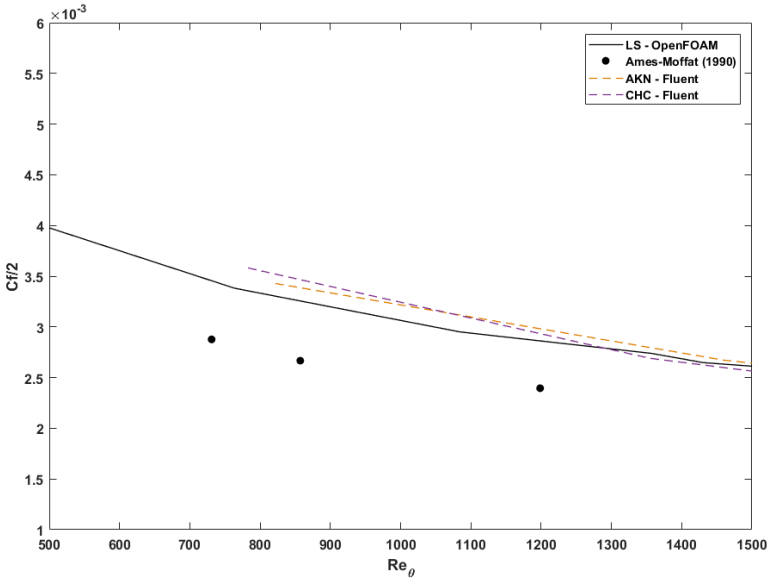


Figure 4.11: Skin friction coefficient for flow over flat plate for high initial turbulent intensity of Tu = 25.7% in OpenFOAM - baseline results

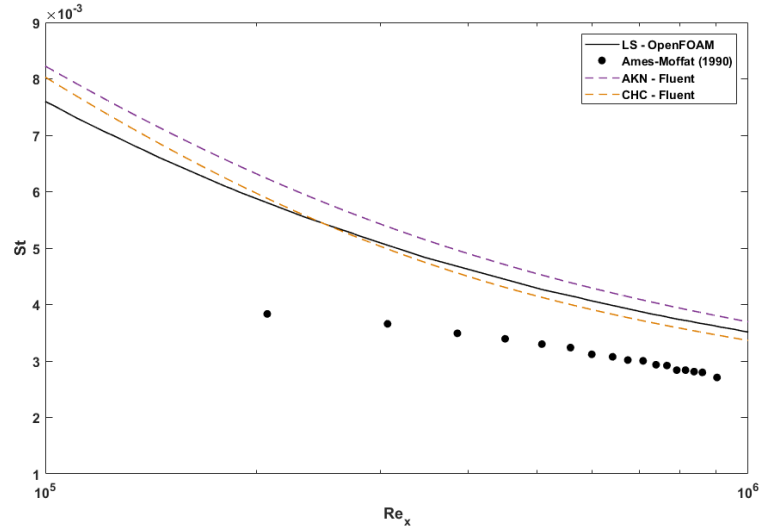


Figure 4.12: Stanton number for flow over flat plate for high initial turbulent intensity of Tu = 25.7% in OpenFOAM - baseline results

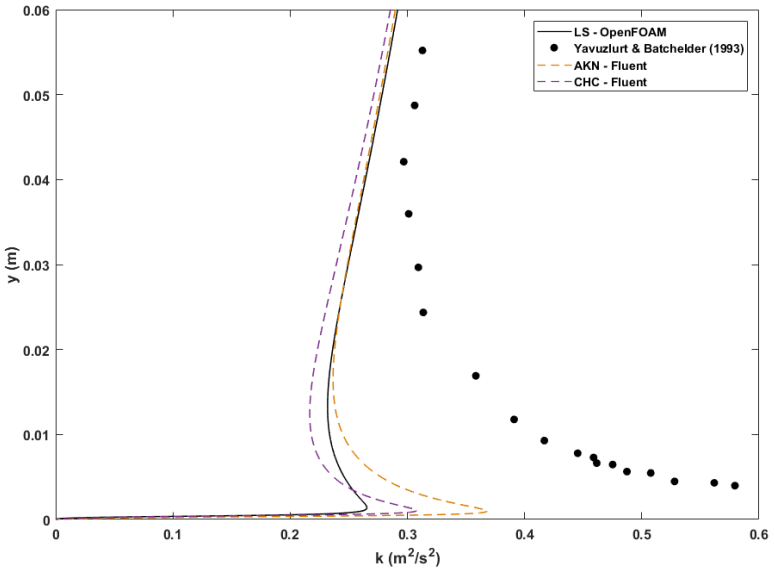


Figure 4.13: Stanton number for for flow over flat plate high initial turbulent intensity of Tu = 25.7% in OpenFOAM - baseline results

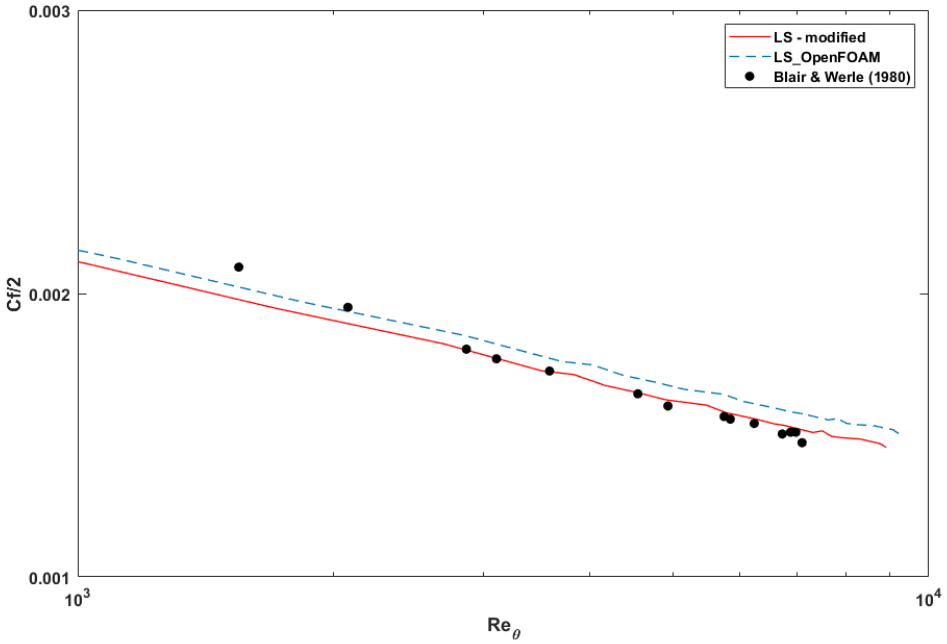


Figure 4.14: Skin friction coefficient prediction for flow over flat plate by the FST diffusion model for moderate initial turbulent intensity of $Tu_i = 6.53$ % implemented in OpenFOAM

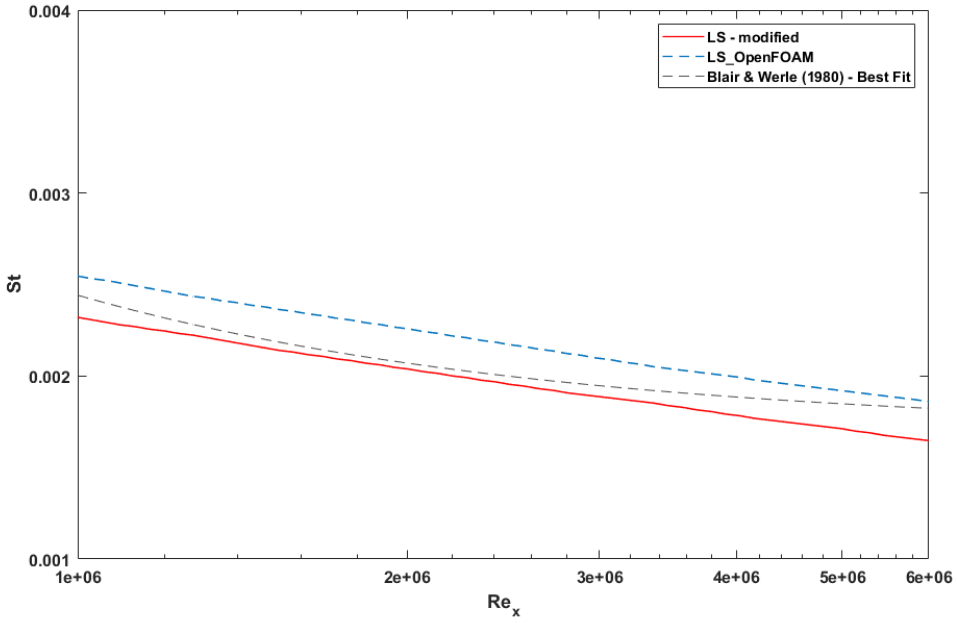


Figure 4.15: Stanton number prediction for flow over flat plate by the FST diffusion model for moderate initial turbulent intensity of $Tu_i = 6.53$ % implemented in OpenFOAM

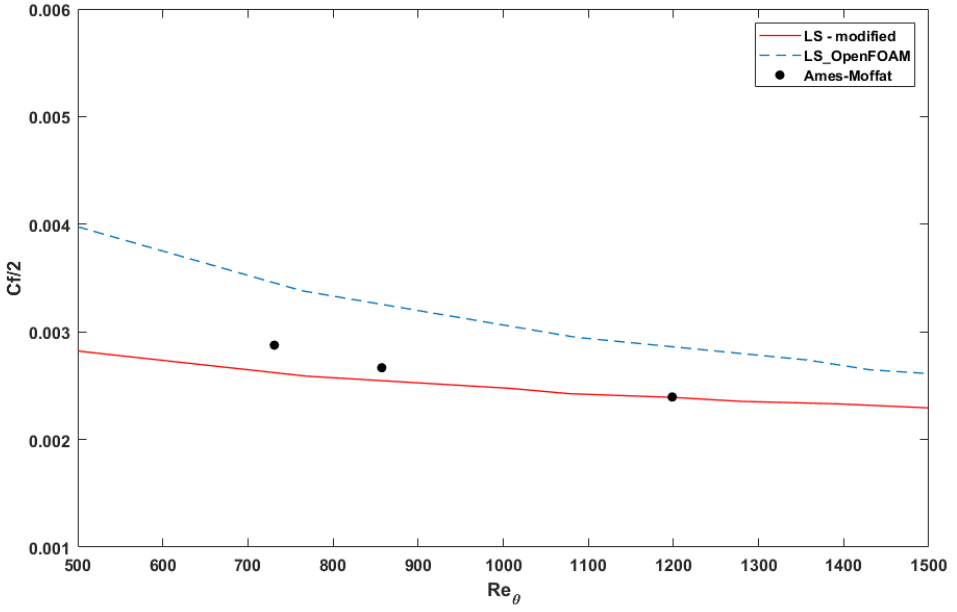


Figure 4.16: Skin friction coefficient prediction for flow over flat plate by FST diffusion model for high initial turbulent intensity of $Tu_i = 25.7$ % implemented in OpenFOAM

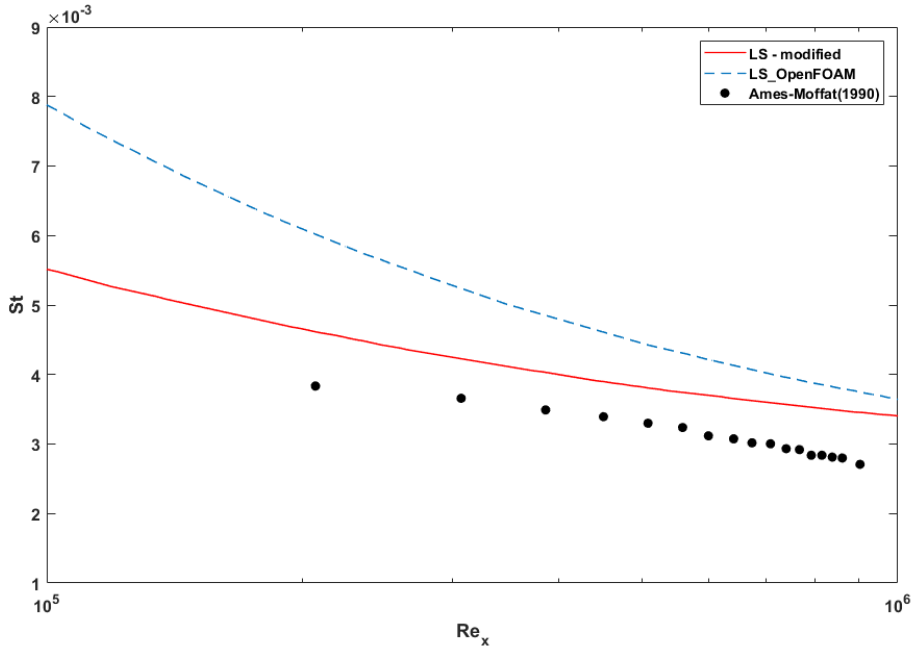


Figure 4.17: Stanton number prediction for flow over flat plate by the FST diffusion model for high initial turbulent intensity of $Tu_i = 25.7$ % implemented in OpenFOAM

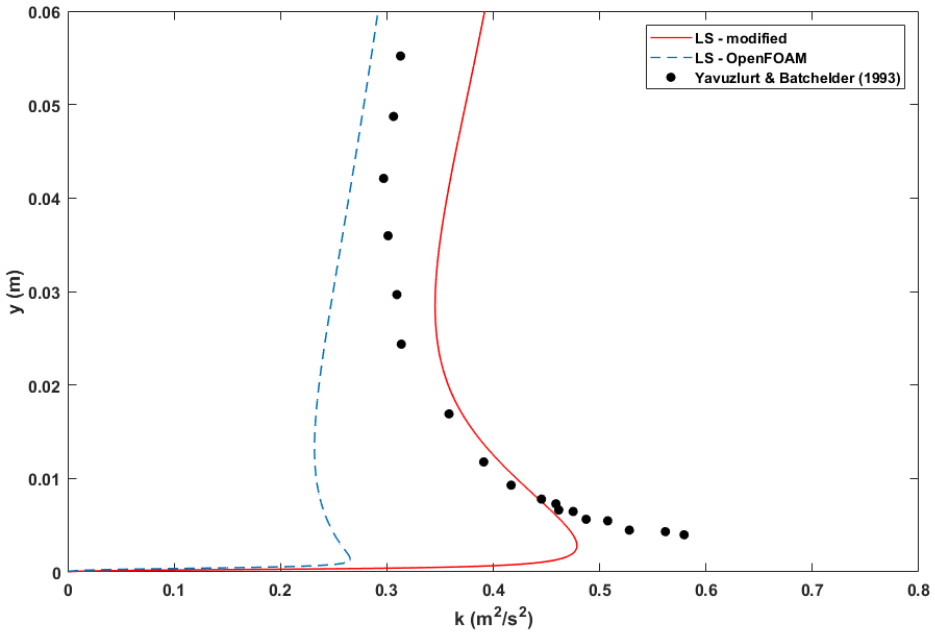


Figure 4.18: Turbulent kinetic energy profile for flow over flat plate at x = 2.08 m for FST diffusion model at high initial turbulent intensity of $Tu_i = 25.7$ % implemented in OpenFOAM

Chapter 5
New Model Implementation on a Curved Surface of a Gas Turbine Blade

Real test of the present model would be its prediction capability on curved surfaces since gas turbine blades are curved. Convex surfaces decelerates flow and can lead to flow separation, and concave surface cause flow acceleration and thereby stable boundary layer. So, the geometry should be such that it accounts for both of these effects. Also, it is very important that the test case should have comprehensive experimental data at both low and high FST for comparison. Thus on extensive literature survey, it was decided to use a C3X vane cascade of Hylton et al [21].

5.1 Geometry Selection

A 2-D simulated model to match the computational model of Dees et al. [1] and Dyson et al [2] is used. The C3X vane used for this CFD study is approximately eight times the commercial aircraft engine first vane with a chord length of 531 mm.

Accuracy of the solution highly depends on the quality of the mesh used for simulation. This study focuses on the effect of high freestream turbulence effects on the wall fluxes thus it is very important to correctly resolve the momentum and thermal fields near the wall. This prevents the usage of wall functions and thus low Reynolds number models are used that require fine mesh near the wall. Since the geometry is complex so the mesh generation approach changes. A 6 mm thick prism layer is generated near the wall with the wall y^+ less than 1 as shown in Figure 5.2. This leads to near wall distance of 10^{-5} m for the first grid point. Away from the prism layer in Figure 5.1, the domain is divided into several segments and structured mesh is generated in Pointwise to limit the non-orthogonality of the mesh. The final mesh contains 62250 hexahedra cells with average non-orthogonality of 8.9 limited to a maximum non-orthogonality of 45.9.

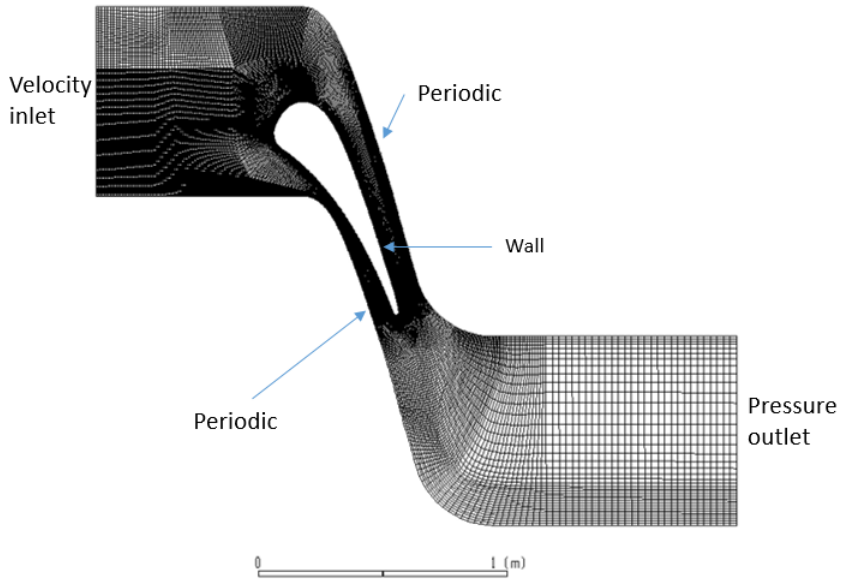


Figure 5.1: Structured hexahedral mesh for the simulation of C3X vane cascade along with the major boundary conditions

Profiles measurements are taken at 7 locations on the vane which are mentioned in Table 5.2 and shown in Figure 5.3. Profile measurements were taken along the normal to the surface at these location which were obtained by developing the equations of the normal lines at these points and locating another point at 10 mm away in the freestream. This was done in the current study since all the experimental results are within this normal distance from the wall.

Table 5.1: Measurement locations for the CFD study of C3X vane cascade

Position name	Location (s/C)
PS1	-0.19
PS2	-0.38
PS3	-0.56
SS1	0.19
SS2	0.38
SS3	0.56
SS4	0.75

5.2 Comparison of RANS Model Results

Current CFD study of the turbine vane cascade is performed in OpenFOAM using two different RANS models - k-ω SST (4 equation model) and Launder-Sharma (LS) model (2 equation). k-ω SST is selected for validation of current CFD study with that of Dyson et al [2] where LS model is selected to implement the TKE diffusion model. Along with the OpenFOAM, baseline results of

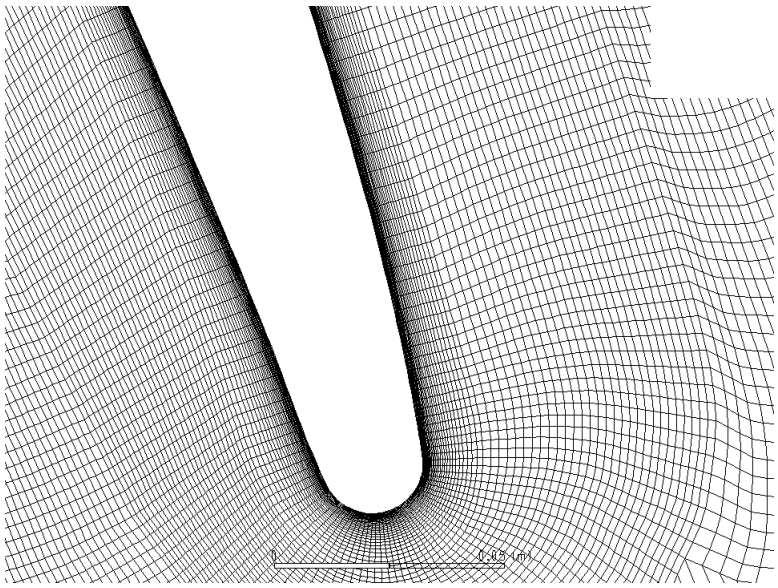


Figure 5.2: Near wall view of the mesh at the trailing edge of the airfoil to demonstrate y+ < 1 near the wall for simulation using low Reynolds number models

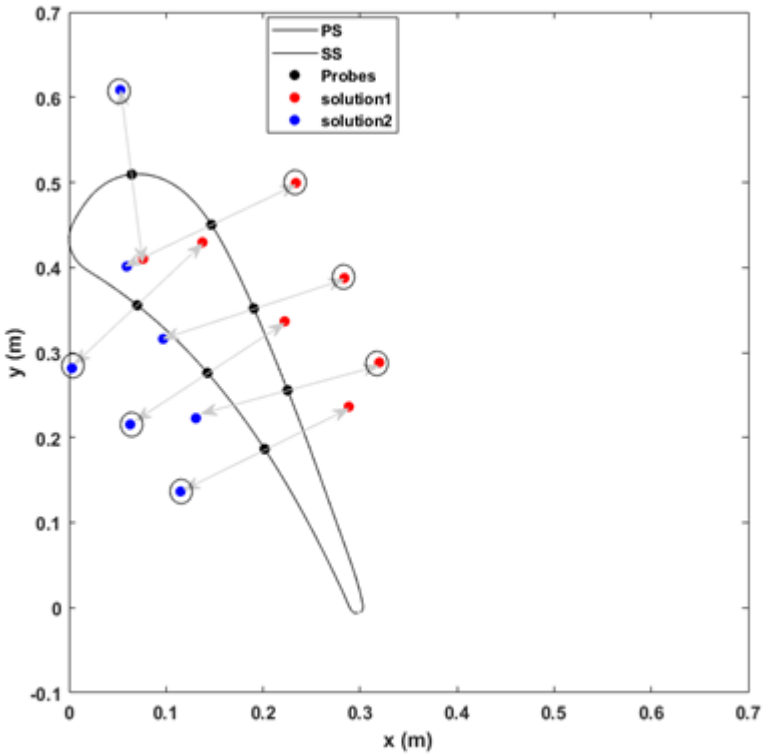


Figure 5.3: Location of the measurement probes and the starting and end points where the profiles are obtained

kw-SST model from ANSYS-Fluent are also compared since the computational study by Dyson et al [2] used the same software.

Comparison of pressure coefficient around the vane matches well with the literature data throughout the vane surface as can be observed in 5.4. Current CFD in OpenFOAM slightly under-predicts the coefficient of pressure on the suction side (SS) at the location of maximum acceleration that is at s/C = 0.27. This can be explained by the fact that the inlet velocity in the present study is 5.48 m/s which is slightly less than 5.8 m/s which was used in the experiments. The experimental value of velocity led to instability in some of the OpenFOAM cases so a lower value of velocity was selected. But the effect of velocity should be negligible since all the comparisons are made in non-dimensional form.

Due to acceleration in the passage, the velocity continuously changes and this made finding the edge velocity at each location ambiguous. Thus, the data was normalized in two ways. On the pressure side, the freestream velocity was extrapolated to obtain the velocity at the wall for inviscid condition, U_p. On the suction side, the velocity was continuously increasing as the wall was approached. Thus, depending on the point of selection, maximum velocity, U_{max} was selected to normalize the profiles.

Temperature profiles were plotted by using a non-dimensional temperature such that θ given in Equation 5.1 is 0 at the wall and 1 in the freestream. A constant freestream temperature of of 305 K wall selected for all the simulations. Wall conditions of constant heat flux and constant temperature were compared by Dyson et al [2] which showed negligible variation. Also, there was no difference in the hot or cold wall on the non-dimensional thermal profiles since the flow properties were constant due to low temperature difference between the wall and the free-stream and such a change affects only the direction of heat flow. Thus, for simplification a constant wall temperature, $T_w = 330K$, is selected for the current study.

$$\theta = \frac{T - T_w}{T_\infty - T_w} \tag{5.1}$$

To be consistent with the previous CFD study the boundary layer thickness was taken as 95 % instead of common 99. The rational behind this was that in the latter case even small differences would create large mismatch in the experimental and computational values. Density variation produced very little effect of less than 0.2% change in coefficient of pressure thus the case was solved as incompressible using *buoyantBousinesqSimpleFoam* with the modifications described in the previous chapter.

Before discussing the results from OpenFOAM simulations, a common model from Dyson [2], ANSYS-FLUENT and OpenFOAM was selected to check the correct implementation of the boundary conditions. All the boundary conditions were kept as close to Dyson et al [2] except the velocity inlet condition which was kept at 5.48 m/s in the current simulations. Figure 5.4 shows

the coefficient of pressure around the vane which shows excellent comparison for all the studies and the experimental data. Investigation of Nusselt number in Figure 5.5a at low freestream turbulence of 0.5 % shows good overall comparison on the suction side after the transition but under-prediction as compared to the data. Pressure side trends are also in good match except that the current study in Fluent predicts Nusselt number almost exactly as the data whereas the OpenFOAM model starts to deviate near the trailing edge. None of the computational studies could resolve the dip in Nusselt number at s/C = 0.4. Current study in Fluent shows the dip slightly earlier as compared to the data and the Dyson's CFD study. Figure 5.5b shows Nusselt number comparison at higher initial freestream turbulence. Here the results from the current CFD study in Fluent almost exactly matches to that of Dyson et al [2] and very good match after transition on the suction side. kwSST model in OpenFOAM under-predicts Nusselt number on the suction side. All CFD studies give same result on the pressure side but all start deviating from the data after s/C = 0.25. None of the CFD study could predict the dip in the Nusselt number on the suction side instead they all show completely opposite behavior by having a peak value of Nusselt number at this location.

The differences in present study in Fluent and literature can be attributed to the fact that the present study is 2-D whereas Dyson conducted a 3-D study which enables them to capture Taylor-Gortler vortices on the pressure side for low turbulence level. Also, at lower initial turbulence level the length scale at the inlet boundary was not defined in the literature which can also explain this behavior though the effect at this low turbulence level will be minimal. OpenFOAM model shows similar trend to Fluent k-ω-SST model but with slight differences. This is due to the fact that current simulation in Fluent uses low-Reynolds number correction option applied to kw-SST model which is absent in OpenFOAM. This study establishes the correctness of the case setup so the focus can now be shifted to k-ε model which is the focus of the present study.

5.2.1 Low FST, Tu_i = 0.5% Results

Inlet velocity is taken as 5.48 m/s and the outlet as zero-gradient pressure. Zero-gradient pressure boundary condition sets the Neumann boundary condition for pressure at the inlet and the pressure inside the domain is calculated based on the reference value given at the inlet. Boundary surfaces are shown in Figure 5.1. Inlet turbulent intensity is 0.5 % corresponding to integral length scale, $\Lambda = 320mm$. The length scale used by OpenFOAM and Fluent is the energy length scale with is approximately one-sixth of the integral length scale. The initial value of $\varepsilon_i = 0.007m^2/s^3$ was calculated from equation 5.3. The wall boundary condition was kept as no-slip and machine epsilon values for k and ε was kept at the wall to prevent division by 0. Turbulent viscosity was given a fixed value of 0 at the wall or equivalently can be defined using *nutLowReWallFunction*. Since the mesh points on the top and bottom periodic boundaries were different they were defined using *cyclicAMI* boundary condition. Details of the implementations can be found in Appendix-B.

Figure 5.6 shows comparison of Nusselt number predictions low initial turbulence intensity. It is found that LS model gives overall better prediction at the pressure side than k-ω-SST model. LS model results in over-prediction of 17 % on pressure side and 61 % on suction side of the turbine vane whereas k-ω-SST over-predicts by 25.7% on pressure side and under-predicts by 37 % on the suction side. Similar to other models, LS model completely fails to predict the dip in heat transfer near the transition region on the suction side and instead it deviates the most when compared to other models. The trend for LS model is similar to realizable k-ε (RKE) model used by Dyson. It is to be noted the present study using LS model showed a very high peak for Nusselt number at the leading edge stagnation point, so, this portion was clipped from the figure. It seems that LS model is unable to predict heat transfer at the region of very strong acceleration and also does not predict transition. Another reason for the deviation from experimental data can be the presence of Taylor-Gortler vortex which could not be predicted accurately by the present models. Also, computational study has smoother edges as compared to the experiments which further prevents the exact behavior of these vortices formations. It has to be noted that due to strong acceleration of the pressure side, it was experimentally determined that the flow is laminar. But the LS model used is fully turbulent thus leading to over-prediction. LS model matches well on the suction side given the turbulent boundary layer but it lacks any kind of transition prediction capability.

LS model predicts velocity boundary layer profile almost exactly at pressure side location PS2 near the wall except that it over-predicts boundary layer thickness by 20% compared to the data [1]. Further downstream it almost exactly matches the data with deviation of only 8%. On the suction side, k-ω-SST model predictions almost exactly matches the boundary layer thickness given in data but none of the model could predict exact near wall behavior. LS model shows poor prediction away from the wall. Near the end of transition at SS3, none of the model could capture the velocity profile. k-ω-SST model came close in predicting the boundary layer thickness but overall profile shape was poor. After the transition at SS4, LS model shows much better predictions almost exactly matching the data whereas k-ω-SST model could not show good prediction. LS model showed 100% over-prediction of velocity boundary layer thickness on the suction side compared to the data of Dees et al [1] with the predictions improving downstream on the suction side. On an average, LS model differed from the data by 2.3% on the pressure side and 8.1 % on the suction side with standard deviation in the average error being 3.4% and 3.5% for pressure and suction sides respectively.

Both the models shows good near wall behavior at PS1 but LS model deviates away from the wall showing a more turbulent kind of profile rather than near laminar profile as predicted by the data. LS model outperforms k-ω-SST model at PS2 for the near wall behavior but then a similar trend to PS1 is observed. At SS2, both the models perform poorly to determine temperature profile. k-ω-SST model predicts correct thermal boundary layer thickness but the shape of the profile predicted by LS model is better. LS model over-predicts thermal boundary layer thickness by 115 % at this location. Just after transition at SS3, k-ω-SST model seems to perform better

than LS model to predict the thermal profile but LS model provides a smoother profile. The poor prediction of temperature profile seems to be due to an over-prediction of thermal diffusion. Thermal profile predictions showed an average error of 6% on the pressure side and 5% on the suction side for LS model.

k-ω-SST model predicted turbulence level of 2.7% near the wall at PS1. This was worse for LS model which was predicted 9.1 % turbulence level at the same location. Similar poor behavior was observed at SS2 with LS model predicting very high freestream kinetic energy values. Prediction for TKE profiles become better at SS3 for kwSST but LS model completely fails to provide good prediction. This shows incapability of LS model to perform under deceleration. In all, both the models over-predicted TKE profiles with an error exceeding 100 %.

5.2.2 High FST, Tu_i = 20% Case

The inlet velocity is kept same as the low turbulence case at $U_\infty = 5.48$ m/s. Turbulent intensity at the inlet is taken as 47% which decays to the desired value of 20 % at $\frac{x}{c} = -0.27$. The initial length scale is 3.082 mm matching that of the experiment. The value of k is calculated from Equation 5.2 using the inlet velocity and turbulent intensity as 9.95 m^2/s^2. Corresponding to this value of k and the selected length scale ε and ω are calculated to be 910 m^2/s^3 and 1023 s^{-1} respectively from Equation 5.3 and 5.4.

$$k = \frac{3}{2}(Tu_\infty U)^2 \tag{5.2}$$

$$\varepsilon = c_\mu \frac{k^{1.5}}{l} \tag{5.3}$$

$$\omega = \frac{k^{0.5}}{l} \tag{5.4}$$

Increase in turbulent intensity shows a clear increase in Nusselt number over the entire section of the airfoil as shown in Figure 5.8. In Figure 5.7, Predictions for the kwSST model becomes better but for LS model higher over-prediction is observed. LS model over-predicts Nusselt number by 19.3 % on the pressure side and 56.6 % on the suction side as compared to the data. k-ω-SST model almost exactly matches the data on the pressure side upto s/C = -0.4 but then it diverges and the prediction becomes poorer than LS model with over-prediction of 21.2 % from the data near the trailing edge of pressure side. On the suction side after transition, k-ω-SST model performs better than its performance at lower turbulence intensity but it under-predicts the Nusselt number an average by 20% as compared to the data. On the other hand high over-prediction is observed for LS model on average of 35%. Near the transition point, data shows dip in Nusselt number but all the model shows opposite behavior with a peak in the value of Nusselt number with LS model performing the worst. Just as in the case of low turbulence level, LS model highly over-predicts Nusselt number at the stagnation point. So, the peak is replaced by a dashed line at the stagnation region. The trend of both the model prediction is the same. It seems that LS

model was already over-predicting TKE values in the passage and with the increase in initial TKE it further degrades the model performance. As expected, higher turbulence levels helps the prediction for the k-ω-SST model by improving the TKE values in the passage.

Both the model perform similar by over-predicting the velocity boundary layer thickness on the pressure side as can be observed in Figure 5.10b. Similar behavior was observed at PS1 and PS3 location as well. Boundary layer thickness was 120 % higher at PS1 but matched exactly to the data at PS3. At SS2, both the model highly over-predicts the boundary layer thickness with LS model performing worse than k-ω-SST model. Error for LS model was 300 %. Further downstream at SS3 and SS4, LS model performs better than k-ω-SST model to determine velocity profile but none of the model could exactly predict the correct profile. On pressure side, LS model showed an average error of 3 % to match the velocity profiles as compared to 6% error on the suction side. For k-ω-SST model this error was 8% on both pressure side and suction side.

At PS2, both the models perform equally to determine near wall thermal profile but both of them deviate away from the wall with LS model over-prediction of 35 % being worse than k-ω-SST. k-ω-SST model performs a good job to determine thermal profile before the transition at SS1 with LS model slightly under-performing. But both the model gave unsatisfactory results near the transition location SS2 with LS model over-predicting thermal boundary layer thickness by 100 %. All the models and codes are for fully turbulent flow and thus show incapability to predict transition. Temperature profile prediction improves for kwSST model at SS3 and SS4 but LS model could never catch up after transition. This could be due to the effect of adverse pressure gradient which LS model cannot account for properly. On average, both the models showed an error of 5% in matching the thermal profiles.

Prediction for TKE profiles improve for both the models at the pressure side location PS2 with correct trends near the wall. Here LS model slightly over-predicts free-stream TKE value but overall performs better than k-ω-SST model. None of the model could predict peak TKE value at PS3 which is consistent with the results from the flat plate. At SS2, both the models under-predicts near wall TKE value with k-ω-SST model being worse. But k-ω-SST model preserves the shape of the TKE profile predicting correct freestream values whereas LS model TKE profile looks like the velocity profile. Similar behavior is observed at SS3 with k-ω-SST model predictions improving as compared to the data whereas LS model now over-predicting the peak TKE value and overall TKE profile. Average error for LS model on pressure side was 12 % and for k-ω-SST model was 32 %. On suction side, LS model showed an error of 51 % whereas k-ω-SST model performed a little better with an error of 34 %.

Overall, the models did not perform well at at the suction side. They over-predicted the thermal boundary layer thickness. k-ω-SST model performed better at higher freestream turbulence which is strange since all the predictions were poor at high FST for flat plate but the results were consistent with LS model.

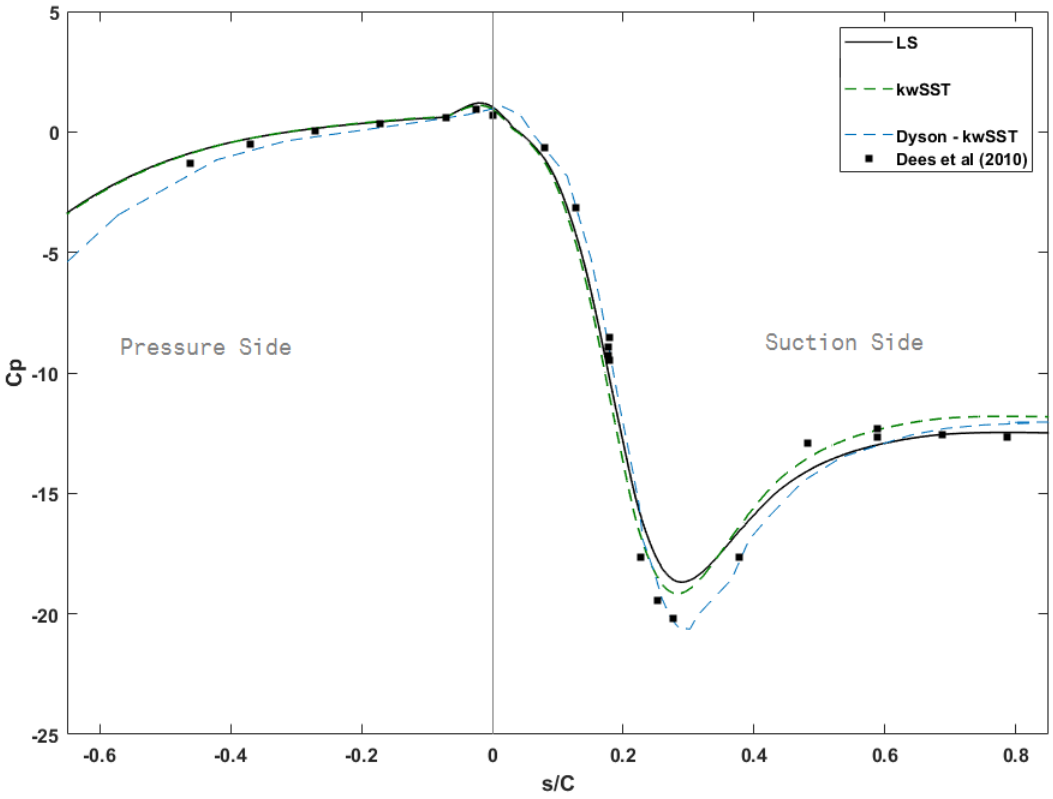
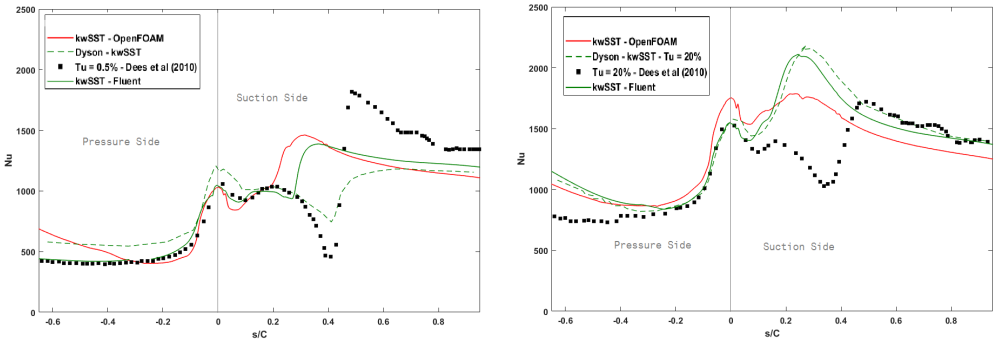


Figure 5.4: Comparison of coefficient of pressure for C3X vane with the experimental data of Dees et al [1] and CFD study of Dyson et al [2] in FLUENT



(a) Tu = 0.5 %

(b) Tu = 20 %

Figure 5.5: Nusselt number comparison for C3X vane cascade using k-ω-SST model in Fluent, OpenFOAM and CFD study by Dyson et al [2] - baseline results

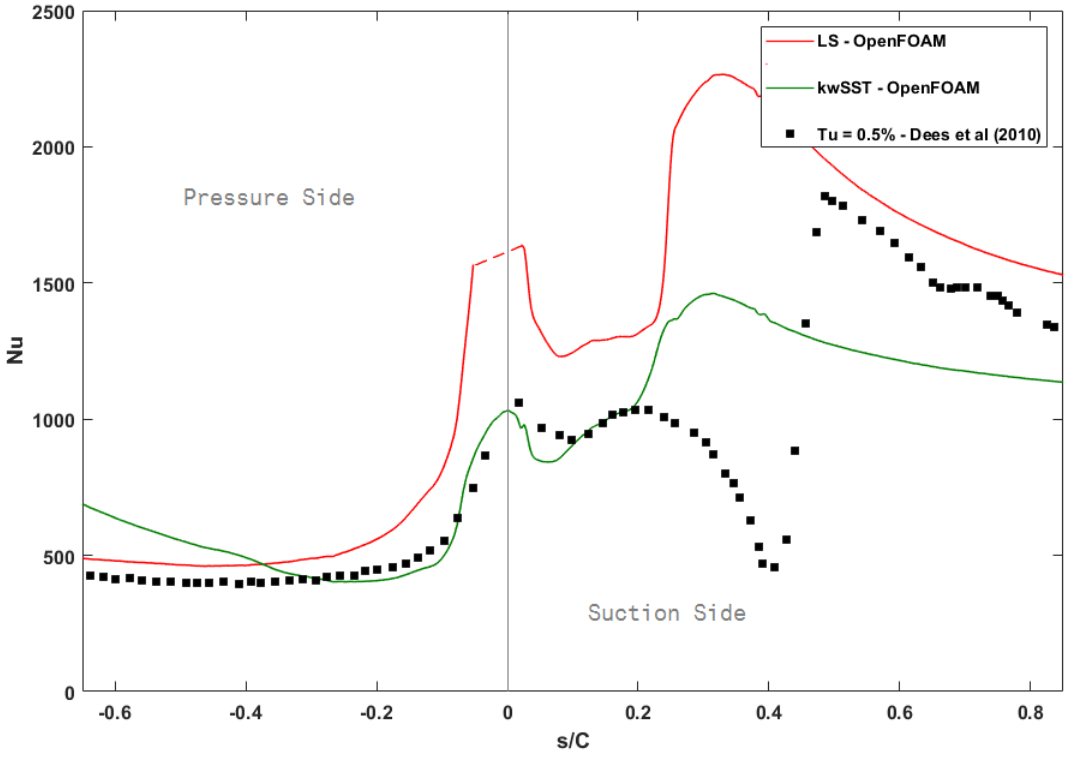


Figure 5.6: Comparison of Nusselt number at Tu = 0.5 % using LS and k-ω-SST model in OpenFOAM with the experimental data of Dees et al [1] - baseline results

Table 5.2: Velocity and thermal boundary layer thickness for flow over C3X vane at the measurement locations for low and high initial turbulent intensities

		Location (s/C)						
	Thickness (mm)	PS1	PS2	PS3	SS1	SS2	SS3	SS4
Tu = 0.5 %	δ	2.8	1.7	1.3	0.4	3.4	6.4	6.9
	δ_T	4.4	4.0	3.2	1.5	2.9	5.2	5.7
Tu = 20 %	δ	3.3	2.1	1.4	0.4	4.4	6.2	5.9
	δ_T	4.6	3.9	3.2	1.9	3.1	4.3	4.5

5.3 Implementation of High FST TKE Diffusion Model

Comparison of LS model at two different turbulence levels shows quite some deviations from the data with TKE profiles being poorly predicted in almost all the cases. Thus, it seems that the additional term in the TKE transport equation can improve TKE profiles near the wall by the model developed for high FST flow and thereby thermal and momentum field predictions since it did so in case of flat plate as was shown in Chapter-4.

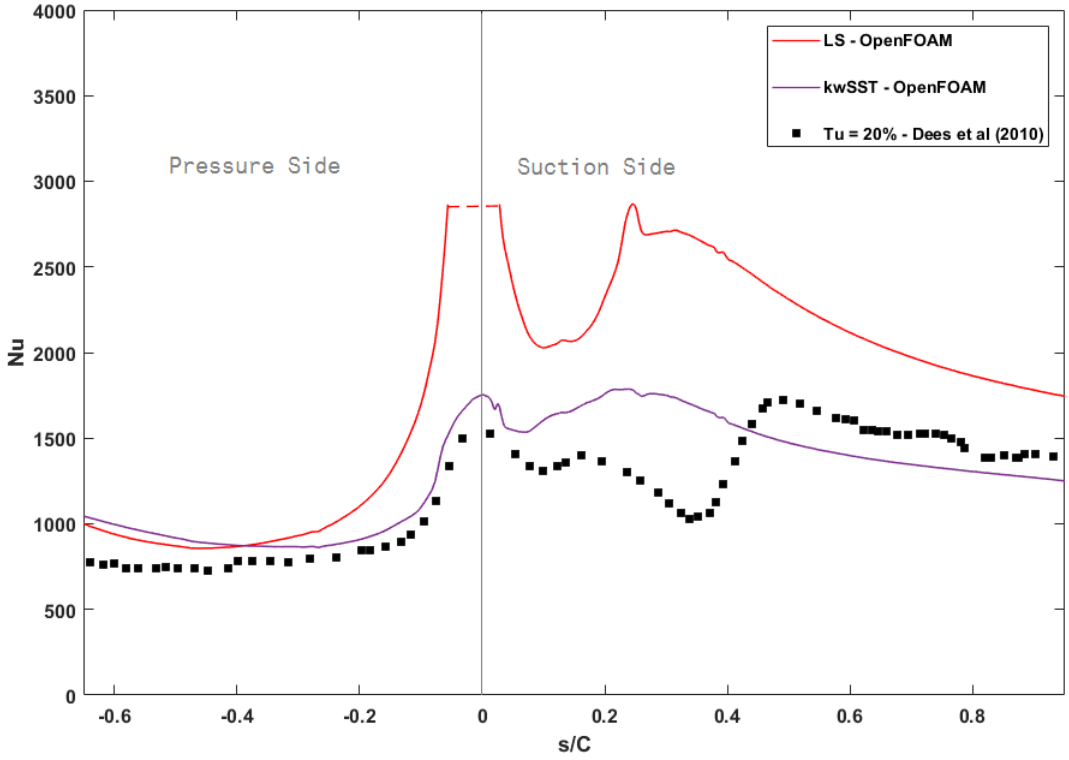


Figure 5.7: Comparison of Nusselt number at Tu = 20 % using LS and k-ω-SST model in OpenFOAM with the experimental data of Dees et al [1] - baseline results

Selection of free-stream in the case of vane cascade is not straight-forward as was in the case of flat plate since now the velocity is continuously changing across the passage. So, the mid-surface between the suction side and the pressure side is selected as the free-stream location which in the current case happens to be periodic boundary identified as *top-periodic*. But the concern is to which free-stream location is to be associated with any point on the surface since the shape of the boundary and the surface is not similar and there is no one-to-one relation, wall normal location or matching x-coordinate location is selected. On investigation, it was found that the changes in the TKE in negligible between these two location, so, to simply the implementation FST value corresponding to the matching x-coordinate was selected. Similar selection was made for the ε values. FST is required for the calculation c_μ, and both FST and free-stream ε is required to calculate the length scale. This implementation can be seen in the contour plot shown in figure 5.19. Here,it can be observed that there is significant generation of TKE in the vane passage for low initial FST which shows a decrease and then increase of c_μ along the passage. Whereas the trend for high initial FST is monotonic where c_μ increases along the passage in Figure 5.19b.

Another important aspect of the implementation is the boundary layer thickness which is required to calculate the constant C_{FST}. As discussed in the previous chapter, analytical value of δ is sufficient and also prevents excessively small values thus keeping a bound on C_{FST}. Here,

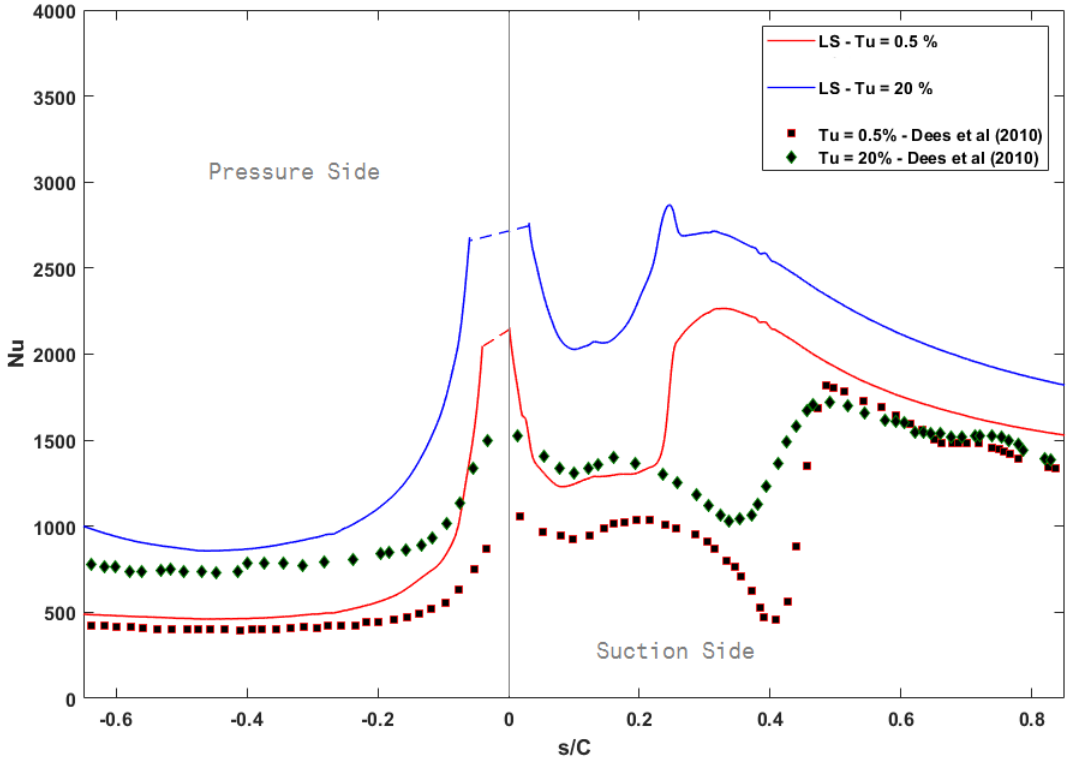


Figure 5.8: Nusselt number results over C3X vane using LS model in OpenFOAM for low and high FST - baseline results

flat plate turbulent boundary layer thickness equation was implemented on the surface but with replacement of the streamwise x-coordinate with distance, s, from the stagnation point along the vane surface for both pressure side as well as suction side.

5.4 Observations

Several trials for the selection of appropriate definition of FST and boundary layer thickness were conducted to select the best model for the curved surface. It was found that the free-stream turbulence should be defined based on the local velocity rather than the initial reference velocity since it is required for calculation of c_μ and affects the results considerably. To evaluate C_{FST}, definition of length scale was left the same as was for the case of turbulent flat plate. It was found necessary to put a bound on this C_{FST} since very large values produced unrealistic results and it was decided to keep it less than 0.1. This was achieved by the selection of boundary layer thickness definition such that nowhere in the passage, the length to boundary layer thickness ratio exceeded ~4. This makes sense from the experimental studies in the literature which limits this ratio to about 2 as was observed in the case of flat plate [7] [23].

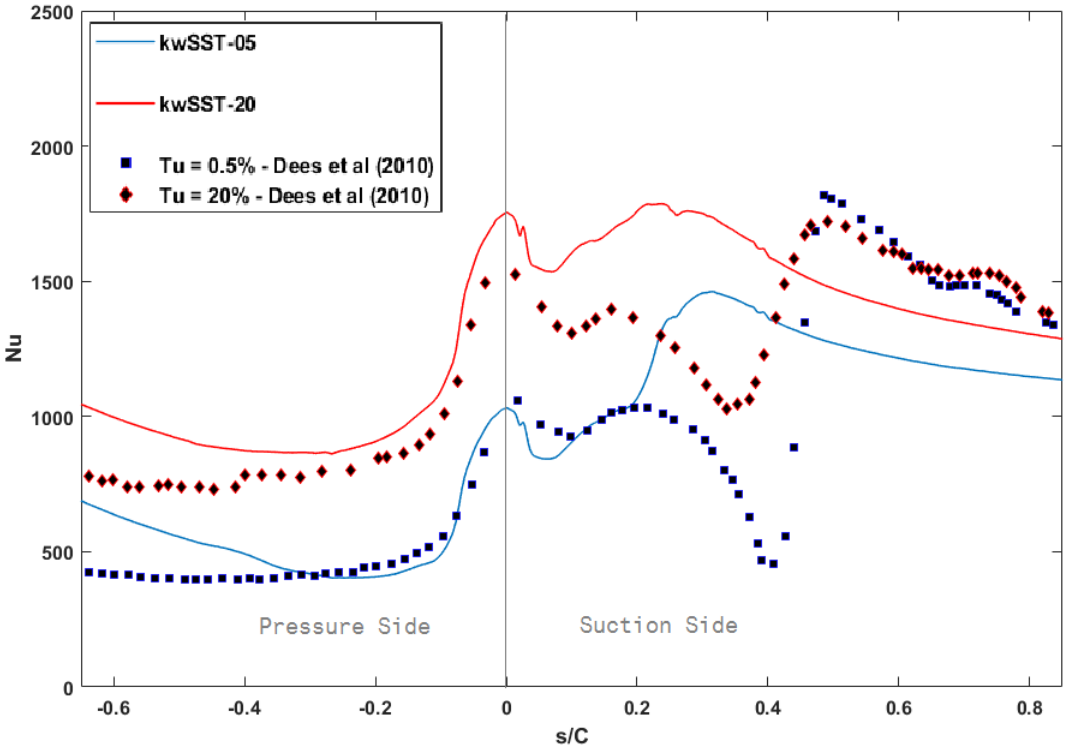
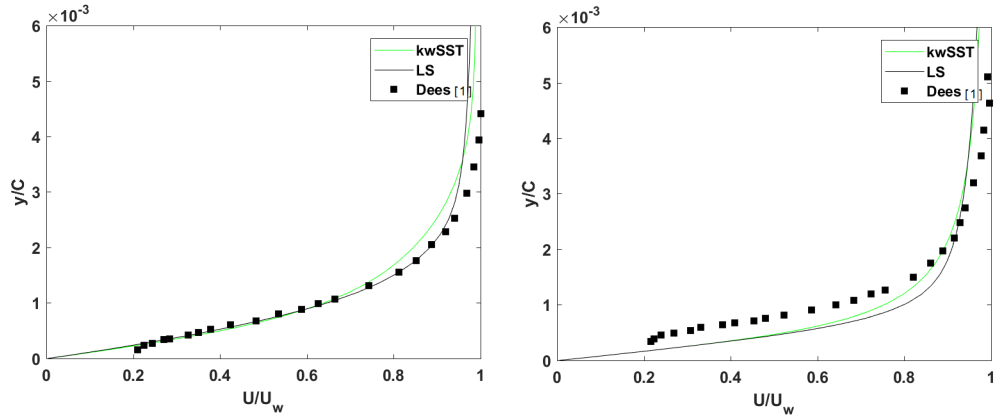


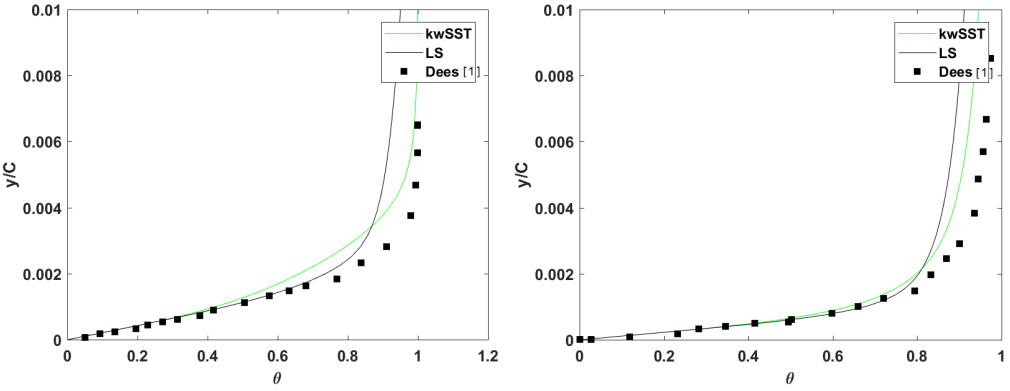
Figure 5.9: Nusselt number results over C3X vane using k-ω-SST model in OpenFOAM for low and high FST - baseline results

Results for the Nusselt number at low initial FST of 0.5 % shows better prediction with the new model. The prediction on the pressure side is almost same as that of the original model which is slightly over-predicted by 15 % but can be said as a good match with the data. The effect of the model can be observed on the suction side just before transition where better predictions are observed with an average error of only 11 %. There is slight increase in Nusselt number as compared to the original LS model after the dip but the results almost exactly match the data in the later portion on the suction side. The dip observed in Nusselt number can be found to be closer to the data but still it can be said that the separation is not fully captured. This can be explained by velocity, temperature and TKE profiles at the measurement location. The profiles match with the original LS model at the pressure side with an average error of just 6% but peak TKE is still predicted higher with turbulent intensity of 8% which is less than the LS model. Velocity and thermal profiles at SS2 show a better match with the data near the wall and away from the wall with slight mismatch in between the region. An average error of 14 % with standard deviation in the error being 6 % is observed on suction side but with excellent match near the wall. Thermal profile at SS3 exactly matches the data points which represents good prediction of Nusselt number at suction side. Thermal profile shows an average error of just 6 % from the data. The improvement of near wall velocity and thermal profile is the result the improvement in TKE profiles. The peak value of TKE with the new model comes closer to the



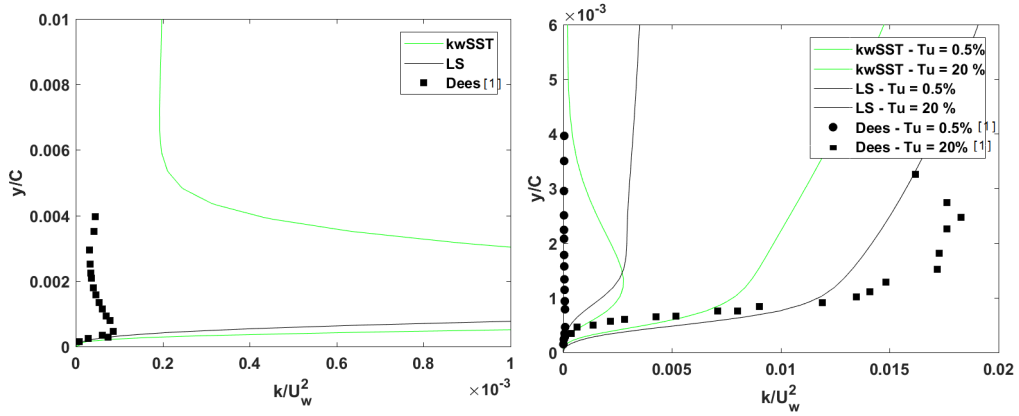
(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.10: Normalized velocity profiles at the pressure side location PS2 - baseline results



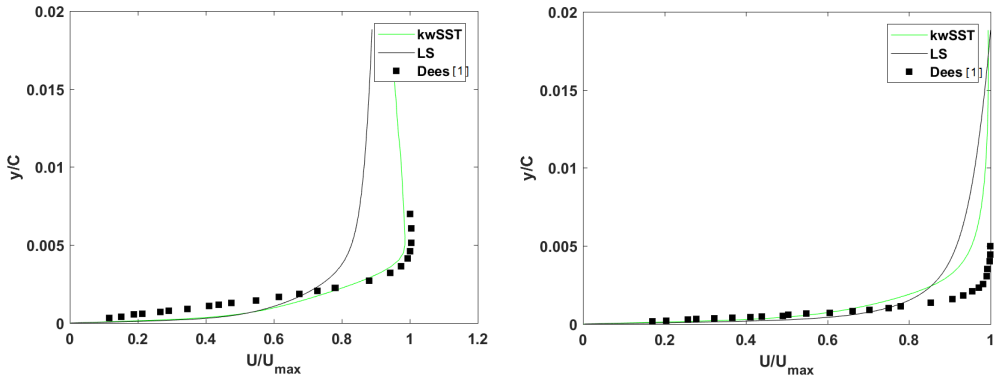
(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.11: Non-dimensional thermal profiles at the pressure side location PS2 - baseline results



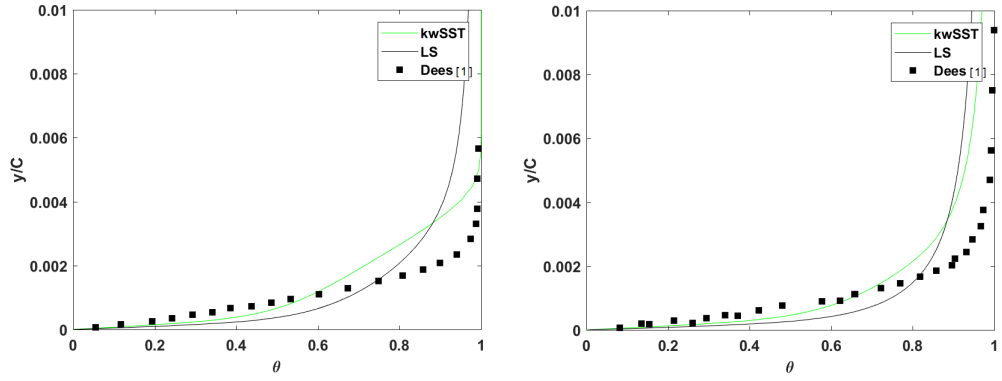
(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.12: TKE profiles at the pressure side location PS2 - baseline results



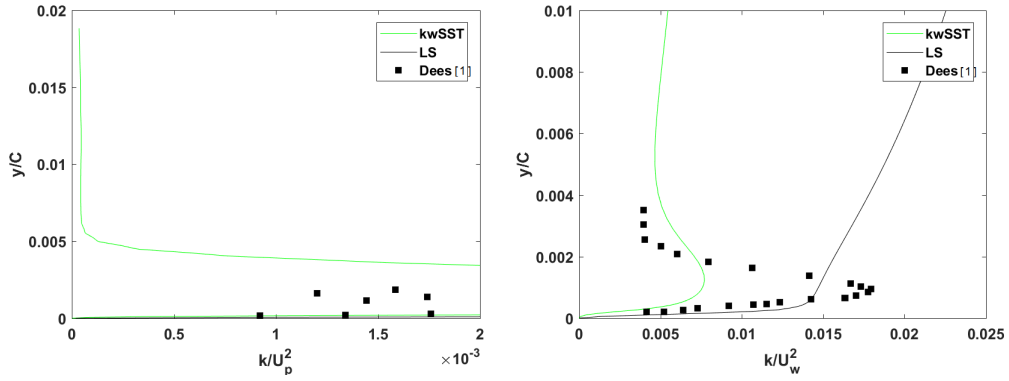
(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.13: Normalized velocity profiles at the suction side location SS2 - baseline results



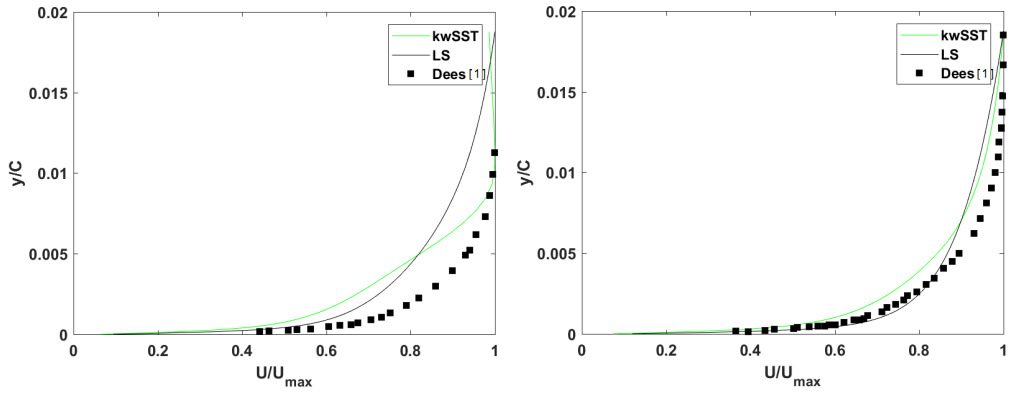
(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.14: Non-dimensional thermal profiles at the suction side location SS2 - baseline results



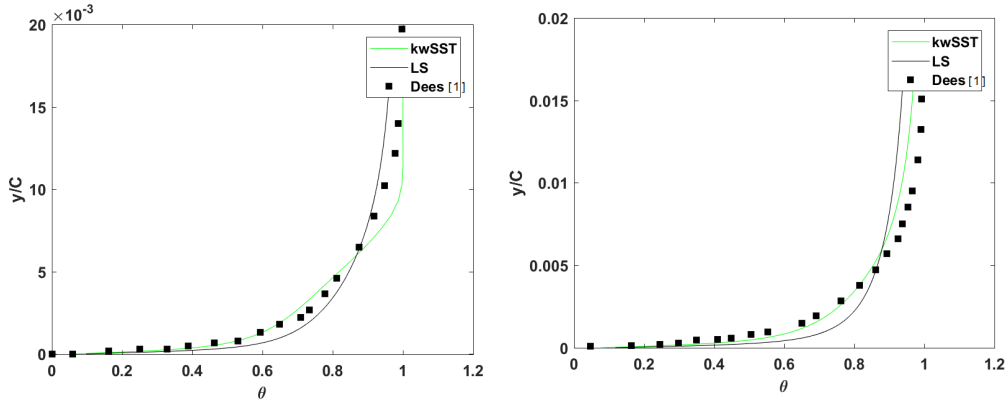
(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.15: TKE profiles at the suction side location SS2 - baseline results



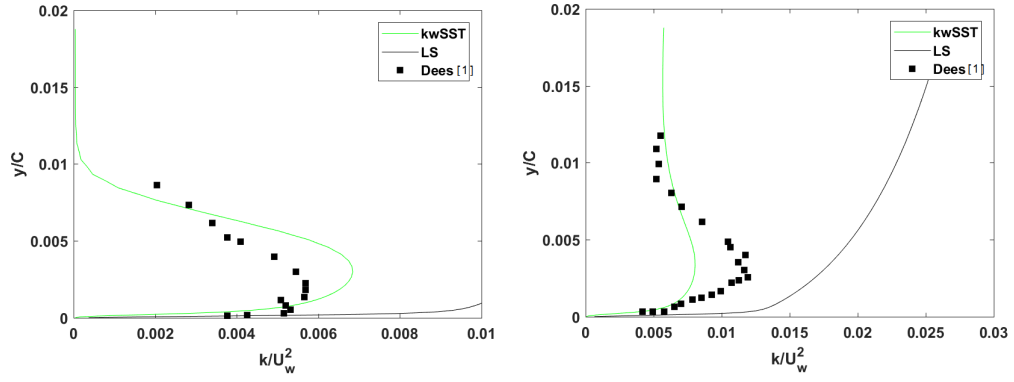
(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.16: Normalized velocity profiles at the suction side location SS3 - baseline results



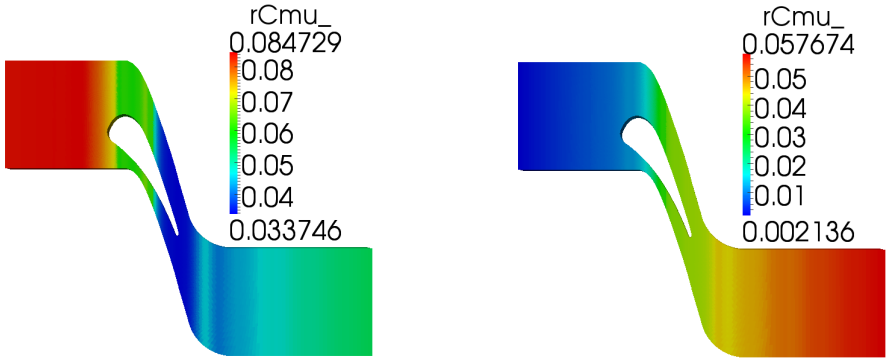
(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.17: Non-dimensional thermal profiles at the suction side location SS3 - baseline results



(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.18: TKE profiles at the suction side location SS3 - baseline results



(a) Tu = 0.5 %

(b) Tu = 20 %

Figure 5.19: Contour plots of c_μ based on free-stream turbulence in the turbine vane cascade

data as compared to LS model but still with high over-prediction in near wall TKE. This shows the effect of the new model to reduce the excessive production of TKE in the passage resulting in better predictions.

Real effect of the model can be observed when comparing the Nusselt number at high initial freestream turbulence level of 20 % as shown in Figure 5.21. Excellent match with the data is observed on the pressure side with an error of just 8% from the data. This improvement can also be observed in Figures 5.22b, 5.23b and 5.24b where the profiles show closer match with the data. Velocity profile at this location is predicted with an average error of 7 % whereas thermal profile has an error of 12 %. The error reduces to just 4% downstream on the pressure side. Significant improvement in TKE profile is observed which is predicted with an error of 13 % but with much better profiles as compared to the LS model. Excellent improvement on the suction side results for Nusselt number after separation is observed where the results almost exactly match the data with an error of 11 % in contrast to the LS model which highly over-predicts Nusselt number by 56.5 % average error. This results is also accompanied by significant improvement of velocity, thermal and TKE profiles. TKE profile in Figure 5.30b almost exactly match with the data with only little under-prediction for the peak TKE at an error of just 6 %. New model outperforms LS model by capturing the correct trend on the Nusselt number at the beginning region of suction side showing a clear dip in the results. This effect was not captured by LS model which instead showed an opposite behavior by producing a peak value of Nusselt number. Though the results are much better but these are slightly under-predicted as compared to the data. The profiles of velocity and temperature at SS2 exactly matches the data near the wall and away from the wall with only little difference in the mid-region. Average error in TKE profiles for the new model is just 26 % as compared to 51.5% for LS model. One region where the new model fails just as the original LS model is the stagnation point where Nusselt number is highly over-predicted. This region was clipped as explained in the previous section.

Overall, high FST TKE diffusion model performs better than baseline LS model in all aspects.

It shows promising results for the real turbine situation of high FST by showing an improvement of over 40 % to predict Nusselt number along with the correct prediction of the trend where the baseline models fail completely.

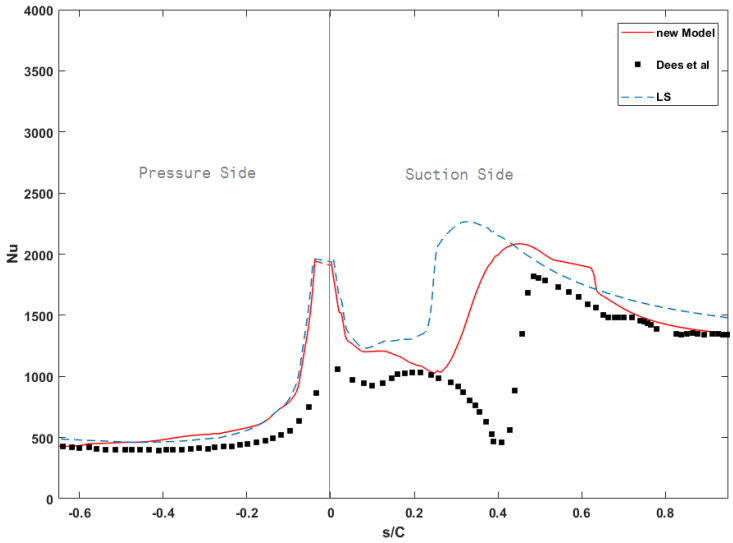


Figure 5.20: Nusselt number results for the high FST TKE diffusion model implementation in OpenFOAM for low initial turbulent intensity of $Tu_i = 0.5\%$

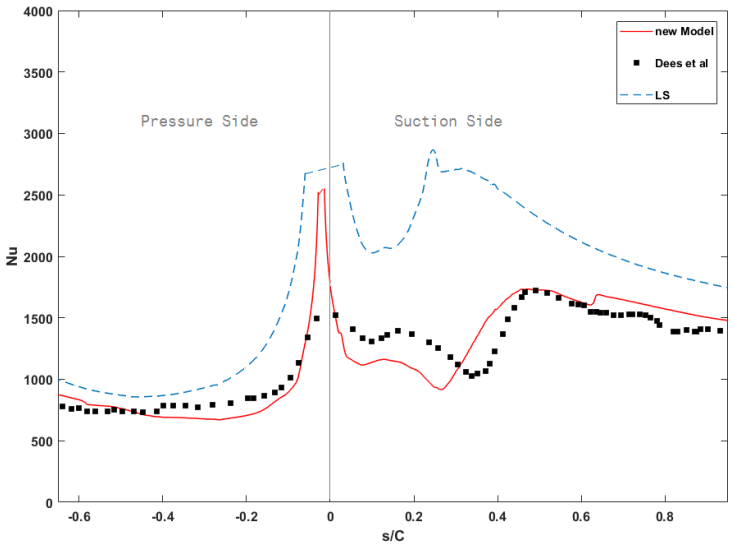
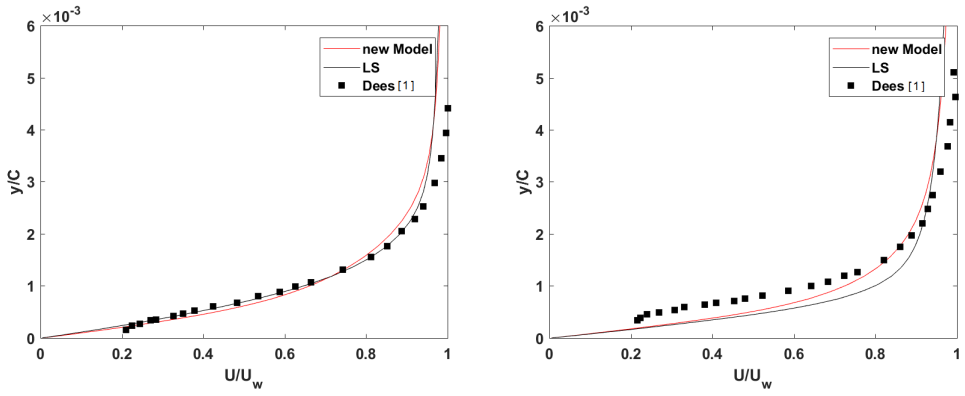
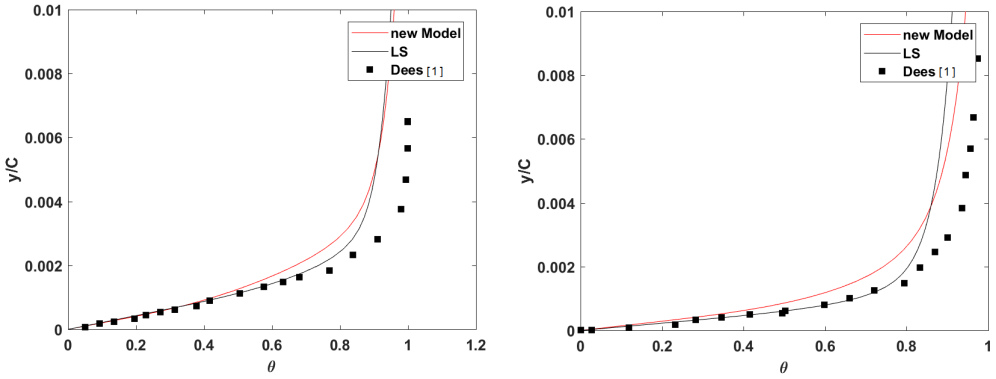


Figure 5.21: Nusselt number results for the high FST TKE diffusion model implementation in OpenFOAM for high initial turbulent intensity of $Tu_i = 20\%$



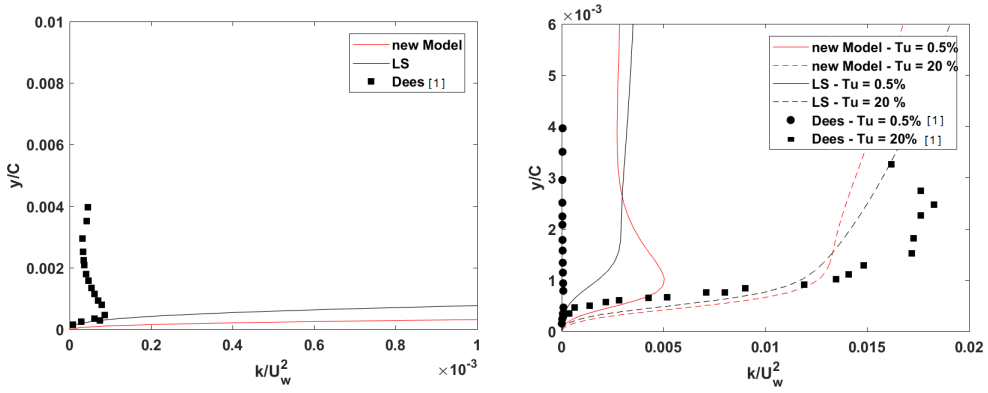
(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.22: Normalized velocity profiles at the pressure side location PS2 for new model



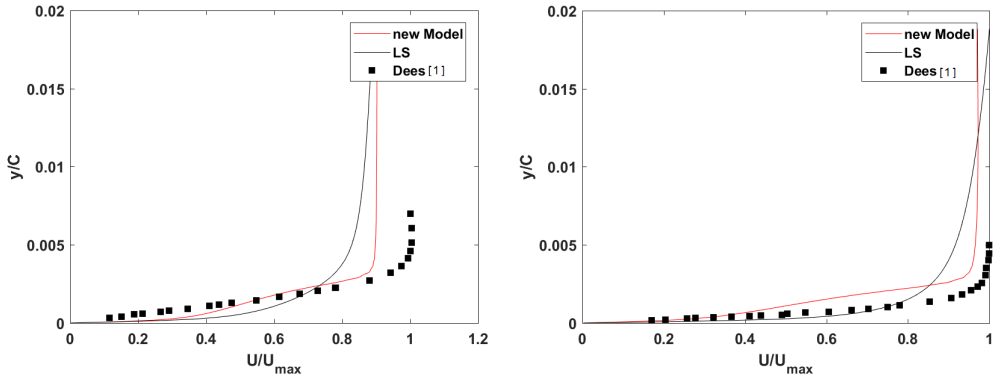
(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.23: Non-dimentional thermal profiles at the pressure side location PS2 for new model



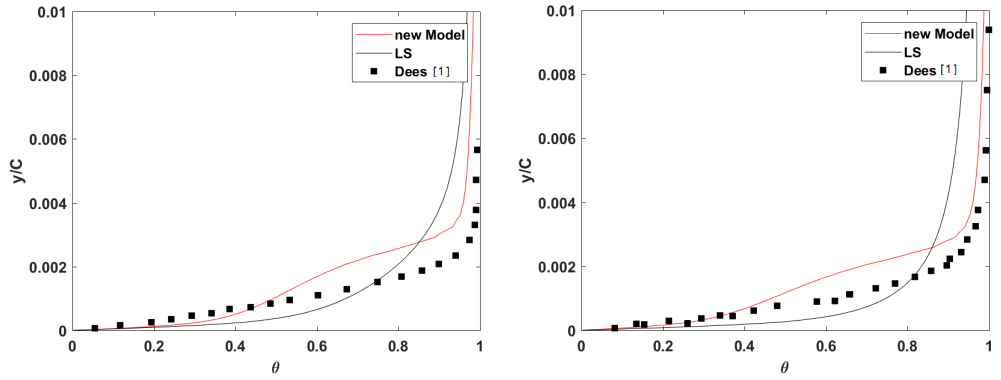
(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.24: TKE profiles at the pressure side location PS2 for new model



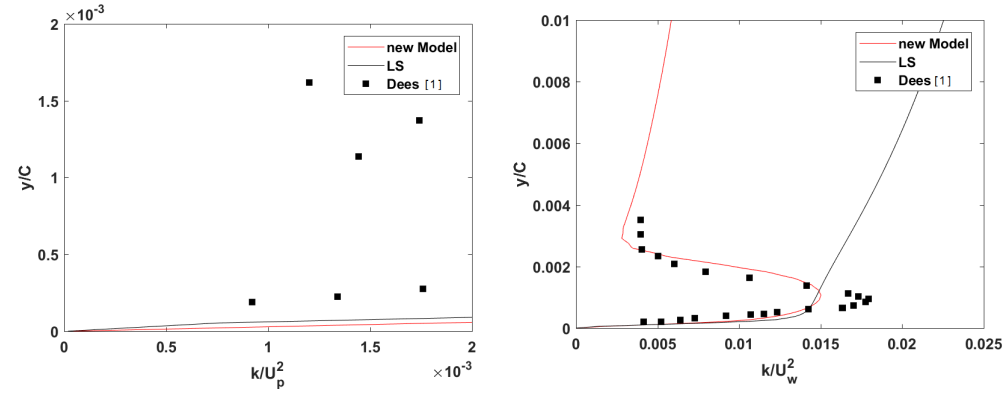
(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.25: Normalized velocity profiles at the suction side location SS2 for new model



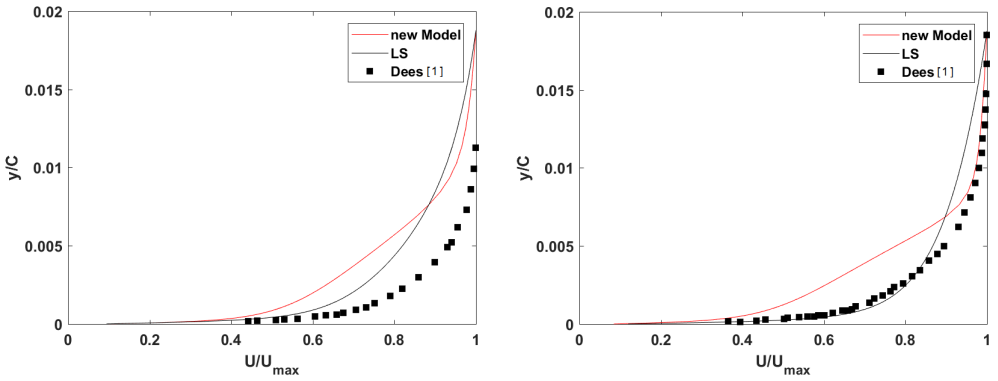
(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.26: Non-dimensional thermal profiles at the suction side location SS2 for new model



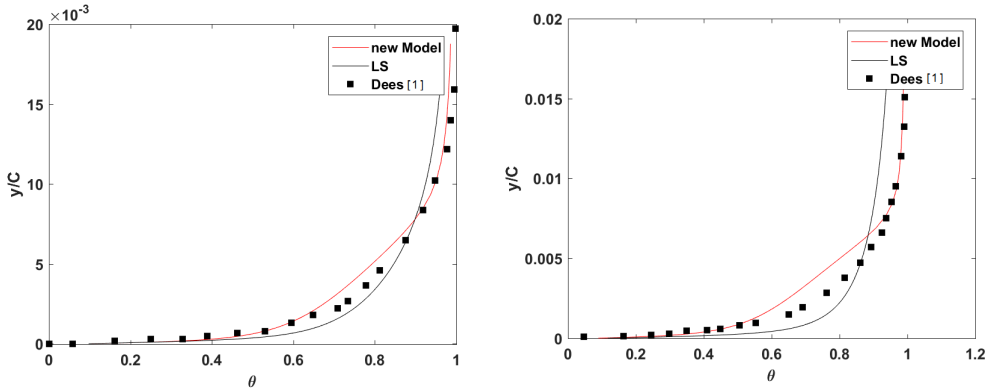
(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.27: TKE profiles at the suction side location SS2 for new model



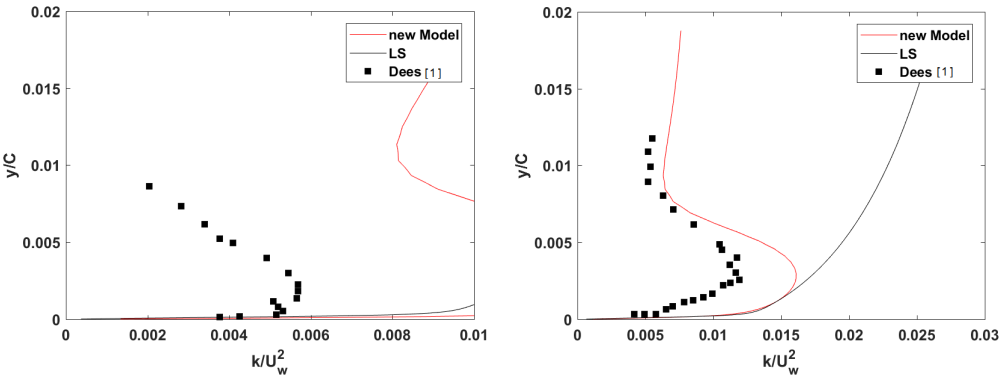
(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.28: Normalized velocity profiles at the suction side location SS3 for new model



(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.29: Non-dimensional thermal profiles at the suction side location SS3 for new model



(a) Tu = 0.5 % (b) Tu = 20 %

Figure 5.30: TKE profiles at the suction side location SS3 for new model. Clear improvement for both near wall and far away TKE values is observed. The real life case of high FST shows very good agreement with the data.

Chapter 6
Summary and Conclusions

Overall performance of the high FST TKE diffusion model was much better showing significant improvements over the original LS model and k-ω-SST model. This improvement was due to the correct prediction of TKE profiles by reducing the excessive production in the vane passage. For the flat plate, peak TKE value needed an increase compared to the baseline LS model whereas for the vane cascade a reduction was required. The new model was able to adapt to both these conditions normalizing the values of TKE near the wall. Correct prediction of TKE ensures correct prediction of surface heat flux and shear stress since turbulent viscosity calculation is better compared to the case with no correction. The effect of variable c_μ is also important to reduce the calculated value of turbulent viscosity by adapting to the real situation of the free-stream turbulence Very close to the wall. Viscous damping reduces the tangential velocity fluctuations while kinematic blocking reduces the normal fluctuations. Towards the outer part of the near-wall region, however, turbulence is rapidly augmented by the production of turbulent kinetic energy due to the large gradients in mean velocity.

6.1 Contribution of the present research

Thus, the contribution of the current study can be concluded as:

- Comprehensive comparison of the results on a turbulent flat plate for various low Reynolds number models along with detailed boundary and initial conditions
- Pointing out the mistake in the Fluent's implementation of Launder-Sharma model and testing UDF implemented LS model on turbulent flat plate to show better prediction comparable to other low Reynolds number models
- Studying the implementation of variable C_μ for the LRN models of Fluent and proving its inability to convert the constant c_μ to a local variable
- Implementation of an additional term in kinetic energy transport equation in OpenFOAM along with conversion of constant c_μ to a local variable to account for the effect of free-

stream turbulence. High FST TKE diffusion model showed 20 %, 15 % and 38 % improvement in Stanton number, skin friction coefficient and near wall peak turbulent kinetic energy respectively.

- Comparison of RANS models in Fluent and OpenFOAM to predict heat transfer from a stationary turbine vane cascade by resolving the viscous sub-layer highlights the failure of baseline models to correctly predict heat transfer from turbine blade.
- Implementation of the high FST TKE diffusion model for the curved surface which showed improvement of 11 % on the pressure side and 45 % on the suction side as compared to the original $k - \varepsilon$ model.

6.2 Future studies

The new model shows promising results in all sense but nonetheless it lacks in certain aspects of real life applications. The following recommendation for the future research can be made:

- The current study is limited to only few data sets to calculate variable c_μ. New studies in this regards can produce better results therefore appropriate data for c_μ is needed under high FST.
- The model could not be implemented on the Fluent's original model. But it can still be implemented by writing complete UDF for some low Reynolds number model (LS, AKN etc) including the transport equation for k and ε, turbulent viscosity definition and variable c_μ.
- Nusselt number shows a peak at the stagnation point with the value of approximately 7000 as compared to the data value of 1500 for the high FST case. Appropriate changes in the model can limit the high gradients to prevent such a behavior.
- The current implementation of the model prevents the usage of parallel computing for calculation since traversing throughout the domain is required. Current implementation requires traversing through the entire free-stream surface for every data point in the domain. Thus decomposed domain require communication and data transfer from other decomposed domains during run-time. The normal decomposition of the entire domain fails without provision of special commands for communication. So, currently the new model implementation is solved on a single core. Communication links for mesh decomposition can be established to enable parallel computing with the high FST TKE diffusion model.
- The model is implemented as a 3-D model but has been tested for 2-D geometries. 3D results may produce comprehensive report on the performance for its ability to capture complex phenomena such as secondary flow and Taylor-Gortler vortices so extensive 3-D data would be needed for the comparison.

Bibliography

[1] Dees, J. E., D. G. Bogard, G. A. Ledezma, G. M. Laskowski, and A. K. Tolpadi (2012) "Experimental measurements and computational predictions for an internally cooled simulated turbine vane," *Journal of Turbomachinery*, **134**(6), p. 061003.

[2] Dyson, T. E., D. G. Bogard, and S. D. Bradshaw (2013) "A CFD evaluation of multiple RANS turbulence models for prediction of boundary layer flows on a turbine vane," in *ASME Turbo Expo 2013: Turbine Technical Conference and Exposition*, American Society of Mechanical Engineers, pp. V03CT14A014–V03CT14A014.

[3] Yavuzkurt, S. and G. R. Iyer (2002) "A model for diffusion of turbulent kinetic energy for calculation of momentum and heat transfer in high FST flows," in *ASME Turbo Expo 2002: Power for Land, Sea, and Air*, American Society of Mechanical Engineers, pp. 419–427.

[4] Han, J.-C., S. Dutta, and S. Ekkad (2012) *Gas turbine heat transfer and cooling technology*, CRC Press.

[5] Powel, S. (1990) "On the leading edge: combining maturity and advanced technology on the F404 turbofan engine," in *ASME 1990 International Gas Turbine and Aeroengine Congress and Exposition*, American Society of Mechanical Engineers, pp. V002T02A006–V002T02A006.

[6] Hancock, P. E. and P. Bradshaw (1980) *The effect of free-stream turbulence on turbulent boundary layers*, Ph.D. thesis, University of London.

[7] Blair, M. and M. Werle (1980) *The Influence of Free-Stream Turbulence on the Zero Pressure Gradient Fully Turbulent Boundary Layer.*, *Tech. rep.*, DTIC Document.

[8] Sugawara, S., T. Sato, H. Komatsu, and H. Osaka (1958) "The effect of free-stream turbulence on heat transfer from a flat plate," .

[9] Simonich, J. and P. Bradshaw (1978) "Effect of free-stream turbulence on heat transfer through a turbulent boundary layer," *ASME, Transactions, Journal of Heat Transfer*, **100**, pp. 671–677.

[10] Hancock, P. and P. Bradshaw (1983) "The effect of free-stream turbulence on turbulent boundary layers," *ASME Journal of Fluids Engineering*, **105**, pp. 284–289.

[11] Blair, M., D. Edwards, and R. Dring (1982) *The Effects of Free-Stream Turbulence on the Turbulence Structure and Heat Transfer in Zero Pressure Gradient Boundary Layers.*, *Tech. rep.*, DTIC Document.

[12] MacMullin, R., W. Elrod, and R. Rivir (1989) "Free-stream turbulence from a circular wall jet on a flat plate heat transfer and boundary layer flow," *Journal of Turbomachinery*, **111**(1), pp. 78–86.

[13] Yavuzkurt, S. and K. Batchelder (1993) "A correlation for heat transfer under high free stream turbulence conditions," in *Proceedings of the Ninth Symposium on Turbulent Shear Flows, Kyoto, Japan, August*, pp. 16–18.

[14] Dullenkopf, K. and R. Mayle (1994) "An account of free-stream-turbulence length scale on laminar heat transfer," in *ASME 1994 International Gas Turbine and Aeroengine Congress and Exposition*, American Society of Mechanical Engineers, pp. V004T09A026–V004T09A026.

[15] Thole, K. and D. Bogard (1996) "High freestream turbulence effects on turbulent boundary layers," *Transactions-ASME Journal of Fluids Engineering*, **118**, pp. 276–284.

[16] Barrett, M. J. and D. K. Hollingsworth (2003) "Heat Transfer in Turbulent Boundary Layers Subjected to Free-Stream TurbulenceâĂŤPart I: Experimental Results," *Journal of turbomachinery*, **125**(2), pp. 232–241.

[17] Gibson, M. M., C. A. Verriopoulos, and N. S. Vlachos (1984) "Turbulent boundary layer on a mildly curved convex surface," *Experiments in Fluids*, **2**(1), pp. 17–24.
URL `https://doi.org/10.1007/BF00266314`

[18] Schultz, M. P. and R. J. Volino (2001) "Effects of concave curvature on boundary layer transition under high free-stream turbulence conditions," in *ASME Turbo Expo 2001: Power for Land, Sea, and Air*, American Society of Mechanical Engineers, pp. V003T01A065–V003T01A065.

[19] Stefes, B. and H.-H. Fernholz (2004) "Skin friction and turbulence measurements in a boundary layer with zero-pressure-gradient under the influence of high intensity free-stream turbulence," *European Journal of Mechanics-B/Fluids*, **23**(2), pp. 303–318.

[20] Nix, A., T. Diller, and W. Ng (2007) "Experimental measurements and modeling of the effects of large-scale freestream turbulence on heat transfer," *Journal of turbomachinery*, **129**(3), pp. 542–550.

[21] Hylton, L., M. Mihelc, E. Turner, D. Nealy, and R. York (1983) "Analytical and experimental evaluation of the heat transfer distribution over the surfaces of turbine vanes," .

[22] Ames, F. E. (1995) "The influence of large scale high intensity turbulence on vane heat transfer," in *ASME 1995 International Gas Turbine and Aeroengine Congress and Exposition*, American Society of Mechanical Engineers, pp. V004T09A021–V004T09A021.

[23] Ames, F. and R. Moffat (1990) "Effects of simulated combustor turbulence on boundary layer heat transfer," *In: Heat transfer in turbulent flow (A93-54620 24-34), p. 11-17.*, pp. 11–17.

[24] Chen, S., J. Lai, J. Milthorpe, and N. Mudford (1998) "A new modified low-Reynolds-number k-epsilon model," in *29th AIAA, Fluid Dynamics Conference*, p. 2553.

[25] Hoda, A. and S. Acharya (1999) "Predictions of a film coolant jet in crossflow with different turbulence models," in *ASME 1999 International Gas Turbine and Aeroengine Congress and Exhibition*, American Society of Mechanical Engineers, pp. V003T01A028–V003T01A028.

[26] Phillips, W., D. F. Hunsaker, and R. Spall (2010) "Smooth-Wall Boundary Conditions for Dissipation-Based Turbulence Models," *AIAA Paper*, **1103**, p. 2010.

[27] Henkes, R. and C. Hoogendoorn (1995) "Comparison exercise for computations of turbulent natural convection in enclosures," *Numerical Heat Transfer, Part B Fundamentals*, **28**(1), pp. 59–78.

[28] Kwon, O. and F. E. Ames (1995) "Advanced k-epsilon Modeling of Heat Transfer," .

[29] Iyer, G. R. and S. Yavuzkurt (1999) "Comparison of low Reynolds number k–ε models in simulation of momentum and heat transport under high free stream turbulence," *International journal of heat and mass transfer*, **42**(4), pp. 723–737.

[30] Aldemir, H. O. and S. Yavuzkurt (2006) "Implementation of a Free Stream Turbulence Diffusion Model in FLUENT Code for Calculation of Heat and Momentum Transport in a Flat Plate Boundary Layer," in *ASME Turbo Expo 2006: Power for Land, Sea, and Air*, American Society of Mechanical Engineers, pp. 297–305.

[31] Foroutan, H. and S. Yavuzkurt (2013) "A Model for Simulation of Turbulent Flow With High Free Stream Turbulence Implemented in OpenFOAM®," *Journal of Turbomachinery*, **135**(3), p. 031022.

[32] Mathur, A. and S. He (2013) "Performance and implementation of the Launder–Sharma low-Reynolds number turbulence model," *Computers & Fluids*, **79**, pp. 134–139.

[33] Luo, J., E. H. Razinsky, and H.-K. Moon (2013) "Three-Dimensional RANS Prediction of Gas-Side Heat Transfer Coefficients on Turbine Blade and Endwall," *Journal of Turbomachinery*, **135**(2), p. 021005.

[34] Balogh, M., A. Parente, and C. Benocci (2012) "RANS simulation of ABL flow over complex terrains applying an Enhanced k-ε model and wall function formulation: Implementation and comparison for fluent and OpenFOAM," *Journal of wind engineering and industrial aerodynamics*, **104**, pp. 360–368.

[35] Cotas, C. I. P. (2015) *Modelling of Fiber Suspensions Flow in Pipes*.

[36] Furbo, E. (2010) *Evaluation of RANS turbulence models for flow problems with signigicant impact of boundary layers*.

[37] García Unzue, J. (2011) "CFD analysis on the blades of an axial gas turbine," .

[38] Jones, D. A., M. Chapuis, M. Liefvendahl, D. Norrison, and R. Widjaja (2016) *RANS Simulations using OpenFOAM Software*, *Tech. rep.*, Defence Science and Technology Group Fishermans Bend Victoria Australia.

[39] Anderson, M. R. and D. L. Bonhaus (2014) "Validation Results For A Diverse Set Of Turbomachinery Cases Using A Density-Based OpenFOAM® Solver," in *ASME Turbo Expo 2014: Turbine Technical Conference and Exposition*, American Society of Mechanical Engineers, pp. V02BT39A020–V02BT39A020.

[40] Mangani, L., M. Cerutti, M. Maritano, and M. Spel (2010) "Conjugate heat transfer analysis of NASA C3X film cooled vane with an object-oriented CFD code," *ASME Paper No. GT2010-23458*, pp. 1805–1814.

[41] Mangani, L., C. Bianchini, A. Andreini, and B. Facchini (2007) "Development and validation of a C++ object oriented CFD code for heat transfer analysis," *ASME Summer Heat Transfer*.

[42] Laskowski, G. M., A. K. Tolpadi, and M. C. Ostrowski (2007) "Heat transfer predictions of film cooled stationary turbine airfoils," *ASME GT2007-27497*.

[43] Erney, R. W. (2009) "Verification and validation of single phase and cavitating flows using an open source CFD tool," .

[44] Jones, D. A. (2015) *CFD RANS Simulations on a Generic Conventional Scale Model Submarine: Comparison between Fluent and OpenFOAM*, *Tech. rep.*, DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION FISHERMANS BEND (AUSTRALIA) MARITIME DIV.

[45] Launder, B. E. and D. B. Spalding (1974) "The numerical computation of turbulent flows," *Computer methods in applied mechanics and engineering*, **3**(2), pp. 269–289.

[46] Shih, T.-H., W. W. Liou, A. Shabbir, Z. Yang, and J. Zhu (1995) "A new k-Ït eddy viscosity model for high reynolds number turbulent flows," *Computers & Fluids*, **24**(3), pp. 227–238.

[47] Wilcox, D. (1993) "A two-equation turbulence model for wall-bounded and free-shear flows," in *1993 AIAA 24 th Fluid Dynamics Conference*.

[48] Iyer, G. R. (1998) *Prediction of effect of high free stream turbulence on momentum transport and heat transfer in a flat plate turbulent boundary layer*.

[49] 2012, S. G. I. C., "OpenFOAM: The Open Source Computational Fluid Dynamics (CFD) Toolbox," .
URL `http://www.openfoam.com`

[50] Spalart, P. and S. Allmaras (1992) "A one-equation turbulence model for aerodynamic flows," in *30th aerospace sciences meeting and exhibit*, p. 439.

[51] Launder, B. E. and D. B. Spalding (1972) *Mathematical models of turbulence*, Academic press.

[52] Fluent, A. (2009) "12.0 Theory Guide," *Ansys Inc*, **5**.

[53] Patel, V. C., W. Rodi, and G. Scheuerer (1985) "Turbulence models for near-wall and low Reynolds number flows- A review," *AIAA journal*, **23**(9), pp. 1308–1319.

[54] Menter, F. R. et al. (1994) "Two-equation eddy-viscosity turbulence models for engineering applications," *AIAA journal*, **32**(8), pp. 1598–1605.

[55] Caretto, L., A. Gosman, S. Patankar, and D. Spalding (1973) "Two calculation procedures for steady, three-dimensional flows with recirculation," in *Proceedings of the third international conference on numerical methods in fluid mechanics*, Springer, pp. 60–68.

[56] Guide, A. F. U. (2011) "Release 14.0, ANSYS," *Inc., USA, November*.

[57] Hirsch, C. (2007) *Numerical computation of internal and external flows: The fundamentals of computational fluid dynamics*, Butterworth-Heinemann.

[58] Vaughn Jr, M. E. (2008) *Guidelines for Gridding Simple Flows-The Flat Plate in Laminar Flow*, *Tech. rep.*, DTIC Document.

[59] Foroutan, H. and S. Yavuzkurt (2015) "Numerical Simulations of the Near-Field Region of Film Cooling Jets Under High Free Stream Turbulence: Application of RANS and Hybrid URANS/Large Eddy Simulation Models," *Journal of Heat Transfer*, **137**(1), p. 011701.

[60] Kays, W. M. (2012) *Convective heat and mass transfer*, Tata McGraw-Hill Education.

[61] Abe, K., T. Kondoh, and Y. Nagano (1994) "A new turbulence model for predicting fluid flow and heat transfer in separating and reattaching flowsâĂŤI. Flow field calculations," *International journal of heat and mass transfer*, **37**(1), pp. 139–151.

[62] Pointwise, R. (2015), "Release 17, Pointwise User Manual, Pointwise," .

[63] Chien, K.-Y. (1982) "Predictions of channel and boundary-layer flows with a low-Reynolds-number turbulence model," *AIAA journal*, **20**(1), pp. 33–38.

[64] Wieghardt, K. and W. Tillmann (1951) "On the turbulent friction layer for rising pressure," .

[65] Klebanoff, P. (1955) "Characteristics of turbulence in boundary layer with zero pressure gradient," .

Appendix A
Fluent UDF

A.1 Launder and Sharma Model

This is the correct implementation of Launder-Sharma model based on Mathur and He [32] modified for any geometry.

Libraries and constants

```
#include "udf.h"
#include "mem.h"
#define SIG_K 1.0
#define SIG_D 1.3
#define C1_D 1.44
#define C2_D 1.92
#define MU_L 1.7894e-5
#define DEN 1.225
```

Variable and function definitions

```
enum { TKE, TDR, RK, SRM, N_REQUIRED_UDS};
enum { DE, MUT, RET, N_REQUIRED_UDM};

real C_MU(cell_t c, Thread *t) { return 0.09;}

real f_mu(cell_t c, Thread *t) { return exp(-3.4/SQR(1+C_UDMI(c,t,RET)/50)); }

real f_1(cell_t c, Thread *t) { real a = 1.0; return a; }

real f_2(cell_t c, Thread *t)
{ return (1.-0.3*exp(-C_UDMI(c,t,RET)*C_UDMI(c,t,RET))); }
```

TKE equation

```
DEFINE_SOURCE(k_src1, c, t, dS, eqn)
{
C_UDSI(c,t,SRM) = C_STRAIN_RATE_MAG(c,t);
C_UDSI(c,t,RK) = sqrt(C_UDSI(c,t,TKE));
dS[eqn] =
    -2.*C_R(c,t)*C_R(c,t)*C_MU(c,t)*f_mu(c,t)*C_UDSI(c,t,TKE)/C_UDMI(c,t,MUT);
return
    -C_R(c,t)*C_R(c,t)*C_MU(c,t)*f_mu(c,t)*SQR(C_UDSI(c,t,TKE))/C_UDMI(c,t,MUT);
}

DEFINE_SOURCE(k_src2, c, t, dS, eqn)
{
real G_k;
C_UDMI(c,t,DE) = 2.*C_MU_L(c,t)*NV_MAG2(C_UDSI_G(c,t,RK));
G_k = C_UDMI(c,t,MUT)*SQR(C_STRAIN_RATE_MAG(c,t));
dS[eqn] = 0;
return G_k - C_UDMI(c,t,DE);
}
```

Epsilon equation

```
DEFINE_SOURCE(d_src, c, t, dS, eqn)
{
real G_d, E_d;
G_d =
    C1_D*f_1(c,t)*C_UDMI(c,t,MUT)*SQR(C_STRAIN_RATE_MAG(c,t))/C_UDSI(c,t,TKE);
E_d = 2.*C_UDMI(c,t,MUT)*C_MU_L(c,t)*(NV_MAG2(C_UDSI_G(c,t,SRM)))/C_R(c,t);
dS[eqn] = G_d - 2.*C2_D*f_2(c,t)*C_R(c,t)* C_UDSI(c,t,TDR)/C_UDSI(c,t,TKE);
return G_d*(C_UDSI(c,t,TDR)) - C2_D*f_2(c,t)*C_R(c,t)*SQR(C_UDSI(c,t,TDR))/C_UDSI(c,t,TKE) + E_
}

DEFINE_DIFFUSIVITY(ke_diff, c, t, eqn)
{
switch(eqn)
{ case TKE: return C_UDMI(c,t,MUT)/SIG_K + C_MU_L(c,t); break;
  case TDR: return C_UDMI(c,t,MUT)/SIG_D + C_MU_L(c,t); break;
          default: return C_UDMI(c,t,MUT) + C_MU_L(c,t);
}
}
```

Definition of turbulent viscosity

```
DEFINE_TURBULENT_VISCOSITY(turb_vis, c, t)
```

```
{ return C_UDMI(c,t,MUT); }

DEFINE_ADJUST(adj_func,d)
{
Thread *t;
cell_t c;
thread_loop_c(t,d)
{
begin_c_loop(c,t)
{
C_UDMI(c,t,MUT) =
            C_R(c,t)*C_MU(c,t)*f_mu(c,t)*SQR(C_UDSI(c,t,TKE))/C_UDSI(c,t,TDR);
C_UDMI(c,t,RET) = C_R(c,t)*SQR(C_UDSI(c,t,TKE))/(C_MU_L(c,t)*C_UDSI(c,t,TDR));
C_UDSI(c,t,SRM) = C_STRAIN_RATE_MAG(c,t);
C_UDSI(c,t,RK) = sqrt(C_UDSI(c,t,TKE));
}
end_c_loop(c,t)
}
}

DEFINE_INIT(init_func_lowTU,d)
{   /* this is random */
cell_t c;
real MVEL = 30.48;
Thread *t;
real temp,tempa,tempb,ustar,yplus,a,b,RE,x[ND_ND], DH;
thread_loop_c(t,d)
{
begin_c_loop(c,t)
{
C_CENTROID(x,c,t);
RE = MVEL * x[0]*DEN/MU_L;
DH = 2.0*0.38*x[0]*pow(RE,-0.2);
temp = 0.027*pow(RE,-1./7)*DEN*MVEL*MVEL;
ustar = sqrt(temp/DEN);
a = 2.*pow(DEN/MU_L,2)*pow(ustar,5)*DH/2;
b = 4.734*pow(ustar,3)/DH;
yplus = C_WALL_DIST(c,t)*ustar* DEN/MU_L;
if(yplus < 17.5) temp = 1.5+1.5*sin(3.141/17.5*(yplus-8.75));
else temp = 3;
temp = temp*ustar*ustar;
```

```
C_UDSI(c,t,TKE) = 3.56;
C_UDSI(c,t,RK) = sqrt(C_UDSI(c,t,TKE));
temp = 2.*C_WALL_DIST(c,t)/DH;
tempa=a*temp;
tempb=b/temp;
if (tempa < tempb) temp = tempa;
else temp = tempb;
C_UDSI(c,t,TDR) = 70;
C_UDMI(c,t,MUT) = DEN*C_MU(c,t)*f_mu(c,t)*SQR(C_UDSI(c,t,TKE))/(C_UDSI(c,t,TDR));
C_UDMI(c,t,RET) = DEN*SQR(C_UDSI(c,t,TKE))/MU_L/(C_UDSI(c,t,TDR));
}
end_c_loop(c,t)
}
}

DEFINE_PRANDTL_T(Prt, c, t)
{   /* variable Prandtl number */
real C, Pet, Pr, Prtinf, Prt;
C = 0.2;
Prtinf = 0.85;
Pr = 0.74;
Pet = Pr*C_MU_T(c, t)/C_MU_L(c, t);
Prt = 1.0/( (0.5/Prtinf) + C*Pet/sqrt(Prtinf) - C*C*Pet*Pet*(1.0 - exp(-1/(C*Pet*sqrt(Prtinf)))
return Prt;
}
```

A.2 High FST TKE diffusion model implementation

This implementation include extraction of free-stream value of TKE for the flat plate and add an extra diffusion term to TKE equation.

```
#include "udf.h"
#include "mem.h"
#define SIG_K 1.0
#define SIG_D 1.3
#define C1_D 1.44
#define C2_D 1.92
#define MU_L 1.7894e-5
#define DEN 1.225
#define MVEL 30.48
#define IN 6.53
#define DELTA 0.00615
```

```
enum { TKE, TDR, RK, SRM, KFST, N_REQUIRED_UDS};
enum { DE, MUT, RET, xTKE, xTDR, CFST, N_REQUIRED_UDM};
```

Extract the freestream TKE value

```
void FSTcalc(real xLoc)
{
Domain *d = Get_Domain(1);
real x[ND_ND], k, xPos, yPos, pos, delta;
pos = 0;
cell_t c;
Thread *t;
k = 0.;
thread_loop_c(t,d)
{
begin_c_loop(c,t)
{
C_CENTROID(x,c,t);
xPos = x[0];
yPos = x[1];
if (xPos == xLoc && yPos > 0.3)
{
C_UDMI(c,t,xTKE) = C_UDSI(c,t,TKE);
C_UDMI(c,t,xTDR) = C_UDSI(c,t,TDR);
delta = 0.38*xLoc*pow(DEN*MVEL*xLoc/MU_L,-0.2);
C_UDMI(c,t,CFST) =
    0.076*pow(pow(C_UDMI(c,t,xTKE),1.5)/(C_UDMI(c,t,xTDR)*delta),0.208);
pos = 1;
break;
}
if (pos==1)
break;
}
end_c_loop(c,t)
}
}
```

Definition of variable c_μ

```
real C_MU(cell_t c, Thread *t)
{
real x[ND_ND], xLoc, CMU, k, Tu;
```

```
C_CENTROID(x,c,t);
xLoc = x[0];
k = C_UDMI(c,t,xTKE);
Tu = 100*sqrt(2.*k/3)/MVEL;
CMU = 0.082 - 0.024*log(Tu);
if (CMU>0.09)
CMU = 0.09;
if (CMU<0.0024)
CMU = 0.0024;
return CMU;
}

real f_mu(cell_t c, Thread *t)
{ return exp(-3.4/SQR(1+C_UDMI(c,t,RET)/50)); }

real f_1(cell_t c, Thread *t)
{ real a = 1.0; return a; }

real f_2(cell_t c, Thread *t)
{ return (1.-0.3*exp(-C_UDMI(c,t,RET)*C_UDMI(c,t,RET))); }

DEFINE_SOURCE(k_src1, c, t, dS, eqn)
{
C_UDSI(c,t,SRM) = C_STRAIN_RATE_MAG(c,t);
C_UDSI(c,t,RK) = sqrt(C_UDSI(c,t,TKE));
dS[eqn] =
    -2.*C_R(c,t)*C_R(c,t)*C_MU(c,t)*f_mu(c,t)*C_UDSI(c,t,TKE)/C_UDMI(c,t,MUT);
return
    -C_R(c,t)*C_R(c,t)*C_MU(c,t)*f_mu(c,t)*SQR(C_UDSI(c,t,TKE))/C_UDMI(c,t,MUT);
}

DEFINE_SOURCE(k_src2, c, t, dS, eqn)
{
real G_k;
C_UDMI(c,t,DE) = 2.*C_MU_L(c,t)*NV_MAG2(C_UDSI_G(c,t,RK));
G_k = C_UDMI(c,t,MUT)*SQR(C_STRAIN_RATE_MAG(c,t));
dS[eqn] = 0;
return G_k - C_UDMI(c,t,DE);
}
```

Additional source term

```
DEFINE_SOURCE(k_hosein2, c, t, dS, eqn)
```

```
{
real h_tke1, h_tke2;
C_UDSI(c,t,KFST) =  C_UDMI(c,t,CFST)*C_UDSI(c,t,MUT)*C_U_G(c,t)[1];
h_tke1 = C_UDSI(c,t,KFST)*C_U_G(c,t)[1];
h_tke2 = C_U(c,t)*C_UDSI_G(c,t,KFST)[1];
dS[eqn] = 0;
return h_tke1 + h_tke2;
}

DEFINE_SOURCE(d_src, c, t, dS, eqn)
{
real G_d, E_d;
G_d =
    C1_D*f_1(c,t)*C_UDMI(c,t,MUT)*SQR(C_STRAIN_RATE_MAG(c,t))/C_UDSI(c,t,TKE);
E_d = 2.*C_UDMI(c,t,MUT)*C_MU_L(c,t)*(NV_MAG2(C_UDSI_G(c,t,SRM)))/C_R(c,t);
dS[eqn] = G_d - 2.*C2_D*f_2(c,t)*C_R(c,t)* C_UDSI(c,t,TDR)/C_UDSI(c,t,TKE);
return G_d*(C_UDSI(c,t,TDR)) - C2_D*f_2(c,t)*C_R(c,t)*SQR(C_UDSI(c,t,TDR))/C_UDSI(c,t,TKE) + E_
}

DEFINE_DIFFUSIVITY(ke_diff, c, t, eqn)
{
switch(eqn)
{ case TKE: return C_UDMI(c,t,MUT)/SIG_K + C_MU_L(c,t); break;
  case TDR: return C_UDMI(c,t,MUT)/SIG_D + C_MU_L(c,t); break;
          default: return C_UDMI(c,t,MUT) + C_MU_L(c,t);
}
}

DEFINE_TURBULENT_VISCOSITY(turb_vis, c, t)
{ return C_UDMI(c,t,MUT); }
```

Adjusts the values of the variables after every iteration

```
DEFINE_ADJUST(adj_func,d)
{
Thread *t;
cell_t c;
real x[ND_ND], xLoc;
thread_loop_c(t,d)
{
begin_c_loop(c,t)
{
C_CENTROID(x,c,t);
```

```
xLoc = x[0];
C_UDMI(c,t,MUT) =
            C_R(c,t)*C_MU(c,t)*f_mu(c,t)*SQR(C_UDSI(c,t,TKE))/C_UDSI(c,t,TDR);
C_UDMI(c,t,RET) = C_R(c,t)*SQR(C_UDSI(c,t,TKE))/C_MU_L(c,t)/(C_UDSI(c,t,TDR));
C_UDSI(c,t,SRM) = C_STRAIN_RATE_MAG(c,t);
C_UDSI(c,t,RK) = sqrt(C_UDSI(c,t,TKE));
FSTcalc(xLoc);
C_UDSI(c,t,KFST) =  C_UDMI(c,t,CFST)*C_UDSI(c,t,MUT)*(C_U_G(c,t)[1]);
}
end_c_loop(c,t)
}
}
```

Initialization function for the new variables

```
DEFINE_INIT(init_func,d)
{
cell_t c;
Thread *t;
real temp,tempa,tempb,ustar,yplus,a,b,RE,x[ND_ND], DH;
thread_loop_c(t,d)
{
begin_c_loop(c,t)
{
C_CENTROID(x,c,t);
RE = MVEL * x[0]*DEN/MU_L;
DH = 2.0*0.38*x[0]*pow(RE,-0.2);
temp = 0.027*pow(RE,-1./7)*DEN*MVEL*MVEL;
ustar = sqrt(temp/DEN);
a = 2.*pow(DEN/MU_L,2)*pow(ustar,5)*DH/2;
b = 4.734*pow(ustar,3)/DH;
yplus = C_WALL_DIST(c,t)*ustar* DEN/MU_L;
C_UDSI(c,t,TKE) = 5.94;
C_UDSI(c,t,RK) = sqrt(C_UDSI(c,t,TKE));
C_UDSI(c,t,TDR) = 10;
C_UDMI(c,t,MUT) =
            DEN*C_MU(c,t)*f_mu(c,t)*SQR(C_UDSI(c,t,TKE))/(C_UDSI(c,t,TDR));
C_UDMI(c,t,RET) = DEN*SQR(C_UDSI(c,t,TKE))/MU_L/(C_UDSI(c,t,TDR));
C_UDMI(c,t,xTKE) = C_UDSI(c,t,TKE);
C_UDMI(c,t,xTDR) = C_UDSI(c,t,TDR);
C_UDMI(c,t,CFST) =
            0.076*pow(pow(C_UDSI(c,t,TKE),1.5)/(C_UDSI(c,t,TDR)*DELTA),0.208);
C_UDSI(c,t,KFST) = 1;
```

```
}
end_c_loop(c,t)
}
}
```

A.3 Variable Prandtl number

Based on Kays emprical correlation

```
DEFINE_PRANDTL_T(Prt, c, t)
{
real C, Pet, Pr, Prtinf, Prt;
C = 0.2;
Prtinf = 0.85;
Pr = 0.74;
Pet = Pr*C_MU_T(c, t)/C_MU_L(c, t);
Prt = 1.0/( (0.5/Prtinf) + C*Pet/sqrt(Prtinf) - C*C*Pet*Pet*(1.0 - exp(-1/(C*Pet*sqrt(Prtinf)))
return Prt;
}
```

Appendix B
OpenFOAM codes

B.1 Addition of Diffusion Term to Launder-Sharma Model

The new model is implemented by making a copy of existing LS model and modifying it. The file name and the function names should be changed to prevent over-writing of LS model in the library.

High FST TKE diffusion term in the TKE transport equation

```
tmp<fvScalarMatrix> kEqn
(
    fvm::ddt(k_)
  + fvm::div(phi_, k_)
  - fvm::laplacian(DkEff(), k_)
 ==
    G - fvm::Sp((epsilonTilda_ + D)/k_, k_) - fvc::div(CFST*(R()&U_))
);
```

Definition of new model constant

```
volScalarField fullLaunderSharmaKE::CFSTcalc
(
const volScalarField& length,
const volScalarField& delta
)const
{
return 0.076*pow(length/delta,0.208);
}
```

Calculation of variable c_μ

```
// Variable Cmu calculation
volScalarField fullLaunderSharmaKE::rCmu
```

```
(
    volScalarField& FST,
    volScalarField& rCmu,
    volScalarField& Feps
)
{
    label topID = 2; //check for particular geometry
    label inletID = 0; //check for particular geometry
    scalar x1, x2, dx;
    dimensionedScalar Ucell("Ucell",U_.dimensions(),1.0); //dimension of velocity
    const dimensionedScalar Ustart = Ucell*max(mag(U_.boundaryField()[inletID]));
//free-stream velocity

    volScalarField Tu //turbulent intensity
    (
IOobject
(
"Tu",
   runTime_.timeName(),
                mesh_,
                IOobject::NO_READ,
                IOobject::NO_WRITE
         ),
         sqrt((2.0/3.0)*k_)*100.0/Ustart,
         zeroGradientFvPatchField<scalar>::typeName
    );

    const fvPatchScalarField& topTKE(k_.boundaryField()[topID]); //TKE at free-stream
    const fvPatchScalarField& topEPS(epsilonTilda_.boundaryField()[topID]); //espilon of free-s

    forAll(rCmu.internalField(),cellI) //looping for internal cells
    {
x1 = mesh_.C()[cellI].x();
forAll(topTKE, faceI)
{
x2 = mesh_.Cf().boundaryField()[topID][faceI].x();
dx = fabs(x1-x2);
if (dx<scalar(1e-07))
{
FST[cellI] = topTKE[faceI];
Feps[cellI] = topEPS[faceI];
```

```
break;
}
}
Tu[cellI] = (100.0)*sqrt((2.0/3.0)*FST[cellI])/Ustart.value();
if(Tu[cellI]<scalar(8.0))
rCmu[cellI] = scalar(0.0837) - (0.061)*log10(Tu[cellI]);
else
rCmu[cellI] = (0.7)*pow(Tu[cellI], -1.538);
    }
    return rCmu;
}
```

Function to calculate boundary layer thickness

```
volScalarField fullLaunderSharmaKE::calcDelta
(
volScalarField& del
)
{
label inletID = 0;
     scalar x1, y2; //x2, dx
     dimensionedScalar Ucell("Ucell",U_.dimensions(),1.0);
dimensionedScalar dU("dU",U_.dimensions(),0.015);
//dimensionedScalar xDim("xDim",dimensionSet(0,1,0,0,0,0,0),1.0);
const dimensionedScalar Ustart = Ucell*max(mag(U_.boundaryField()[inletID]));
forAll(del.internalField(),cellI)
{
x1 = mesh_.C()[cellI].x();

if(x1<scalar(0.024))
{
del[cellI] = scalar(1e-3);
}
else
{ y2 = 0.38*x1*mag(pow(1.225*mag(x1)*Ustart.value()/1.7894e-5, -0.2)); //analytical function on
/*
forAll(del.internalField(),iter)
{
x2 = mesh_.C()[iter].x();
dx = fabs(x2 - x1);
Ucell = Ucell*mag(U_[iter][0]);
//Ucell = Ucell*mag(U_[iter][0] - 0.995*Ustart);
if( (dx < scalar(1e-7)) )//&& (Ucell < dU) )
```

```
{
del[cellI] = mesh_.C()[iter].y();
break;
}
}*/
del[cellI] = y2;
    }
}
return del;
}
```

B.2 Modification for curved surface

Selection of FST values

FST is calculated at the periodic boundary and the local velocity is considered to calculate turbulent intensity. This function is a sub-part of the function that calculates variable c_μ

```
volScalarField LSfullCurved3::rCmu
(
    volScalarField& FST,
    volScalarField& rCmu,
    volScalarField& Feps
)
{
    label topID = 14; //periodic bottom //check for particular geometry
    label inletID = 0; //check for particular geometry
    label outletID = 1;
    //label PSID = [2,3,4,5,6];
    //label SSID = [7,8,9,10,11];
    label bottomID = 13;
    scalar x1, x2, dx;
    dimensionedScalar Ucell("Ucell",U_.dimensions(),1.0); //dimension of velocity
    const dimensionedScalar Ustart = Ucell*max(mag(U_.boundaryField()[inletID]));
//free-stream velocity

    volScalarField Tu //turbulent intensity
    (
IOobject
(
"Tu",
   runTime_.timeName(),
               mesh_,
```

```
                IOobject::NO_READ,
                IOobject::NO_WRITE
            ),
            sqrt((2.0/3.0)*k_)*100.0/Ustart,
            zeroGradientFvPatchField<scalar>::typeName
        );
        volScalarField FU //free-stream speed
        (
IOobject
(
"FU",
   runTime_.timeName(),
                mesh_,
                IOobject::NO_READ,
                IOobject::NO_WRITE
            ),
            mag(U_),
            zeroGradientFvPatchField<scalar>::typeName
        );

        const fvPatchScalarField& topTKE(k_.boundaryField()[topID]); //TKE at free-stream
        const fvPatchScalarField& topEPS(epsilonTilda_.boundaryField()[topID]); //espilon of free-s
        const fvPatchScalarField& topU(FU.boundaryField()[topID]); //velocity of free-stream

        forAll(rCmu.internalField(),cellI) //looping for internal cells
        {
x1 = mesh_.C()[cellI].x();
forAll(topTKE, faceI)
{
x2 = mesh_.Cf().boundaryField()[topID][faceI].x();
dx = fabs(x1-x2);
if (dx<scalar(1e-2))
{
FST[cellI] = topTKE[faceI];
Feps[cellI] = topEPS[faceI];
  FU[cellI] = topU[faceI];
break;
}
   }
Tu[cellI] = (100.0)*sqrt((2.0/3.0)*FST[cellI])/FU[cellI];
if(Tu[cellI]<scalar(8.0))
```

```
rCmu[cellI] = scalar(0.0837) - (0.061)*log10(Tu[cellI]);
else
rCmu[cellI] = (0.7)*pow(Tu[cellI], -1.538);
    }
    return rCmu;
}
```

Boundary layer thickness

The definition of the boundary layer thickness given analytically and can be varied to limit the value of the model constant to keep the results in bound.

```
volScalarField LSfullCurved3::calcDelta
(
volScalarField& del
)
{
label inletID = 0;
label outletID = 1;
//label PSID = [2,3,4,5,6];
//label SSID = [7,8,9,10,11];
label bottomID = 13;
label topID = 14;
     scalar x1, y1, y2, originValue, sSS, sPS; //x2, dx
     dimensionedScalar Ucell("Ucell",U_.dimensions(),1.0);
dimensionedScalar dU("dU",U_.dimensions(),0.015);
//dimensionedScalar xDim("xDim",dimensionSet(0,1,0,0,0,0,0),1.0);
const dimensionedScalar Ustart = Ucell*max(mag(U_.boundaryField()[inletID]));

forAll(del.internalField(),cellI)
{
x1 = mesh_.C()[cellI].x();
y1 = mesh_.C()[cellI].y();
//originValue = y1 + 1.46762*x1 - 0.431635; //mid-line
originValue = y1 + 2.2953*x1 - 0.6810;
if(x1<scalar(1.2197e-4)) // before the vane
{
del[cellI] = 5*(1.2197e-4)*mag(pow(1.225*(1.2197e-4)*Ustart.value()/1.7894e-5, -0.5));;
}
else if (x1>scalar(0.3033)) //after the vane
{
del[cellI] = 5*0.5471*mag(pow(1.225*0.5471*Ustart.value()/1.7894e-5, -0.5));;
}
else //around vane cascasde
```

```
{
if(originValue < scalar(0) && y1<0.4325) // PRESSURE SIDE
{
sPS = 2.3332*x1*mag(x1) + 1.0825*x1 + 0.014;
del[cellI] = 5*sPS*mag(pow(1.225*mag(sPS)*Ustart.value()/1.7894e-5, -0.5));
}
else // SUCTION SIDE
{

sSS = 20.2006*x1*pow(mag(x1),2) - 4.45996*x1*mag(x1) + 1.80812*x1;
if(sSS < 0.2119)
del[cellI] = 5*sSS*mag(pow(1.225*mag(sSS)*Ustart.value()/1.7894e-5, -0.5));
else // Separation
del[cellI] = 5*0.2119*mag(pow(1.225*mag(0.2119)*Ustart.value()/1.7894e-5, -0.5));
}
}
}
return del;
}
```

B.3 New Utility for Calculation of Temperature Gradient at the Wall

```
#include "fvCFD.H"
#include "wallFvPatch.H"
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
int main(int argc, char *argv[])
{
    timeSelector::addOptions();
    #include "setRootCase.H"
    #include "createTime.H"
    instantList timeDirs = timeSelector::select0(runTime, args);
    #include "createMesh.H"

    forAll(timeDirs, timeI)
    {
        runTime.setTime(timeDirs[timeI], timeI);
        Info<< "Time = " << runTime.timeName() << endl;

        IOobject Theader
```

```
(
    "T",
    runTime.timeName(),
    mesh,
    IOobject::MUST_READ
);

// Check Texists
if (Theader.headerOk())
{
    mesh.readUpdate();

    Info<< "    Reading T" << endl;
    volScalarField T(Theader, mesh);

    Info<< "    Calculating wallGradT" << endl;

    volScalarField wallGradT
    (
        IOobject
        (
            "wallGradT",
            runTime.timeName(),
            mesh,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        mesh,
        dimensionedScalar
        (
            "wallGradT",
            T.dimensions()/dimLength,
            0
        )
    );

    const fvPatchList& patches = mesh.boundary();

    forAll(wallGradT.boundaryField(), patchi)
    {
        const fvPatch& currPatch = patches[patchi];
```

```
                if (isA<wallFvPatch>(currPatch))
                {
                    wallGradT.boundaryField()[patchi] =
                        -T.boundaryField()[patchi].snGrad();
                }
            }

            wallGradT.write();
        }
        else
        {
            Info<< "    No T" << endl;
        }
    }

    Info<< "End" << endl;

    return 0;
}
// ************************************************************************* //
```

B.4 Code for Obtaining Points and Normals on any Surface

This is specifically for C3X vane and can be modified for other surfaces after obtaining the surface points where normal points at a certain distance has to be located.

```
#include "fvMesh.H"
#include "volFields.H"
#include "Time.H"
#include "argList.H"
#include "surfaceFields.H"

using namespace Foam;

int main(int argc, char *argv[])
{
    # include "addTimeOptions.H"
    argList::validArgs.append("CS");
    # include "setRootCase.H"
    # include "createTime.H"
    # include "createMesh.H"
```

```
    Info<< "Dump face centres and normals of given patch\n" << endl;

    //ps1
    const word patchName_ps1 = "PS1";//args[1];
    label patchI = mesh.boundaryMesh().findPatchID(patchName_ps1);
    const polyPatch& ps1 = mesh.boundaryMesh()[patchI];
    //const surfaceVectorField& wallSf_ps1 = mesh.Sf();
    //const surfaceScalarField& wallmagSf_ps1 = mesh.magSf();
    Info<<ps1.faceCentres().size()<<endl;
    Info<<"("<<endl;
    forAll(ps1.faceCentres(), faceI)
    {
        scalar x = ps1.faceCentres()[faceI].x();
        scalar y = ps1.faceCentres()[faceI].y();
Info<<x<<" "<<y<<" "<<" "<<(mesh.Sf().boundaryField()[patchI][faceI])/(mesh.magSf().boundaryFie
    }
    Info<<")"<<endl;

    //Similarly for other surfaces by replacing ps1 by 'surfaceName'

   return 0;
}
```

B.5 OpenFOAM Case Files

0/U

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.2                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volVectorField;
    object      U;
}
```

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -1 0 0 0 0];

internalField   uniform (6 0 0);

boundaryField
{
    inlet
    {
        type            fixedValue;
        value uniform (6 0 0);
    }

    outlet
    {
        type            zeroGradient;

    }

    flatPlate
    {
        type            fixedValue;
        value           uniform (0 0 0);
    }

    topPlane
    {
type  symmetry;
    }
    bottomPlane
    {
type slip;
   }

    frontAndBack
    {
        type            empty;
    }
}
```

```
// ************************************************************************* //
```

0/epslion

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.2                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      epsilon;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -3 0 0 0 0];

internalField   uniform 75;

boundaryField
{
    inlet
    {
        type            fixedValue;
        value uniform 75;
    }

    outlet
    {
        type            zeroGradient;

    }

    flatPlate
    {
        type           fixedValue;
        value           uniform 1e-12;
    }
```

```
    topPlane
    {
type  symmetry;

    }
    bottomPlane
    {
type slip;
    }
    frontAndBack
    {
        type            empty;
    }
}

// ************************************************************************* //
```

0/k

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.2                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      k;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 3.56;

boundaryField
{
    inlet
```

```
    {
        type            fixedValue;
        value uniform 3.56;
    }

    outlet
    {
        type            zeroGradient;

    }

    flatPlate
    {
        type            fixedValue;
        value           uniform 1e-14;
    }
    bottomPlane
    {
type slip;
    }
    topPlane
    {
type  symmetry;
    }

    frontAndBack
    {
        type            empty;
    }
}

// ************************************************************************* //
```

0/nut

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.2                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
```

```
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      nut;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -1 0 0 0 0];

internalField   uniform 0.007;

boundaryField
{
    inlet
    {
        type            calculated;
        value uniform 0.007;
    }

    outlet
    {
        type            calculated;
        value uniform 0;
    }

    flatPlate
    {

      type            nutLowReWallFunction;
      value           uniform 0;
    }
    bottomPlane
    {
type  slip;
value uniform 0;
    }
    topPlane
    {
type symmetry;
value uniform 0;
```

```
    }

    frontAndBack
    {
        type            empty;
    }
}

// ************************************************************************* //
```

0/p_rgh

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.x                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    location    "6853";
    object      p_rgh;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 2 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
    inlet
    {
        type            zeroGradient;
    }
    outlet
    {
        type            fixedValue;//totalPressure;
value           uniform 0;
```

```
    }
    topPlane
    {
        type            symmetry;//zeroGradient;
    }
    bottomPlane
    {
        type            slip;
    }
    flatPlate
    {
        type            zeroGradient;
    }
    frontAndBack
    {
        type            empty;
    }
}


// ************************************************************************* //
```

0/T

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.2                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      T;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 0 0 1 0 0 0];

internalField   uniform 295;
```

```
boundaryField
{
    inlet
    {
        type            fixedValue;
value  uniform 295;

    }

    outlet
    {
        type            zeroGradient;
// value uniform 0;
    }

    flatPlate
    {
        type            fixedValue;
value uniform 310;
    }
    bottomPlane
    {
type slip;//symmetry;//zeroGradient;
    }
    topPlane
    {
type  symmetry;//zeroGradient;
    }

    frontAndBack
    {
        type            empty;
    }
}

// ************************************************************************* //
```

constant/g

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
```

```
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       uniformDimensionedVectorField;
    location    "constant";
    object      g;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions      [0 1 -2 0 0 0 0];
value           ( 0 0 0 );


// ************************************************************************* //
```

constant/RASProperties

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      RASProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

RASModel        LSfull;//LaunderSharmaKE;

turbulence      on;

printCoeffs     on;
```

constant

constant/transportProperties

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      transportProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

transportModel  Newtonian;

rho             rho [ 1 -3 0 0 0 0 0 ] 1.225;
nu              nu [ 0 2 -1 0 0 0 0 ] 1.4607e-05;
Pr Pr [ 0 0 0 0 0 0 0] 0.705;
Prt Prt [0 0 0 0 0 0 0] 0.85;
TRef TRef [0 0 0 1 0 0 0] 300;
beta beta [0 0 0 -1 0 0 0] 0.0;
```

constant/turbulenceProperties

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "constant";
    object      turbulenceProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

simulationType RASModel;
```

system/controlDict

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.2                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
```

```
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

//application     simpleFoam;
application buoyantBoussinesqSimpleFoam;
startFrom       latestTime;
startTime       0;
stopAt          endTime;
endTime         150000;
deltaT          1;
writeControl    timeStep;
writeInterval   1000;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable true;
libs
(
"libmyFullCurved3.so"
);
functions
{
#include "forceCoeffs"
}
// ************************************************************************* //
```

system/fvSchemes

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.2                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
```

```
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSchemes;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

ddtSchemes
{
    default         steadyState;
}

gradSchemes
{
    default         cellMDLimited Gauss linear 1.0;
}

divSchemes
{
    default           none;
    div(phi,U) bounded Gauss linearUpwindV cellMDLimited Gauss linear 1.0;
    div(phi,T) bounded Gauss linearUpwind cellMDLimited Gauss linear 1;
    div(phi,k)      bounded Gauss upwind;
    div(phi,epsilon) bounded Gauss upwind;
    div(phi,omega) bounded Gauss upwind;
    div((nuEff*dev(T(grad(U))))) Gauss linear;
    div((0.05*(R&U))) Gauss linear;
}

laplacianSchemes
{
   default         Gauss linear limited corrected 0.33;
}

interpolationSchemes
{
    default         linear;
```

```
}

snGradSchemes
{
    default        limited corrected 0.33;
}

fluxRequired
{
    default         no;
    p               ;
    p_rgh;
    pCorr;
    Phi;
    //T ;
}
```

system/fvSolution

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.2.2                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      fvSolution;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

solvers
{
    "(p|pCorr|p_rgh)"
    {

solver PCG;
preconditioner DIC;
```

```
tolerance 1e-6;
relTol 0.001;
//maxIter 100;
    }

    "(pCorr|p|p_rgh)Final"
    {
$p;
tolerance 1e-6;
relTol 0;
    }
    "(U|T|k|epsilon|omega)"
    {
        solver          PBiCG;
        preconditioner  DILU;
        tolerance       1e-6;
        relTol          0.0001;
     // minIter 1;
    }

    "(U|T|k|epsilon|omega)Final"
    {
        $U;
        tolerance       1e-6;
        relTol          0.0;
     // minIter 1;
    }
    Phi
    {
solver GAMG;
smoother DIC;
cacheAgglomeration on;
nCellsInCoarsestLevel 40;
mergeLevels 1;
tolerance 1e-8;
relTol 0.001;
    }



}
```

```
SIMPLE
{
    nNonOrthogonalCorrectors 1;
    pRefCell 0;
    pRefValue 0;

residualControl
{
p_rgh 1e-8;
p 1e-8;
U 1e-8;
k 1e-8;
epsilon 1e-8;
                T 1e-8;
omega 1e-8;
}

}

relaxationFactors
{
    fields
    {
p               0.2;
    }
    equations
    {
      default  0.5;
U 0.4;
k 0.5;
epsilon 0.5;
omega 0.5;
T 0.4;
    }
}

potentialFlow
{
nNonOrthogonalCorrectors 10;
}
```

```
// ********************************************************************** //
```

system/sample

```
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  2.3.0                                 |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      sampleDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

setFormat raw;

surfaceFormat raw;

interpolationScheme cellPoint;

fields
(
    U
    T
    k
//    omega
    epsilon
    wallShearStress
    yPlus
    wallHeatFlux
    wallGradU
    p
    wallGradT
);
```

```
sets
(
    probe_P1
    {
type uniform;//midPointAndFace;
        axis         xyz;
        start        (0.066883 0.35862 0);
        end          (0.06023 0.35115 0);
nPoints 350;
    }

    probe_P2
    {
type uniform;//midPointAndFace;
        axis         xyz;
        start        (0.13503 0.28587 0);
        end          (0.12719 0.27966 0);
nPoints 350;
    }

    probe_P3
    {
type uniform;//midPointAndFace;
        axis         xyz;
        start        (0.19274 0.20162 0);
        end          (0.18416 0.19648 0);
     nPoints 350;
    }

    probe_S1
    {
type uniform;//midPointAndFace;
        axis         xyz;
        start        (0.05829 0.50825 0);
        end          (0.055973 0.51798 0);
nPoints 350;
    }

    probe_S2
    {
type uniform;//midPointAndFace;
```

```
        axis         xyz;
        start        (0.1409 0.46008 0);
        end          (0.14937 0.46539 0);
nPoints 350;
    }

    probe_S3
    {
type uniform;//midPointAndFace;
        axis         xyz;
        start        (0.18352 0.36946 0);
        end          (0.19281 0.37315 0);
nPoints 350;
    }

    probe_S4
    {
type uniform;//midPointAndFace;
        axis         xyz;
        start        (0.21759 0.27759 0);
        end          (0.22702 0.28092 0);
nPoints 350;
    }

);

surfaces
(
    ps1
    {
        type            patch;
        patches         ("PS1");
    }

    ps2
    {
type patch;
patches ("PS2");
    }

    ps3
```

```
    {
type patch;
patches ("PS3");
    }

    ps4
    {
type patch;
patches ("PS4");
    }

    ps5
    {
type patch;
patches ("PS5");
    }

    ss1
    {
type patch;
patches ("SS1");
    }

    ss2
    {
type patch;
patches ("SS2");
    }

    ss3
    {
type patch;
patches ("SS3");
    }

    ss4
    {
type patch;
patches ("SS4");
    }
```

```
    ss5
    {
type patch;
patches ("SS5");
    }
);
// ************************************************************************* //
```

Appendix C
MATLAB codes

C.1 Post-processing for C3X turbine vane

Properties of fluid and geometry

```
rho = 1.225;
mu = 1.7894e-5;
Cp_air = 1004.4;
Pr = 0.705;
Uin = 5.48;
C = 0.531;%0.562;
Twall = 330;
Tinf = 305;
```

Read the experimental data from the pre-generated .csv or .txt files

```
/%% Read Dees data
DeesData = "Dees/";
U_SS1_Tu05 = dlmread(strcat(DeesData,'U_SS1_Tu05.csv'));
U_SS2_Tu05 = dlmread(strcat(DeesData,'U_SS2_Tu05.csv'));
U_SS3_Tu05 = dlmread(strcat(DeesData,'U_SS3_Tu05.csv'));
U_PS1_Tu05 = dlmread(strcat(DeesData,'U_PS1_Tu05.csv'));
U_PS2_Tu05 = dlmread(strcat(DeesData,'U_PS2_Tu05.csv'));
U_PS3_Tu05 = dlmread(strcat(DeesData,'U_PS3_Tu05.csv'));
U_SS4_Tu05 = dlmread(strcat(DeesData,'U_SS4_Tu05.csv'));
k_SS1_Tu05 = dlmread(strcat(DeesData,'k_SS1_Tu05.csv'));
k_SS2_Tu05 = dlmread(strcat(DeesData,'k_SS2_Tu05.csv'));
k_SS3_Tu05 = dlmread(strcat(DeesData,'k_SS3_Tu05.csv'));
k_PS1_Tu05 = dlmread(strcat(DeesData,'k_PS1_Tu05.csv'));
k_PS2_Tu05 = dlmread(strcat(DeesData,'k_PS2_Tu05.csv'));
k_PS3_Tu05 = dlmread(strcat(DeesData,'k_PS3_Tu05.csv'));
```

```
k_SS4_Tu05 = dlmread(strcat(DeesData,'k_SS4_Tu05.csv'));
T_SS1_Tu05 = dlmread(strcat(DeesData,'T_SS1_Tu05.csv'));
T_SS2_Tu05 = dlmread(strcat(DeesData,'T_SS2_Tu05.csv'));
T_SS3_Tu05 = dlmread(strcat(DeesData,'T_SS3_Tu05.csv'));
T_PS1_Tu05 = dlmread(strcat(DeesData,'T_PS1_Tu05.csv'));
T_PS2_Tu05 = dlmread(strcat(DeesData,'T_PS2_Tu05.csv'));
T_PS3_Tu05 = dlmread(strcat(DeesData,'T_PS3_Tu05.csv'));
T_SS4_Tu05 = dlmread(strcat(DeesData,'T_SS4_Tu05.csv'));

U_SS1_Tu20 = dlmread(strcat(DeesData,'U_SS1_Tu20.csv'));
U_SS2_Tu20 = dlmread(strcat(DeesData,'U_SS2_Tu20.csv'));
U_SS3_Tu20 = dlmread(strcat(DeesData,'U_SS3_Tu20.csv'));
U_PS1_Tu20 = dlmread(strcat(DeesData,'U_PS1_Tu20.csv'));
U_PS2_Tu20 = dlmread(strcat(DeesData,'U_PS2_Tu20.csv'));
U_PS3_Tu20 = dlmread(strcat(DeesData,'U_PS3_Tu20.csv'));
U_SS4_Tu20 = dlmread(strcat(DeesData,'U_SS4_Tu20.csv'));
k_SS1_Tu20 = dlmread(strcat(DeesData,'k_SS1_Tu20.csv'));
k_SS2_Tu20 = dlmread(strcat(DeesData,'k_SS2_Tu20.csv'));
k_SS3_Tu20 = dlmread(strcat(DeesData,'k_SS3_Tu20.csv'));
k_PS1_Tu20 = dlmread(strcat(DeesData,'k_PS1_Tu20.csv'));
k_PS2_Tu20 = dlmread(strcat(DeesData,'k_PS2_Tu20.csv'));
k_PS3_Tu20 = dlmread(strcat(DeesData,'k_PS3_Tu20.csv'));
k_SS4_Tu20 = dlmread(strcat(DeesData,'k_SS4_Tu20.csv'));
T_SS1_Tu20 = dlmread(strcat(DeesData,'T_SS1_Tu20.csv'));
T_SS2_Tu20 = dlmread(strcat(DeesData,'T_SS2_Tu20.csv'));
T_SS3_Tu20 = dlmread(strcat(DeesData,'T_SS3_Tu20.csv'));
T_PS1_Tu20 = dlmread(strcat(DeesData,'T_PS1_Tu20.csv'));
T_PS2_Tu20 = dlmread(strcat(DeesData,'T_PS2_Tu20.csv'));
T_PS3_Tu20 = dlmread(strcat(DeesData,'T_PS3_Tu20.csv'));
T_SS4_Tu20 = dlmread(strcat(DeesData,'T_SS4_Tu20.csv'));

Cp_Dees = dlmread('Dees/Cp_Dees.csv');
Nu_Dees_Tu05 = dlmread('Dees/Nu_Tu05_Dees.csv');
Nu_Dees_Tu20 = dlmread('Dees/Nu_Tu20_Dees.csv');

DysonData = "Dyson/";
Cp_Dyson = dlmread('Dees/Cp_Dyson.csv');
Nu_Dyson_Tu05_kwSST = dlmread('Dees/Nu_Tu05_Dyson_kwSST.csv');
Nu_Dyson_Tu20_kwSST = dlmread('Dees/Nu_Tu20_Dyson_kwSST.csv');
Nu_Dyson_Tu05_RKE = dlmread('Dees/Nu_Tu05_Dyson_RKE.csv');
Nu_Dyson_Tu20_RKE = dlmread('Dees/Nu_Tu20_Dyson_RKE.csv');
```

Reading the data of the simulation from the files generated using sample utility. Changes in the folder name should be made based on the name of the case and the location of the post-processing folder

```
/%% read CFD data
folderName = "Tu20_old/surfaces/111000/";
wallName = ["ps1","ps2","ps3","ps4","ps5","ss1","ss2","ss3","ss4","ss5"];
nWalls = length(wallName);
nWallPoint = zeros(nWalls,1);

i=1;
/% ps1
shear_ps1 = dlmread(strcat(folderName,'wallShearStress','_',wallName(i),'.raw'),'',2,0);
heatData_ps1 =  dlmread(strcat(folderName,'wallHeatFlux','_',wallName(i),'.raw'),'',2,3);
wallGradT_ps1 = dlmread(strcat(folderName,'wallGradT','_',wallName(i),'.raw'),'',2,3);
wallGradU_ps1 = dlmread(strcat(folderName,'wallGradU','_',wallName(i),'.raw'),'',2,3);
yPlusData_ps1 = dlmread(strcat(folderName,'yPlus','_',wallName(i),'.raw'),'',2,3);
p_ps1 = dlmread(strcat(folderName,'p','_',wallName(i),'.raw'),'',2,3);
i=i+1;

/% ps2
shear_ps2 = dlmread(strcat(folderName,'wallShearStress','_',wallName(i),'.raw'),'',2,0);
heatData_ps2 =  dlmread(strcat(folderName,'wallHeatFlux','_',wallName(i),'.raw'),'',2,3);
wallGradT_ps2 = dlmread(strcat(folderName,'wallGradT','_',wallName(i),'.raw'),'',2,3);
wallGradU_ps2 = dlmread(strcat(folderName,'wallGradU','_',wallName(i),'.raw'),'',2,3);
yPlusData_ps2 = dlmread(strcat(folderName,'yPlus','_',wallName(i),'.raw'),'',2,3);
p_ps2 = dlmread(strcat(folderName,'p','_',wallName(i),'.raw'),'',2,3);
i=i+1;

/% ps3
shear_ps3 = dlmread(strcat(folderName,'wallShearStress','_',wallName(i),'.raw'),'',2,0);
heatData_ps3 =  dlmread(strcat(folderName,'wallHeatFlux','_',wallName(i),'.raw'),'',2,3);
wallGradT_ps3 = dlmread(strcat(folderName,'wallGradT','_',wallName(i),'.raw'),'',2,3);
wallGradU_ps3 = dlmread(strcat(folderName,'wallGradU','_',wallName(i),'.raw'),'',2,3);
yPlusData_ps3 = dlmread(strcat(folderName,'yPlus','_',wallName(i),'.raw'),'',2,3);
p_ps3 = dlmread(strcat(folderName,'p','_',wallName(i),'.raw'),'',2,3);
i=i+1;

/% ps4
shear_ps4 = dlmread(strcat(folderName,'wallShearStress','_',wallName(i),'.raw'),'',2,0);
heatData_ps4 =  dlmread(strcat(folderName,'wallHeatFlux','_',wallName(i),'.raw'),'',2,3);
wallGradT_ps4 = dlmread(strcat(folderName,'wallGradT','_',wallName(i),'.raw'),'',2,3);
```

```
wallGradU_ps4 = dlmread(strcat(folderName,'wallGradU','_',wallName(i),'.raw'),'',2,3);
yPlusData_ps4 = dlmread(strcat(folderName,'yPlus','_',wallName(i),'.raw'),'',2,3);
p_ps4 = dlmread(strcat(folderName,'p','_',wallName(i),'.raw'),'',2,3);
i=i+1;

/% ps5
shear_ps5 = dlmread(strcat(folderName,'wallShearStress','_',wallName(i),'.raw'),'',2,0);
heatData_ps5 =  dlmread(strcat(folderName,'wallHeatFlux','_',wallName(i),'.raw'),'',2,3);
wallGradT_ps5 = dlmread(strcat(folderName,'wallGradT','_',wallName(i),'.raw'),'',2,3);
wallGradU_ps5 = dlmread(strcat(folderName,'wallGradU','_',wallName(i),'.raw'),'',2,3);
yPlusData_ps5 = dlmread(strcat(folderName,'yPlus','_',wallName(i),'.raw'),'',2,3);
p_ps5 = dlmread(strcat(folderName,'p','_',wallName(i),'.raw'),'',2,3);
i=i+1;

/% ss1
shear_ss1 = dlmread(strcat(folderName,'wallShearStress','_',wallName(i),'.raw'),'',2,0);
heatData_ss1 =  dlmread(strcat(folderName,'wallHeatFlux','_',wallName(i),'.raw'),'',2,3);
wallGradT_ss1 = dlmread(strcat(folderName,'wallGradT','_',wallName(i),'.raw'),'',2,3);
wallGradU_ss1 = dlmread(strcat(folderName,'wallGradU','_',wallName(i),'.raw'),'',2,3);
yPlusData_ss1 = dlmread(strcat(folderName,'yPlus','_',wallName(i),'.raw'),'',2,3);
p_ss1 = dlmread(strcat(folderName,'p','_',wallName(i),'.raw'),'',2,3);
i=i+1;

/% ss2
shear_ss2 = dlmread(strcat(folderName,'wallShearStress','_',wallName(i),'.raw'),'',2,0);
heatData_ss2 =  dlmread(strcat(folderName,'wallHeatFlux','_',wallName(i),'.raw'),'',2,3);
wallGradT_ss2 = dlmread(strcat(folderName,'wallGradT','_',wallName(i),'.raw'),'',2,3);
wallGradU_ss2 = dlmread(strcat(folderName,'wallGradU','_',wallName(i),'.raw'),'',2,3);
yPlusData_ss2 = dlmread(strcat(folderName,'yPlus','_',wallName(i),'.raw'),'',2,3);
p_ss2 = dlmread(strcat(folderName,'p','_',wallName(i),'.raw'),'',2,3);
i=i+1;

/% ss3
shear_ss3 = dlmread(strcat(folderName,'wallShearStress','_',wallName(i),'.raw'),'',2,0);
heatData_ss3 =  dlmread(strcat(folderName,'wallHeatFlux','_',wallName(i),'.raw'),'',2,3);
wallGradT_ss3 = dlmread(strcat(folderName,'wallGradT','_',wallName(i),'.raw'),'',2,3);
wallGradU_ss3 = dlmread(strcat(folderName,'wallGradU','_',wallName(i),'.raw'),'',2,3);
yPlusData_ss3 = dlmread(strcat(folderName,'yPlus','_',wallName(i),'.raw'),'',2,3);
p_ss3 = dlmread(strcat(folderName,'p','_',wallName(i),'.raw'),'',2,3);
i=i+1;
```

```
/% ss4
shear_ss4 = dlmread(strcat(folderName,'wallShearStress','_',wallName(i),'.raw'),'',2,0);
heatData_ss4 =  dlmread(strcat(folderName,'wallHeatFlux','_',wallName(i),'.raw'),'',2,3);
wallGradT_ss4 = dlmread(strcat(folderName,'wallGradT','_',wallName(i),'.raw'),'',2,3);
wallGradU_ss4 = dlmread(strcat(folderName,'wallGradU','_',wallName(i),'.raw'),'',2,3);
yPlusData_ss4 = dlmread(strcat(folderName,'yPlus','_',wallName(i),'.raw'),'',2,3);
p_ss4 = dlmread(strcat(folderName,'p','_',wallName(i),'.raw'),'',2,3);
i = i+1;

/% ss5
shear_ss5 = dlmread(strcat(folderName,'wallShearStress','_',wallName(i),'.raw'),'',2,0);
heatData_ss5 =  dlmread(strcat(folderName,'wallHeatFlux','_',wallName(i),'.raw'),'',2,3);
wallGradT_ss5 = dlmread(strcat(folderName,'wallGradT','_',wallName(i),'.raw'),'',2,3);
wallGradU_ss5 = dlmread(strcat(folderName,'wallGradU','_',wallName(i),'.raw'),'',2,3);
yPlusData_ss5 = dlmread(strcat(folderName,'yPlus','_',wallName(i),'.raw'),'',2,3);
p_ss5 = dlmread(strcat(folderName,'p','_',wallName(i),'.raw'),'',2,3);
```

Conversion of experimental data into vane coordinate s for both pressure side and suction side

```
/%% Convert into experimental format
/% Center line: y + 1.4676176846x - 0.431634866 = 0
originValue = -0.431634866;
n_SS1 = length(shear_ss1(:,1));
n1 = 1;
n2 = 1;
for i = 1:n_SS1
    wrtOrigin = shear_ss1(i,2) + 1.4676176846*shear_ss1(i,1) - 0.431634866;
    if (wrtOrigin<0)
        xPS(n1) = shear_ss1(i,1);
        yPS(n1) = shear_ss1(i,2);
        shearxPS(n1) = shear_ss1(i,4);
        shearyPS(n1) = shear_ss1(i,5);
        heatPS(n1) = heatData_ss1(i);
        yPlusPS(n1) = yPlusData_ss1(i);
        pPS(n1) = p_ss1(i);
        wallTPS(n1) = wallGradT_ss1(i);
        wallUxPS(n1) = wallGradU_ss1(i,1);
        wallUyPS(n1) = wallGradU_ss1(i,2);
        nxPS(n1) = SS1(i,3);
        nyPS(n1) = SS1(i,4);
        n1 = n1+1;
    else
```

```
        xSS(n2) = shear_ss1(i,1);
        ySS(n2) = shear_ss1(i,2);
        shearxSS(n2) = shear_ss1(i,4);
        shearySS(n2) = shear_ss1(i,5);
        heatSS(n2) = heatData_ss1(i);
        yPlusSS(n2) = yPlusData_ss1(i);
        pSS(n2) = p_ss1(i);
        wallTSS(n2) = wallGradT_ss1(i);
        wallUxSS(n2) = wallGradU_ss1(i,1);
        wallUySS(n2) = wallGradU_ss1(i,2);
        nxSS(n2) = SS1(i,3);
        nySS(n2) = SS1(i,4);
        n2 = n2+1;
    end
end
if (n1>1)
    A(:,1) = flip(xPS');
    A(:,2) = flip(yPS');
    X_PS =    [A(:,1);flip(PS1(:,1));flip(PS2(:,1));flip(PS3(:,1));flip(PS4(:,1));flip(PS5(:,1)
    Y_PS =    [A(:,2);flip(PS1(:,2));flip(PS2(:,2));flip(PS3(:,2));flip(PS4(:,2));flip(PS5(:,2)
    nx_PS = [flip(nxPS');flip(PS1(:,3));flip(PS2(:,3));flip(PS3(:,3));flip(PS4(:,3));
    flip(PS5(:,3))];
    ny_PS = [flip(nyPS');flip(PS1(:,4));flip(PS2(:,4));flip(PS3(:,4));flip(PS4(:,4));
    flip(PS5(:,4))];
    shearStressx_PS = [flip(shearxPS');flip(shear_ps1(:,4));flip(shear_ps2(:,4));flip(shear_ps3
    flip(shear_ps4(:,4));flip(shear_ps5(:,4))];
    shearStressy_PS = [flip(shearyPS');flip(shear_ps1(:,5));flip(shear_ps2(:,5));flip(shear_ps3
    flip(shear_ps4(:,5));flip(shear_ps5(:,5))];
    heatFlux_PS = [flip(heatPS');flip(heatData_ps1);flip(heatData_ps2);flip(heatData_ps3);
    flip(heatData_ps4);flip(heatData_ps5)];
    wallGradUxPS = [flip(wallUxPS');flip(wallGradU_ps1(:,1));flip(wallGradU_ps2(:,1));
    flip(wallGradU_ps3(:,1));flip(wallGradU_ps4(:,1));flip(wallGradU_ps5(:,1))];
    wallGradUyPS = [flip(wallUyPS');flip(wallGradU_ps1(:,2));flip(wallGradU_ps2(:,2));
    flip(wallGradU_ps3(:,2));flip(wallGradU_ps4(:,2));flip(wallGradU_ps5(:,2))];
    wallGradTPS = [flip(wallTPS');flip(wallGradT_ps1(:,1));flip(wallGradT_ps2(:,1));
    flip(wallGradT_ps3(:,1));flip(wallGradT_ps4(:,1));flip(wallGradT_ps5(:,1))];
    yPlus_PS = [flip(yPlusPS');flip(yPlusData_ps1);flip(yPlusData_ps2);flip(yPlusData_ps3);
    flip(yPlusData_ps4);flip(yPlusData_ps5)];
    pressure_PS = [flip(pPS');flip(p_ps1);flip(p_ps2);flip(p_ps3);flip(p_ps4);flip(p_ps5)];

else
```

```
    X_PS = [flip(PS1(:,1));flip(PS2(:,1));flip(PS3(:,1));flip(PS4(:,1));flip(PS5(:,1))];
    Y_PS = [flip(PS1(:,2));flip(PS2(:,2));flip(PS3(:,2));flip(PS4(:,2));flip(PS5(:,2))];
    shearStressx_PS = [flip(shear_ps1(:,4));flip(shear_ps2(:,4));flip(shear_ps3(:,4));
    flip(shear_ps4(:,4));flip(shear_ps5(:,4))];
    shearStressy_PS = [flip(shear_ps1(:,5));flip(shear_ps2(:,5));flip(shear_ps3(:,5));
    flip(shear_ps4(:,5));flip(shear_ps5(:,5))];
    heatFlux_PS = [flip(heatData_ps1);flip(heatData_ps2);flip(heatData_ps3);
    flip(heatData_ps4);flip(heatData_ps5)];
    wallGradUxPS = [flip(wallGradU_ps1(:,1));flip(wallGradU_ps2(:,1));flip(wallGradU_ps3(:,1));
    flip(wallGradU_ps4(:,1));flip(wallGradU_ps5(:,1))];
    wallGradUyPS = [flip(wallGradU_ps1(:,2));flip(wallGradU_ps2(:,2));flip(wallGradU_ps3(:,2));
    flip(wallGradU_ps4(:,2));flip(wallGradU_ps5(:,2))];
    wallGradTPS = [flip(wallGradT_ps1(:,1));flip(wallGradT_ps2(:,1));flip(wallGradT_ps3(:,1));
    flip(wallGradT_ps4(:,1));flip(wallGradT_ps5(:,1))];
    yPlus_PS = [flip(yPlusData_ps1);flip(yPlusData_ps2);flip(yPlusData_ps3);
    flip(yPlusData_ps4);flip(yPlusData_ps5)];
    pressure_PS = [flip(p_ps1);flip(p_ps2);flip(p_ps3);flip(p_ps4);flip(p_ps5)];
    nx_PS = [flip(PS1(:,3));flip(PS2(:,3));flip(PS3(:,3));
    flip(PS4(:,3));flip(PS5(:,3))];
    ny_PS = [flip(PS1(:,4));flip(PS2(:,4));flip(PS3(:,4));
    flip(PS4(:,4));flip(PS5(:,4))];

end
B(:,1) = xSS';
B(:,2) = ySS';

X_SS = cat(1,B(:,1),SS2(:,1),SS3(:,1),SS4(:,1),SS5(:,1));
Y_SS = [B(:,2);SS2(:,2);SS3(:,2);SS4(:,2);SS5(:,2)];
nx_SS = cat(1,nxSS',SS2(:,3),SS3(:,3),SS4(:,3),SS5(:,3));
ny_SS = [nySS';SS2(:,4);SS3(:,4);SS4(:,4);SS5(:,4)];
shearStressx_SS = [shearxSS';shear_ss2(:,4);shear_ss3(:,4);
shear_ss4(:,4);shear_ss5(:,4)];
shearStressy_SS = [shearySS';shear_ss2(:,5);shear_ss3(:,5);
shear_ss4(:,5);shear_ss5(:,5)];
heatFlux_SS = [heatSS';heatData_ss2;heatData_ss3;heatData_ss4;heatData_ss5];
wallGradUxSS = [wallUxSS';wallGradU_ss2(:,1);wallGradU_ss3(:,1);wallGradU_ss4(:,1);
wallGradU_ss5(:,1)];
wallGradUySS = [wallUySS';wallGradU_ss2(:,2);wallGradU_ss3(:,2);wallGradU_ss4(:,2);
wallGradU_ss5(:,2)];
wallGradTSS = [wallTSS';wallGradT_ss2;wallGradT_ss3;wallGradT_ss4;
wallGradT_ss5];
```

```
yPlus_SS = [yPlusSS';yPlusData_ss2;yPlusData_ss3;yPlusData_ss4;
yPlusData_ss5];
pressure_SS = [pSS';p_ss2;p_ss3;p_ss4;p_ss5];

plot(X_PS,Y_PS,X_SS,Y_SS)
legend('PS','SS')
n_SS = length(X_SS);
n_PS = length(Y_PS);
ds_SS = zeros(n_SS,1);
ds_PS = zeros(n_PS,1);
s_SS = zeros(n_SS,1);
s_PS = zeros(n_PS,1);
for i=2:n_SS
    ds_SS(i) = sqrt((X_SS(i)-X_SS(i-1))^2 + (Y_SS(i)-Y_SS(i-1))^2);
    s_SS(i) = s_SS(i-1) + ds_SS(i);
end

for i=2:n_PS
    ds_PS(i) = sqrt((X_PS(i)-X_PS(i-1))^2 + (Y_PS(i)-Y_PS(i-1))^2);
    s_PS(i) = s_PS(i-1) + ds_PS(i);
end
sByC_SS = s_SS./C;
sByC_PS = -s_PS./C;

expLoc = [-0.19 -0.38 -0.57 0.19 0.38 0.57 0.75];
nLoc = length(expLoc);
xLoc = zeros(nLoc,1);
yLoc = xLoc;
for i = 1:nLoc
    if(expLoc(i)<0)
        index = find(sByC_PS<expLoc(i),1);
        xLoc(i) = X_PS(index);
        yLoc(i) = Y_PS(index);
    else
        index = find(sByC_SS>expLoc(i),1);
        xLoc(i) = X_SS(index);
        yLoc(i) = Y_SS(index);
    end
end
```

C.2 Generation of Initial Turbulent Profiles for Flat Plate

The profile is based on the method used by Iyer [48] to generate the initial profiles of k and ε. The generated profiles are exported as text files to import in ANSYS-Fluent or OpenFOAM code. Selection of velocity and temperature profile is based on the user discretion of generally accepted power law (one-seventh, one-fifth etc)

```
/% Initial profiles generation for fully developed turbulent flows.
/% Created by: Vedant Chittlangia (vc.vedant@gmail.com)
/% Date: 11 June 2017
N = 1000;
yPlus = linspace(0,1000,N);
dy = yPlus(2);
/%Uinf = 6;
Uinf = 30.48;
/%delta = 0.008;
delta = 0.00615;
mu = 1.7894e-5;
rho = 1.225;
nu = mu/rho;
/%xi = power((delta/0.37)*((Uinf/nu)^0.25),1/0.75);
xi = 0.318;
Rex = Uinf*xi/nu;
Cf2 = 0.185*((log10(Rex))^-2.584);
tau = Cf2*rho*Uinf*Uinf;
ustar = sqrt(tau/rho);
A = 25;  % Aplus
K = 0.41;   % kappa

Tinf = 295;
Twall = 310;
k = 0.0242;
Cp = 1006.43;
alp = k/(rho*Cp);
Pr = nu/alp;
Prt = 0.85;     %turbulent prandtl number
St = 0.03*(Rex^-0.2)*(Pr^-0.4);

du = zeros(1,N);
U = zeros(1,N);
UG = zeros(1,N);        % U- gradient
T = zeros(1,N);
```

```
y = yPlus.*nu./ustar;
nut = zeros(1,N);
alpt = zeros(1,N);
dt = zeros(1,N);

/%% TKE profile: Tu = 6 %
kinf = 5.94;
/% kinf = 3.56;
l = K.*y;
fmu = 1 - exp(-0.0115.*yPlus);
d = 2.45.*fmu.*l.*(1-exp(-yPlus./A));
nut_new = nu.*(((1-exp(-yPlus./A)).*K.*yPlus).^2).*UG;
kold = (nut_new./d).^2;
Rek = (kold.^0.5).*l./nu;
fmu2 = 1 - exp(-0.029.*Rek);
knew = (nut_new./(0.548.*fmu2.*l)).^2;
relEr  = 1;
iter = 0;
while (relEr >= 0.01)
    iter = iter + 1;
    tke = knew;
    kold = 0.5.*(knew+kold);
    Rek = (kold.^0.5).*l./nu;
    fmu2 = 1 - exp(-0.029.*Rek);
    knew = (nut_new./(0.548.*fmu2.*l)).^2;
    relEr = max(abs(knew-kold)./kold);
end

a1 = -39592630;
b1 = 568480;
c1 = -2491.63;
d1 = 8.97176;

/% a1 = -23209370;
/% b1 = 336323;
/% c1 = -902.6684;
/% d1 = 1.139865;


for i = 1:N
    if y(i)>=0.085*delta/K && y(i)<delta
```

```
        tke(i) = a1*(y(i)^3) + b1*(y(i)^2) + c1*y(i) + d1;
    elseif y(i)>=delta
        tke(i) = kinf;
    end
end

index = find(y>0.085*delta/K,1);
y(index-1);
tke(index-1) = (tke(index)+tke(index-2))*0.5;
figure()
plot(tke,y,'k-','LineWidth',1)
/%title('TKE for Tu_{\infty} = 25.7%')
title('TKE for Tu_{\infty} = 6.53%')
xlabel('TKE (m^2/s^2)')
ylabel('y (m)')
set(gca,'FontSize',12,'fontWeight','bold')
ylim([0 0.01])


/%% Epsilon profile: Tu = 25.7%
eps = zeros(N,1);
epsInf = 175.99;
a2 = -155056200000;
b2 = 2328681000;
c2 = -11021950;
d2 = 15951.78;

/% epsInf = 93.45;
/% a2 = -1447359000;
/% b2 = 25254710;
/% c2 = -125219.9;
/% d2 = 219.9551;

for i = 2:N
    if y(i)<=0.085*delta/K
        eps(i) = 0.164*(tke(i)^1.5)/l(i);
    elseif y(i)<=delta
        eps(i) = a2*(y(i)^3) + b2*(y(i)^2) + c2*y(i) + d2;
        if eps(i)<=0
            eps(i) = epsInf;
        end
```

```
    else
        eps(i) = epsInf;
    end
end
figure()
plot(eps,y,'k-','LineWidth',1)
/%title('Epsilon profile for Tu_{\infty} = 25.7%')
title('Epsilon profile for Tu_{\infty} = 6.53%')
xlabel('\epsilon (m^2/s^3)')
ylabel('y (m)')
set(gca,'FontSize',12,'fontWeight','bold')
ylim([0 0.01])
```