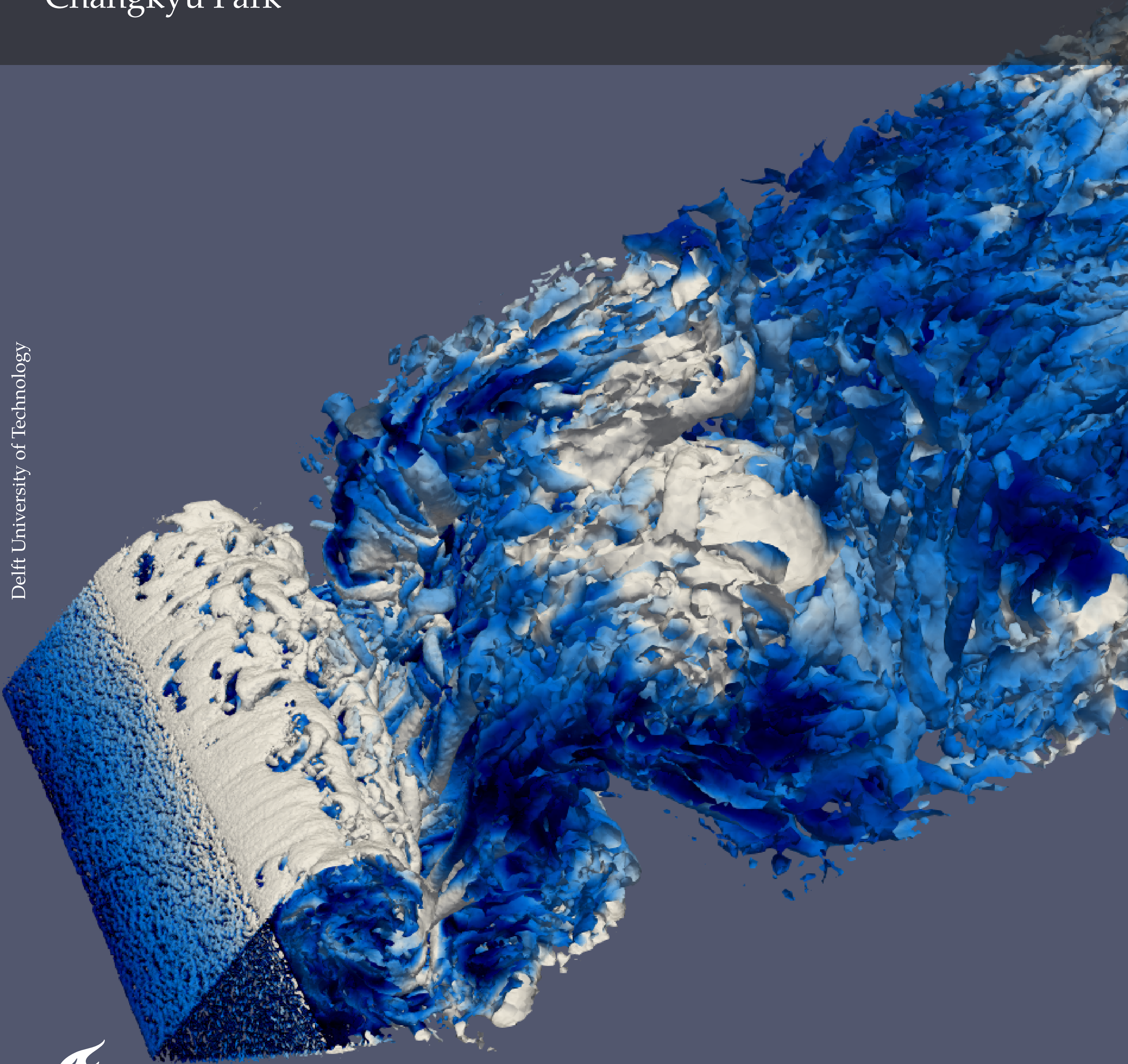
k-corrective frozen PANS: A data-driven stochastic turbulence closure model

Master of Science thesis

Changkyu Park



Delft University of Technology

TU Delft

k-corrective frozen PANS: A data-driven stochastic turbulence closure model

by

Changkyu Park

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday August 19, 2022 at 14:00 PM.

Student number:	4646061	
Project duration:	October 2021 – August 2022	
Thesis committee:	Dr. R. P. Dwight,	TU Delft, supervisor, chair
	Dr. D. Modesti ,	TU Delft, examiner
	Dr. ir. D. J. N. Allaerts ,	TU Delft, examiner

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Preface

This thesis marks the end of my five-year-long education at the Delft University of Technology through Bachelor's and Master's studies in the faculty of aerospace engineering. I appreciate the well-rounded education I received and the conducive environment I had during my stay here in Delft. Though challenging at times, it has been a remarkable learning experience.

I would like to thank Dr. Richard Dwight for the opportunity to work on this thesis topic under his guidance. While I received directions as to how I should approach various parts of the thesis, I was given sufficient freedom to challenge myself in developing various skills and knowledge.

I am extremely grateful for the support my family has given me throughout my entire studies, my friends in Delft who have made my stay here a memorable one and my old friends for always being a call away despite the time zone differences.

I look forward to the next checkpoint in my learning journey, new challenges and discoveries.

Changkyu Park
Delft, August 2022

Abstract

Studies revolving around data-driven methods have been on a rise in recent years to improve highly modelled methods such as the two-equation turbulence models of Reynolds-averaged Navier-Stokes (RANS). Similarly, such data-driven methods are implemented into partially-averaged Navier-Stokes (PANS). PANS is a young bridging method that fulfils the requirements of bridging methods set by Speziale [1]. It can be adjusted according to the fraction of a flow field that is desired by the user to be resolved and modelled by changing the value of f_k, a parameter that takes a value between 0 and 1. In this thesis, PANS is extended via a combination of two data-driven methods: $k-$corrective frozen RANS [2] and data-driven stochastic closure simulation (DSCS) [3]. The $k-$corrective frozen RANS method aims to correct for the model errors in the $k-$equation and the anisotropy of the Reynolds stress tensor derived from the Boussinesq approximation. While DSCS also aims to correct for the anisotropy of the Reynolds stress tensor, it considers that PANS solves for the unresolved part of the flow field and thus corrects for the unresolved anisotropy. While PANS and the two data-driven methods have independently been proved to work as they were theoretically desired, combining these ideas has not yet been attempted. As any turbulence kinetic energy solving RANS turbulence model can be developed into PANS form, the $k - \omega$ SST model was chosen for the best initial prediction. This SST model for PANS is extensively derived and then reformulated to produce the target correction terms. The correction terms are analysed at various values of f_k and they show good agreements with f_k.

Contents

Figures

Tables

Abbreviations

Abbreviation	Definition
2D	2-dimensional
3D	3-dimensional
AI	Artificial intelligence
BC	Boundary condition
BST	Baseline stress transport
CFD	Computational fluid dynamics
CFL	Courant–Friedrichs–Lewy
DES	Detached eddy simulation
DNS	Direct numerical simulation
DSCS	Data-driven Stochastic Closure Simulation
ET	Expression tree
FFT	Fast Fourier transform
GEP	Gene expression programming
HiFi	High-fidelity
IC	Initial condition
LES	Large eddy simulation
LHS	Left hand side
ML	Machine learning
N-S	Navier-Stokes
NN	Neural network
PANS	Partially-averaged Navier-Stokes
POD	Proper orthogonal decomposition
RANS	Reynolds-averaged Navier-Stokes
RHS	Right hand side
S-A	Spalart-Allmaras
SFS	Sub-filter scales
SST	Shear stress transport
TBNN	Tensor basis neural network
URANS	Unstead-RANS
VLES	Very large eddy simulation
WMLES	Wall-modelled large eddy simulation

Symbols

Symbol	Definition
b_{ij}	Anisotropy of Reynolds stress
b_{ij}^0	Baseline anisotropy of Reynolds stress
b_{ij}^Δ	Anisotropy stress model error
$CD_{k\omega}$	Cross diffusion between k and ω
D	Turbulence dissipation spectrum
d	Distance from a flow field point to nearest wall
d_h	Hydraulic diameter
dt	time step
E	Turbulence kinetic energy spectrum
f_k	Ratio of unresolved-to-total k
f_{vs}	Vortex shedding frequency
f_ε	Ratio of unresolved-to-total ε
f_ω	Ratio of unresolved-to-total ω
$\mathcal{I}$	Turbulence intensity
k	Turbulence kinetic energy
k_u	Unresolved k
L	Lift force
$\mathcal{L}$	Characteristic length
ℓ	Eddy size
ℓ_t	Turbulence length scale
P_k	Production term for k
p	Pressure
p'	Pressure fluctuation
$\overline{p}$	Mean pressure
R	$k-$equation model error term
Re	Reynolds number
RS_{ij}	Reynolds stress tensor
$\mathcal{R}_{ij}$	Time-scaled Ω_{ij}
r	Resolution
S_{ij}	Mean strain rate tensor
$\mathcal{S}_{ij}$	Time-scaled S_{ij}
T_k	Transport term for k
T_ω	Transport term for ω
t	time

Symbol	Definition
u	Velocity
u'	Velocity fluctuations
u''	Stochastic unsteadiness
$\overline{u}$	Mean velocity
$\widetilde{u}$	Periodic unsteadiness
u_x	Stream-wise velocity
δ_{ij}	Kronecker delta
ε	Dissipation rate
ε_u	Unresolved ε
η	Smallest turbulence length scale
θ	Phase coordinate
κ	Wave number
Λ	Largest turbulence length scale
μ	Dynamic viscosity
μ_t	Eddy viscosity
ν	Kinematic viscosity
ν_t	Kinematic eddy viscosity
ν_{tu}	Unresolved ν_t
ρ	Density
$\overline{\rho}$	Mean density
τ	Generalised central moment
ϕ	Arbitrary parameter
ϕ_0	Initial value of ϕ
Ω_{ij}	Rotation rate tensor
ω	Specific dissipation rate
ω_u	Unresolved ω

1

Introduction

1.1. Background

The Navier-Stokes equation, a partial differential equation that describes the motion of viscous fluid, has constantly been attempted to be numerically solved with and without turbulence models. Even with the rapid advancement of central and graphical processing unit (more commonly known as CPU and GPU) industries, implementation of direct numerical simulation (DNS) involving no turbulence models on a practical fluid domain at a high Reynolds number Re is still far from feasible. Therefore, methods with turbulence models have remained prevalent despite their shortcomings. To strike a balance between achieving a reasonably accurate solution and requiring moderate computational power, many bridging models have been developed. Such a motivation arose due to large eddy simulation (LES) costing too much computationally and Reynolds-averaged Navier-Stokes (RANS) equations having insufficient accuracy. Bridging methods bridge between highly modelled methods such as RANS and highly resolving methods such as DNS and LES.

A young bridging method called partially-averaged Navier-Stokes (PANS) is studied. PANS is a flexible bridging method which allows for any fraction of the flow to be resolved and modelled, provided a supporting mesh is present. The fraction is controlled via f_k, a variable that stays between 0 and 1. This flexibility is an advantageous characteristic as it is able to be adjusted to maximise the use of a given amount of computational power. The method is made theoretically possible by making use of the averaging invariance property of the Navier-Stokes equation wherein the form of the equation is not altered regardless of the extent of the averaging.

Addition of a data-driven approach that combines $k-$corrective frozen RANS method of [2] and data-driven stochastic closure simulation (DSCS) method of [3] to PANS is experimented in the project. The $k-$corrective method, as its name suggests, corrects for the model error in the $k-$equation while DSCS defines a closure for the purely stochastic part of the flow using triple decomposition. Data-driven studies for computational fluid dynamics have lately gained popularity due to several reasons. Firstly, it allows for making use of existing high-fidelity (HiFi) data that cost an immense amount to produce, reusing them for further studies. Using these data, the turbulence models are trained to give an improved prediction of a given fluid domain. However, it is impossible to have a HiFi dataset for every possible fluid domain in various flow conditions. Thus, data-driven methods, more importantly, aid in unearthing new relations to overcome the limitations that various involved assumptions possess, ultimately figuring out undiscovered physics.

The data-driven study conducted in this project involves correcting for the model errors in the turbulence kinetic energy equation of $k - \omega$ SST turbulence model that is adjusted for PANS and the anisotropy component of the Reynolds stress tensor. The goal is to determine if PANS, with various values of f_k, is capable of producing corrections that well-represents the flaws in the model.

1.2. Research objective and research questions

The main research objective for this thesis project is

> *"To combine two data-driven methods: $k-$corrective frozen RANS and DSCS to implement into the $k_u - \omega_u$ SST PANS turbulence model in improving its prediction of turbulent flows around triangular prism."*

Successfully carrying out the above objective would prove that the two data-driven methods can work together in augmenting the PANS turbulence model as this particular combination of methods has yet to be attempted. Thus, the research would provide a crucial platform for future studies to be built on regardless of the results. Among the broad spectrum of typical data-driven studies, the primary focus of this project is to extract the aforementioned model errors from the turbulence model. The follow-up to this step is the injection of the model errors back into the SST PANS model and the machine learning approach which will remain as recommendations for future work.

To approach the research objective, the main research question is first established:

> *"How can $k-$corrective frozen RANS and DSCS be combined to be implemented into the $k_u - \omega_u$ SST PANS turbulence model for improvement in the prediction of turbulent flows around triangular prism?"*

In answering the main question in a gradual and systematic manner sub-questions are established as well:

- Does PANS work as advertised, resolving a bigger portion of the flow with smaller f_k? How does it work when the mesh resolution is kept fixed? Does it perform better than its baseline model?
- How is $k - \omega$ SST turbulence model re-defined in PANS form? Are there additional assumptions that arise with the derivation? Are these assumptions valid?
- What shortcomings do the two data-driven methods possess? Are there solutions and can they be overcome?
- Can $k-$corrective frozen RANS approach be applied onto PANS with theoretically valid correction terms? What limitations of PANS do these corrections represent?
- Is the combination of DSCS and $k-$corrective frozen RANS valid in theory?
- Does the HiFi large eddy simulation (LES) dataset well represent the flow case? Has it been validated? Is it sufficient to achieve the research objective?
- What are the characteristics of a triangular prism that makes it a viable bluff body to be studied on? Is there a sufficient number of studies conducted for triangular prism at similar Re to use as a reference?
- Is there a clear relationship between f_k and the correction terms? Can a statistical relationship be found?
- Do the obtained correction terms well represent the limitations of PANS? Which correction term is the most influential? In other words, which part of the PANS turbulence model is the furthest from reality?

1.3. Thesis outline

The report begins with a set of literature reviews for the various involved subjects of the thesis before moving on to methodologies of the combination of the data-driven methods and the results obtained.

Chapter 2 briefly covers the basics of turbulence and the reason for requiring turbulence modelling including its state of the art with the $k - \omega$ SST RANS model, the turbulence model under interest in this project, explicitly stated. Next, PANS is studied in Chapter 3 wherein the motivation for its development and some of its characteristics as well as technical features are presented. The Chapter is concluded with an extensive derivation of $k_u - \omega_u$ SST model, a re-defined $k - \omega$ SST model for PANS.

For the last part of the literature review, data-driven methods for turbulence modelling are explored in Chapter 4. Machine learning algorithms are briefly covered in this Chapter to present the common algorithms used in the computational fluid dynamics (CFD) field. However, they are not dwelt into as machine learning is not the focus of this thesis. More importantly, the closure for the turbulence models is provided where different data-driven methods use varying correction terms to "close" the baseline turbulence model that gives a poor prediction of turbulent flows.

The methodology of the project is kicked off with Chapter 5 which gives an overview of the HiFi dataset that is used in this thesis project. Its fluid domain and the mesh that is used for the domain are featured. Additionally, the important flow conditions are extracted and recorded. Chapter 6 walks through how PANS is implemented into OpenFOAM, an open-source CFD software, alongside the mesh that is produced to support the turbulence model. Boundary and initial conditions are set with reference to those used in the HiFi dataset. A pseudo-code is given for an overall idea of how the turbulence model is implemented and the implementation is validated together with other relevant results. Finally, the combination of the two data-driven methods for PANS is covered in Chapter 7. The Chapter begins with pre-processing of the HiFi dataset and the altered PANS for data injection is stated afterwards together with a flow-chart giving an overview of the entire process. Another pseudo-code is given for an explanation of the code implemented into OpenFOAM and the Chapter is concluded with a set of obtained results. The thesis is then wrapped up with a conclusion and recommendations for future work in Chapter 8.

2

Turbulence modelling

Turbulence modelling is built from the fundamental understanding and theories of turbulence and these are covered in Section 2.1. This is followed by Section 2.2 in which a few of the commonly used turbulence models in the community which are relevant for the project are introduced.

2.1. Fundamentals of turbulence

For a holistic understanding of turbulence modelling, it is crucial to have the basic theory of turbulence and its characteristics laid out. Only then can one have an idea of the roles various theories and techniques play in forming the pillars of CFD. Thus, some of these ground theories of turbulence are covered in this section before moving on to the diverse methods that have been developed so far from these concepts.

2.1.1. Characteristics of turbulence

Turbulence by definition is a random and chaotic behaviour. Although the former is a debatable definition and will be covered why so in the latter part of the report, turbulence is inarguably a chaotic behaviour. This unstable reaction generally occurs in a fluid body with a sufficiently high speed as it is sheared by a solid boundary like a wall. Such a behaviour results in redistribution and disintegration of velocity into 3D correlated eddies that are rotational and they evolve continuously, existing in a spectrum of sizes as it was first discovered by Leonardo da Vinci around the year 1510. As such, vortices and eddies make up turbulence and this process demonstrates the non-linear behaviour of the widely known incompressible N-S equations of (2.2) and (2.3) which are presented in Section 2.1.3.

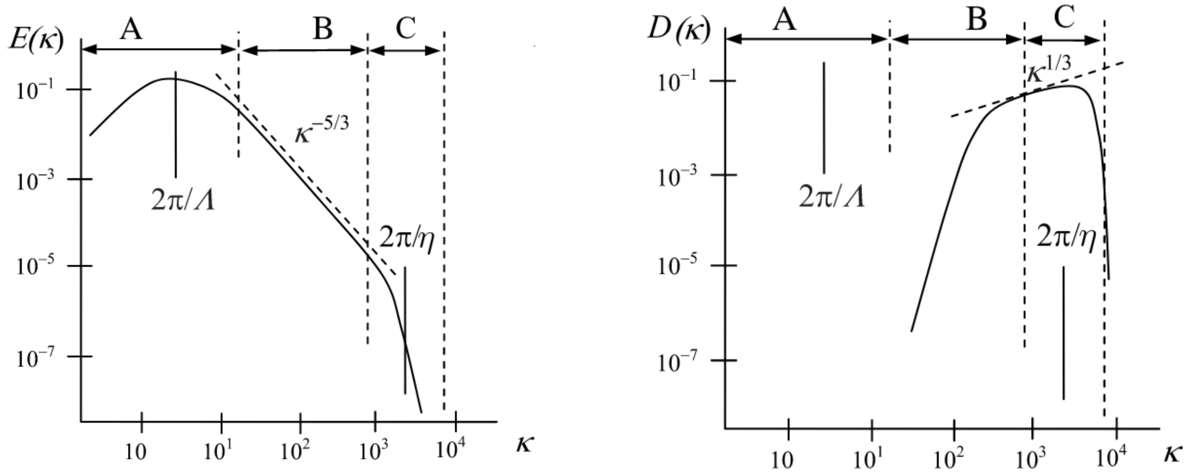
Beneath such chaotic nature of turbulence lives structural order and organisation due to the coexistence of turbulence scales in various sizes which will be furthered in Section 2.1.2. Hence, the unsteadiness in turbulent flows exists in a large spectrum of frequencies and those with lower frequencies are deemed to represent clearly observable coherence or patterns in turbulent flows. This allows for the concept of decomposing fluid velocity and it will be furthered in Section 2.2.1.

2.1.2. Eddies

As aforementioned, a spectrum of eddies in various sizes make up turbulence. These sizes can be categorised into three ranges as shown in Figure 2.1 in which η represents the smallest and Λ represents the largest eddy size in a specific turbulent flow field and each range has its own characteristics and part to play in a turbulent field. Additionally, κ represents the wave number defined as

$$\kappa := \frac{2\pi}{\ell} \tag{2.1}$$

where ℓ is an arbitrary eddy size and this wave number is linearly correlated to the frequency in a uniform velocity flow.



(a) Turbulence kinetic energy $E(\kappa)$ spectrum

(b) Turbulence dissipation $D(\kappa)$ spectrum

Figure 2.1: Distribution of $E(\kappa)$ and $D(\kappa)$ throughout spectrum of eddies in a turbulent flow with sufficiently high Re[4]

For the turbulence kinetic energy spectrum shown in Figure 2.1a, range **A** represents the large eddies and the largest length scale Λ which contains the most amount of energy due to the production of turbulence energy occurring in this range from external forces such as fluid shear. Range **B** represents the inertial subrange in which energy cascade $-$ the transfer of energy from the largest eddies to the smallest eddies $-$ occurs due to vortex stretching that incurs instability and eventually breaks the large eddies down into smaller sizes. Lastly, range **C** represents the smallest eddies and the smallest length scale η which are heavily influenced by viscosity. As for the turbulence dissipation energy shown in Figure 2.1b, dissipation is not taken up by the largest eddies Λ and is gradually taken up with the increase in wave number κ and is mostly taken up by the smallest eddies η through dissipation of energy into other forms of energy such as heat.

2.1.3. Governing equations

The equations that form the backbone of fluid mechanics are the conservation equations for mass, momentum and energy for a fluid body and these are called as the Navier-Stokes (N-S) equations as a whole. The equations that represent conservation of mass and energy are presented in (2.2) and (2.3) respectively as follows:

$$\nabla \cdot \boldsymbol{u} = 0 \tag{2.2}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\nabla \cdot \boldsymbol{u} = -\frac{1}{\rho}\nabla p + \nu \Delta \boldsymbol{u}, \tag{2.3}$$

where $\boldsymbol{u}$ is velocity, ρ is density, p is pressure and ν is kinematic viscosity which can be expressed as

$$\nu = \frac{\mu}{\rho}$$

where μ is dynamic viscosity that is a constant property. Furthermore, these equations assume flow incompressibility and constant fluid density which is a valid assumption at a reasonably low Reynolds number Re. These assumptions allow for the absence of the energy conservation equation as the momentum equation also implies conservation of energy. The N-S equations have a problem that there are only four equations which includes the continuity equation of 2.2 and the three components of the momentum equation stated in (2.3) in 3D domain while there are five unknown variables: the three velocity components $\boldsymbol{u}$, pressure p and density ρ. Hence, this poses a closure problem in which turbulence modelling finds its purpose.

2.2. Turbulence modelling

Turbulence modelling arose from the realisation that turbulent flows can be statistically analysed and the basis of this approach is the Reynolds-averaging on which many other theories and models were developed.

2.2.1. Statistical turbulence

The first record of statistical turbulence modelling dates back to 1894 whereby Obsborne Reynolds suggested in [5] the decomposition of velocity field into its mean $\overline{u}$ and the relative mean u' which is more referred to as 'fluctuation' in more recent literatures. This Reynolds decomposition is presented as

$$u_i = \overline{u}_i + u_i' \tag{2.4}$$

where u_i represents the i-th velocity component.

Additionally, the velocity mean $\overline{u}$ can be interpreted as Favre-averaged velocity which is equivalent to the ensemble-averaged velocity at any location in turbulent flow that is weighted by density as defined in [6]:

$$\overline{u}_i \coloneqq \frac{1}{N\overline{\rho}} \sum_{n=1}^{N} \rho u_i \, , \tag{2.5}$$

where N represents a sufficiently large number of samples over a long period and $\overline{\rho}$ is mean fluid density. For constant density assumption that has already been made in the N-S equations, (2.5) is simplified into

$$\overline{u}_i = \frac{1}{N} \sum_{n=1}^{N} u_i. \tag{2.6}$$

The mean velocity $\overline{u}$ can also be represented using time average over an infinitely long time interval, T:

$$\overline{u}_i = \frac{1}{T} \int_{t-T/2}^{t+T/2} u_i \left(t'\right) dt'. \tag{2.7}$$

Just like it has been done for velocity in (2.4), pressure is also decomposed into its mean and fluctuating parts:

$$p = \overline{p} + p'. \tag{2.8}$$

2.2.2. Reynolds-averaged Navier-Stokes

Prior to setting up Reynolds-averaged Navier-Stokes (RANS) equations, a few Reynolds-averaging rules are stated which are utilised in the derivation and they are:

$$\begin{aligned}
\overline{u'} &= 0, \\
\overline{cu} &= c\overline{u}, \\
\overline{u+v} &= \overline{u} + \overline{v}, \\
\overline{\overline{u}v} &= \overline{u}\,\overline{v} \ \text{ and} \\
\overline{\frac{\partial u}{\partial t}} &= \frac{\partial \overline{u}}{\partial t}
\end{aligned} \tag{2.9}$$

where c represents a constant while u and v represent variables. These rules are strictly valid for variables decomposed in terms of Reynolds decomposition stated in (2.4). Using the Reynolds-averaging rules presented, the RANS equations for steady flow can be derived by substituting Reynolds-decomposed velocity and pressure in (2.4) and (2.8) respectively into N-S equations (2.2) and (2.3). For steady flows, the time interval T that is used to time average the flow velocity as demonstrated in (2.7) is a large value that is much larger than the largest time scale of turbulent motion thus averaging out every characteristic of a turbulent flow. The resulting RANS equations are:

$$\frac{\partial \rho \overline{u}_j}{\partial x_j} = 0 \tag{2.10}$$

$$\frac{\partial \rho \overline{u}_i \overline{u}_j}{\partial x_j} = -\frac{\partial p}{\partial x_j} + \frac{\partial}{\partial x_i}\mu\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right) - \frac{\partial}{\partial x_j}\rho\overline{u_i' u_j'}, \tag{2.11}$$

where $\overline{u_i' u_j'}$ are the Reynolds stresses, often presented as τ_{ij} in literature, with six independent components that need to be closed and defined using turbulence models which are covered in Section 2.2.5. However in this report, RS_{ij} is used to represent the Reynolds stresses term instead of τ_{ij} which is introduced in Section 3.1.2 as generalised central second moment. The difference comes from the extent of averaging wherein RS_{ij} implies a full averaging while τ_{ij} implies averaging of any extent. RS_{ij} is hence defined as

$$RS_{ij} := \overline{u_i' u_j'}. \tag{2.12}$$

2.2.3. Unsteady RANS

Turbulent flows, although defined to be random and chaotic, still reflect coherent structures that are periodic with each of their characteristic time scales which are of a similar order to low-frequency turbulent motions [7]. This feature is mathematically presented using triple decomposition as first introduced in [8] that is furthered from Reynolds decomposition of (2.4) and it is expressed as:

$$u_i = \overline{u}_i + \widetilde{u}_i + u_i'' \tag{2.13}$$

where $\widetilde{u}_i$ represents the periodic unsteadiness that is deterministic while u_i'' is the stochastic unsteadiness. The triple decomposition is visualised in Figure 2.2.

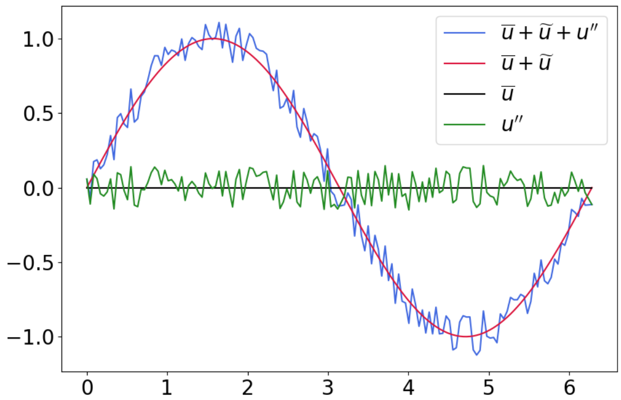


Figure 2.2: Triple decomposition

The distinction is most apparent in flows around bluff bodies such as a square cylinder in which periodic shedding of vortices, also known as von Kármán vortex street, can be observed behind the body due to flow separation as shown in Figure 2.3 taken from [9]. This von Kármán vortex street features the periodic unsteadiness of low frequencies in the wake of the flow behind the circular cylinder in which stochastic unsteadiness of much higher frequencies lives as it can be seen within the large vortices in the figure.

To accommodate for this feature and to resolve the unsteady mean-motion of the flow field which includes the periodic unsteadiness to an arbitrary extent, a finite time averaging should be applied instead of an infinitely long time interval that was demonstrated in (2.4). With $\widehat{u}$ representing this finite time averaging of the flow velocity which is the sum of the mean velocity and the periodic unsteadiness:

$$\widehat{u} := \overline{u} + \widetilde{u}, \tag{2.14}$$

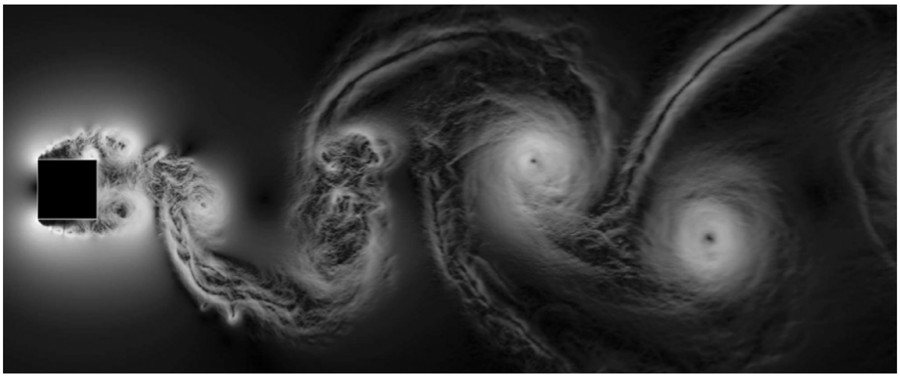


Figure 2.3: Vortex street behind a square cylinder at $Re = 22\,000$ [9]

(2.11) is rewritten as

$$\rho\frac{\partial \widehat{u}_i}{\partial t} + \rho\frac{\partial \widehat{u}_i\widehat{u}_j}{\partial x_j} = -\frac{\partial \widehat{p}}{\partial x_i} + \frac{\partial}{\partial x_j}\mu\left(\frac{\partial \widehat{u}_i}{\partial x_j} + \frac{\partial \widehat{u}_j}{\partial x_i}\right) - \rho\frac{\partial}{\partial x_j}\widehat{u_i'u_j'}, \tag{2.15}$$

which now includes a time derivative term and this represents the URANS equation.

2.2.4. Eddy viscosity

As previously mentioned, implementation of Reynolds-decomposition into the N-S equations brings about six Reynolds stress components and because the fluctuating components u_i' are not calculated directly in RANS, initial modelling is introduced whereby they are related to the mean flow variables $\overline{u}_i$. Thereafter, the role of turbulence models is to relate these stresses to other known flow variables.

Eddy viscosity is a concept proposed by Boussinesq [10] to explicitly express the Reynolds stresses of the RANS equations in terms of the mean velocity gradient. It was suggested by Boussinesq that the behaviour of turbulence can be seen as an analogy to the Brownian motion. This is ultimately a flawed analogy as it omits the spatial coherence of turbulence structures such as two-point correlations and vortical motions [4]. Despite such a flaw, it can be a useful model as it incurs low computational costs and gives a high convergence rate while providing practical results for studies.

This concept is based on the addition of a turbulent viscosity μ_t, also referred to as the eddy viscosity, that is dependent on some properties of the flow to represent the effect of turbulence mixing and diffusion with which shear stress in a simple shear layer is expressed as:

$$-\rho\overline{u'v'} \simeq \mu_t\frac{\partial \overline{u}}{\partial y}. \tag{2.16}$$

μ_t is an unknown artificial constant of proportionality that controls the strength of diffusion (transport of momentum between fluid particles) [11] and it thus requires modelling. Taking into account that the order of multiplication does not matter ($\overline{u'v'} = \overline{v'u'}$) and implementing Einstein notation, a general form of (2.16) can be written as:

$$-\rho\overline{u_i'u_j'} = \mu_t\left(\frac{\partial \overline{u}_i}{\partial x_j} + \frac{\partial \overline{u}_j}{\partial x_i}\right). \tag{2.17}$$

When compared to the RANS equation of (2.11), it can be observed that the eddy viscosity plays a similar role as the viscous stresses. However it is not possible to achieve a single function for scalar μ_t that can satisfy all components of the Reynolds stress. Therefore, a correction is required to make up for this inadequacy.

Moreover, since turbulence kinetic energy k is expressed in terms of u_i' as

$$k = \frac{1}{2}\overline{(u_i')^2} \tag{2.18}$$

and it is possible for k to be present even in uniform (strain-free) and incompressible flow, k needs to be accounted for in the representation of Reynolds stress. With the addition of k, (2.17) is extended to

$$-\rho\overline{u_i'u_j'} \approx \mu_t\left(\frac{\partial\overline{u}_i}{\partial x_j} + \frac{\partial\overline{u}_j}{\partial x_i} - \frac{2}{3}\frac{\partial\overline{u}_k}{\partial x_k}\delta_{ij}\right) - \frac{2}{3}\rho k\delta_{ij}$$

which can be rewritten as

$$RS_{ij} \approx -\nu_t\left(\frac{\partial\overline{u}_i}{\partial x_j} + \frac{\partial\overline{u}_j}{\partial x_i} - \frac{2}{3}\frac{\partial\overline{u}_k}{\partial x_k}\delta_{ij}\right) + \frac{2}{3}k\delta_{ij} \tag{2.19}$$

and this is the complete form of the Boussinesq hypothesis wherein kinematic eddy viscosity ν_t is

$$\nu_t = \frac{\mu_t}{\rho} \tag{2.20}$$

and it will be referred to as eddy viscosity henceforth. To compress and simplify this expression, the mean strain rate tensor S_{ij} is introduced:

$$S_{ij} = \frac{1}{2}\left(\frac{\partial\overline{u}_i}{\partial x_j} + \frac{\partial\overline{u}_j}{\partial x_i}\right) \tag{2.21}$$

with which (2.19) can be re-expressed as

$$RS_{ij} \approx -2\nu_t\left(S_{ij} - \frac{1}{3}\frac{\partial\overline{u}_k}{\partial x_k}\right) + \frac{2}{3}k\delta_{ij}.$$

With the aforementioned incompressibility assumption that implies zero velocity divergence, the above equation is further simplified into

$$RS_{ij} \approx -2\nu_t S_{ij} + \frac{2}{3}k\delta_{ij}. \tag{2.22}$$

Additionally, $2/3\ k$ is the average normal stress which thus implies that the term with eddy viscosity accounts for the deviations from this average value. It should be noted that due to the constant density assumption, the RS_{ij} is expressed without the product of density ρ thus not having the same dimensions as stress. (2.22) can also be expressed as

$$RS_{ij} \approx a_{ij} + \frac{2}{3}k\delta_{ij}$$

where a_{ij} is the anisotropy tensor:

$$a_{ij} = -2\nu_t S_{ij} \tag{2.23}$$

bridging the stress and the strain rate tensor, akin to viscous stress tensor in the linear constitutive equation of Newtonian fluids [12]. Normalising a_{ij} by k results in b_{ij}:

$$b_{ij} = -\frac{\nu_t}{k}S_{ij} \tag{2.24}$$

resulting in zero trace, a set of eigenvalues that sum up to zero and eigenvectors that form optimal basis which describes vector space of b_{ij} [13]. Hence, (2.22) is often presented as

$$RS_{ij} \approx 2k\left(b_{ij} + \frac{1}{3}\delta_{ij}\right). \tag{2.25}$$

2.2.5. Eddy viscosity models

In this section, some of the more commonly used eddy viscosity models are presented. There exist several models that have been developed over the years and some have been proven to produce commendable results, some not so. This has led to users of RANS only making use of a handful number of them. These frequently used models can be subdivided into two categories: one- and two-equation models. As their names suggest, one-equation models use just a single equation to solve the turbulent eddy viscosity term whereby two-equation models handle an extra equation.

One-equation model

Starting with the one-equation model, the Spalart-Allmaras model (1994) developed by Spalart and Allmaras [14] is currently the most extensively used model in this category and it will be briefly introduced as this model will not be furthered in this report. Due to the absence of the turbulent kinetic energy in this model, the Reynolds stresses are represented by (2.22) is reduced to (2.17). It introduces a new variable called the modified eddy viscosity $\widetilde{\nu_t}$ that is closely related to the kinematic eddy viscosity ν_t and their behaviour close to the wall is shown in Figure 2.4 taken from [15].

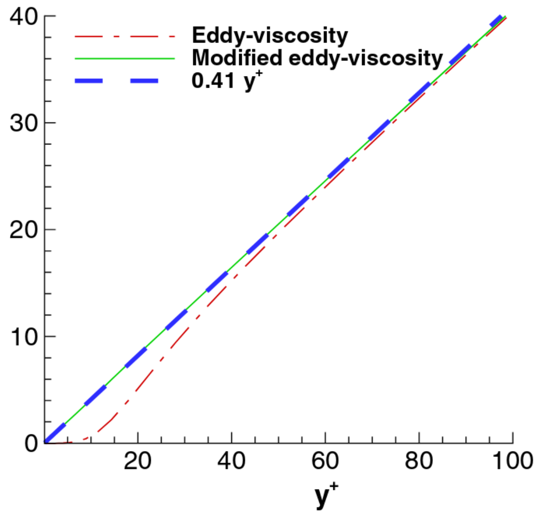


Figure 2.4: (Kinematic) eddy viscosity ν_t and modified eddy viscosity $\widetilde{\nu}$ for S-A model adapted from [15]

The modified eddy viscosity term has a linear relationship with y^+ which overcomes the problem of conventional eddy viscosity having a fourth order relationship with y^+ in the viscous sub-layer ($y^+ < 5$) that requires a large number of cells close to the wall to resolve the flows in the region. The single equation involved is the differential transport equation of this modified eddy viscosity.

Ever since this first version of the S-A model was introduced, many tweaks have been made by various researchers to suit the needs of flow fields under their studies and they have released their own variants such as the S-A model without f_{t2} term (one of the many terms in S-A model) [16] and S-A model for rotating and curved channels [17]. The S-A model and its other variants will not be covered in greater depth in this report due to the absence of turbulent kinetic energy k term that makes them unsuitable models for the main study of PANS which is based on k and it is covered in Chapter 3.

Two-equation models

The two-equation models are the more common types of eddy viscosity models and they are the industry standard. They differ from one-equation counterparts in the consideration of how turbulent length scale evolves throughout the flow field domain using flow properties such as the turbulence kinetic energy which relates to the eddy sizes as described in Section 2.1.2 via a transport equation.

One of the oldest two-equation models is the Jones-Launder $k-\varepsilon$ model from 1972 [18]. Today, Chien's version of the $k-\varepsilon$ model (1982) [19], which is primarily based on Launder-Sharma model (1974) [20], is the most commonly used version. As the name suggests, the two equations that are involved in this model are the turbulence kinetic energy k and dissipation rate ε differential transport equations. ε was introduced to replace the explicitly and algebraically defined mixing length l_m in the mixing length model developed by Prandtl in 1926 and the improved Van Driest mixing model [21]. The transportation equation for ε is solved instead in which the empirically defined model coefficients vary between the model versions. These coefficients are damped instead of the mixing length, each with its damping function, to allow for wall-resolving calculation.

Using these two parameters the eddy viscosity ν_t can be expressed:

$$\nu_t = \beta^* \frac{k^2}{\varepsilon}, \tag{2.26}$$

where $\beta^* = 0.09$. This $k - \varepsilon$ model is only valid for fully turbulent flows. Although it is a relatively easy model to implement and is computationally inexpensive, a major flaw that it possesses is its use of only a single turbulent length scale for dissipation calculation [22] which contradicts Figure 2.1b from which it was previously observed that the inertial subrange also participates in the dissipation. It also performs poorly for flow fields with adverse pressure gradients, separation and highly curved streamlines, resulting in the wrong separation point at the wrong angle for the boundary layers [22, 4]. Despite its shortcomings, the $k - \varepsilon$ model is still heavily used due to its commendable results away from the walls and its simple implementation.

Another heavily used model in the community is the $k - \omega$ model (2008) by Wilcox [23], first proposed in 1942 by Kolmogorov. In this model, specific dissipation rate ω, also known as the turbulence frequency, is utilised and it is defined as

$$\omega := \frac{1}{\beta^*} \frac{\varepsilon}{k}, \tag{2.27}$$

resulting in

$$\nu_t = \frac{k}{\omega}. \tag{2.28}$$

Instead of the ε transport equation, ω transport equation is used in which different empirical coefficients are used as compared to the $\varepsilon-$equation. Another difference is that there is no longer any damping function which performs poorly in the presence of adverse pressure gradients. Thus, the model is said to perform significantly better near walls and also for low Re flows. Additionally, this model is a lot more numerically stable than the $k - \varepsilon$ model [22, 24]. However, it is not all sunshine and rainbows, and the model features some flaws as well. Excessive and early flow separation is typically simulated, and to achieve better near-wall results, the model requires a high mesh resolution near the walls which makes it a much more computationally expensive model. Furthermore, it is highly sensitive to free-stream flow away from the walls, whereby a tiny change results in a large difference in eddy viscosity [25] which ultimately affects forces on the body and the flow separation point.

The advantages of these two models are taken and made into another model which goes by the name of $k - \omega$ SST, developed by Menter in [24] and further fine-tuned in [26, 27, 28]. It is a hybrid model that utilises Wilcox's $k - \omega$ model near the wall where its advantage is at and the standard $k - \varepsilon$ model away from the wall to avoid being sensitive in the free-stream areas using a blending function F_1.

$k - \omega$ SST

The classical model from [24] is given by the following two equations:

$$\rho \frac{\partial k}{\partial t} + \rho \overline{u}_j \frac{\partial k}{\partial x_j} = P_k - \beta^* \rho \omega k + T_k \quad \text{and} \tag{2.29}$$

$$\rho \frac{\partial \omega}{\partial t} + \rho \overline{u}_j \frac{\partial \omega}{\partial x_j} = \frac{\gamma}{\nu_t} P_k - \beta \rho \omega^2 + T_\omega + \rho(1 - F_1) CD_{k\omega}, \tag{2.30}$$

where

$$T_k = \frac{\partial}{\partial x_j}\left[\left(\mu + \sigma_k \mu_t\right)\frac{\partial k}{\partial x_j}\right], \tag{2.31}$$

$$T_\omega = \frac{\partial}{\partial x_j}\left[\left(\mu + \sigma_\omega \mu_t\right)\frac{\partial \omega}{\partial x_j}\right] \tag{2.32}$$

$$CD_{k\omega} = 2\frac{\sigma_{\omega 2}}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, \tag{2.33}$$

$$P_k = \rho R S_{ij}\frac{\partial \overline{u}_i}{\partial x_j} \text{ and}$$

$$\beta^* = 0.09.$$

Since isotropic component has no effect in momentum transport [29], P_k can be simplified into

$$P_k = \rho \nu_t S_{ij}\frac{\partial \overline{u}_i}{\partial x_j}.$$

Additionally F_1 is the blending function that blends the model between $k-\epsilon$ and $k-\omega$ models and it is expressed as

$$F_1 = \tanh\left(\arg_1^4\right) \in [0,1], \tag{2.34}$$

$$\arg_1 = \min\left[\max\left(\frac{\sqrt{k}}{\beta^*\omega d}, \frac{500\nu}{d^2\omega}\right), \frac{4\rho\sigma_{\omega 2}k}{CD_{k\omega}\, d^2}\right], \tag{2.35}$$

where d is the distance from the point in the flow field to the nearest wall. It can be observed that the value of F_1 thus varies for every cell in the fluid domain, varying the extent of blending at every available mesh point. Away from the wall, when $F_1 = 0$, the transport equation is equivalent to that of $k-\epsilon$ model and near the wall, when $F_1 = 1$, it converts into $k-\omega$ model.

The blending function also blends the constants and it requires two values: inner-1 and outer-2 values. For an arbitrary constant ϕ, the two values are blended by

$$\phi = F_1\phi_1 + \left(1 - F_1\right)\phi_2.$$

The inner and outer values of the remaining constants are:

$$\gamma_1, \gamma_2 = (5/9, 0.44),$$
$$\sigma_{k1}, \sigma_{k2} = (0.85, 1.0),$$
$$\sigma_{\omega_1}, \sigma_{\omega_2} = (0.5, 0.856) \text{ and}$$
$$\beta_1, \beta_2 = (0.075, 0.0828).$$

The inclusion of blending function F_1 alone gives the Menter baseline stress transport (BST) model of [24] which was found to still over-predict wall shear stress. Hence in [26], adjustment was made to μ_t to improve the wall shear stress prediction by introducing another blending function F_2 into eddy viscosity term with which the $k-\omega$ SST model was formulated. The newly proposed eddy viscosity term is

$$\mu_t^+ = \rho\frac{a_1 k}{\max\left(a_1\omega, W F_2\right)} \tag{2.36}$$

wherein the blending function F_2 is

$$F_2 = \tanh\left(\arg_2^2\right) \text{ and} \tag{2.37}$$

$$\arg_2 = \max\left(2\frac{\sqrt{k}}{\beta^*\omega d}, \frac{500\nu}{d^2\omega}\right). \tag{2.38}$$

Furthermore, P_k and $CD_{k\omega}$ are adjusted to improve convergence behaviour in computational calculations in CFD programs using limiters and they were updated to

$$P_k^+ = \min\left(P_k, 10\beta^*\rho k\omega\right) \quad \text{and} \tag{2.39}$$

$$CD_{k\omega}^+ = \max\left(2\frac{\sigma_{\omega 2}}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j}, 10^{-20}\right), \tag{2.40}$$

where $a_1 = 0.31$,

$$W = \sqrt{2\Omega_{ij}\Omega_{ij}} \quad \text{and} \tag{2.41}$$

$$\Omega_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right).$$

With the advantages of two parent models, the $k-\omega$ SST model is the most commonly used turbulence model for RANS and gives improved predictions of flow separations above other RANS models. However, since it uses $k-\omega$ model near the walls, it still requires high mesh resolution in those regions just like its parent model. Although much improvement is made from standard $k-\varepsilon$ model, it is still not suitable for flows with large normal strain and for regions with significant level of acceleration or deceleration. The model is however still a very useful model for domains with separating flows.

3
PANS

PANS is an acronym for partially-averaged Navier-Stokes equations. It is a relatively new modelling theory introduced in the year 2006 by Girimaji in [30] and it is actively being researched with a lot of room for improvements and discoveries. In this chapter, the basic idea of PANS is introduced together with commonly used methods of application.

3.1. Introduction to PANS

In this section, the fundamental principles of PANS are given. These are the basic theories that help build PANS into a new set of turbulence models without deviating from the fundamental laws of physics that are prerequisites for any model.

3.1.1. Motivation

Today, the wall-modelled LES, one of the many bridging methods, has emerged as the workhorse of the fluid dynamics industry due to its ability to mitigate the severe disadvantages of RANS and LES. RANS is computationally inexpensive on the bright side but more importantly, the results it gives are highly inadequate while LES gives sufficiently satisfactory results but is computationally too demanding. WMLES, as its name suggests, models the near-wall regions as resolving these regions requires a substantial increase in computational power due to the small-sized eddies that live in these regions that impose the need for using highly refined mesh while resolving the regions away from the walls. Many other bridging methods such as DES and VLES are given attention for the same reasons that there is a need for methods that only resolve the largest turbulent scales that are important for learning the precise physical interpretation of the flow field while modelling the smaller scales in the inertial subrange just to account for their effects using modelling methods such as RANS.

This idea of bridging methods is not new and the need for them was first realised and suggested by Speziale in [1]. Such a method utilises explicitly sized variable filter width to selectively resolve and model different turbulent length scales. Speziale mentioned three qualities in [1] that such a bridging method must possess:

1. The method should function as DNS when the entire spectrum of turbulent scales is resolved.
2. The method should also then function as RANS when the cutoff wave number is in the largest of the turbulent scales.
3. When this cutoff is in the turbulent scales of inertial sub-range, the method should function as an implicit LES subgrid scale model.

3.1.2. Characteristics of PANS

The development of PANS follows from the motivations of Section 3.1.1 and was introduced in [30] to offer an alternative bridging model that unlike many the other models, follows from and abides by the first principles of physics. Its two main parameters are the ratios of unresolved-to-total turbulence dissipation rate f_ε and kinetic energy f_k which control the filter width of PANS and they can take up any value between 0 (entirely resolved) and 1 (entirely modelled).

PANS aims to make use of existing RANS models and convert them to PANS form. This way, the physical effect of PANS when chosen to be fully modelled, matches its RANS counterpart, fulfilling one of the aforementioned properties that Speziale has pointed out to be required by the bridging methods. Thus, for values of the ratios less than 1, more turbulent scales are resolved just like other bridging methods. However, what sets PANS apart from the other bridging methods is the use of turbulence kinetic energy to decompose the turbulent flow field instead of the cutoff wave number as highlighted by [30]. Another characteristic of PANS is that the filtering is implemented implicitly without the need for an explicit filtering operation like top-hat filter in LES for example. Additionally, the physical resolution which is the filter width imposed by the ratios and the numerical resolution in the case of PANS are not dependent on each other.

3.1.3. The filtering approach

The fundamental idea of PANS is the partial-averaging and thus angular brackets are introduced to represent the partially averaged flow variables, such as $\langle u_i \rangle$ and $\langle p \rangle$ for partially averaged velocity and pressure respectively. Analogous to (2.7), The partial averaging operator can be defined as

$$\langle u_i(t) \rangle \coloneqq \frac{1}{\mathcal{T}} \int_{t-\mathcal{T}/2}^{t+\mathcal{T}/2} u_i\left(t'\right) dt' \tag{3.1}$$

where $0 < \mathcal{T} < T$ for T is an infinitely long time interval. Applying this operator to (2.3),

$$\rho \frac{\partial \langle u_i \rangle \langle u_j \rangle}{\partial x_j} = -\frac{\langle p \rangle}{\partial x_j} + \frac{\partial}{\partial x_j} \mu \left(\frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial \langle u_j \rangle}{\partial x_i} \right) - \rho \frac{\partial}{\partial x_j} \tau\left(u_i, u_j\right) \tag{3.2}$$

is achieved where $\tau\left(u_i, u_j\right)$ is not the Reynolds stress but is the generalised central second moment. Since Reynolds decomposition shown in (2.4) that applied full averaging is not applied anymore, the field is split into filtered and sub-filtered instead of mean and fluctuation. From [31], $\tau\left(u_i, u_j\right)$ is defined as

$$\begin{aligned} \tau\left(u_i, u_j\right) &\coloneqq \langle u_i u_j \rangle - \langle u_i \rangle \langle u_j \rangle \\ &\coloneqq \left\langle u_i' u_j' \right\rangle, \end{aligned} \tag{3.3}$$

which when the partial averaging is extended to full averaging, Reynolds stress of (2.12) is recovered. Despite their differences, this generalised central second moment shares similar properties with Reynolds stress as claimed in [30] and thus the sub-filter unresolved turbulence kinetic energy k_u and dissipation rate ε_u (with the subscript u representing unresolved field) can be defined as

$$k_u \coloneqq \frac{1}{2} \tau\left(u_i, u_i\right) \quad \text{and} \tag{3.4}$$

$$\varepsilon_u \coloneqq \nu \tau \left(\frac{\partial u_i}{\partial x_j}, \frac{\partial u_j}{\partial x_i} \right). \tag{3.5}$$

When the values of PANS ratios are 1, replicating RANS, the filtered scales become the average field while the sub-filter scales (SFS) become the fluctuations. Thus, it can be said that the filtration acts as averaging in this case and when substituted into (3.2), it gives back the RANS equation of (2.11). This feature of the N-S equations is called the averaging-invariance property [31].

Using the set of guidelines set by Speziale in [1], Girimaji has also defined three constraints and objectives for PANS [30]:

1. Smooth transition from RANS to DNS controlled by the ratios.
2. These ratios that act as filter-width controllers must be explicitly identified.
3. The structure of the PANS closure model should not change with the values of the ratios.

When these constraints are taken into account many bridging models such as URANS which has an unclear filter-width controller become invalid and PANS aims to replace such a drawback.

3.1.4. Choices of $\mathbf{f_k}$

Unlike the dissipation rate wherein the assumption that the majority of the dissipation occurs in the smallest of the turbulence scales is a highly valid and an acceptable one, it is not as simple of a choice for the turbulence kinetic energy. Hence, most if not all PANS study including [32, 33] set the value of f_ε to 1 while the values of f_k are based on numerous definitions and interpretations. In this section, these different versions of the definitions of f_k are presented together with their performances. There are three categories that the value of f_k can be based on: (i) spatially and temporally varying; (ii) spatially varying and temporally constant; (iii) spatially and temporally constant.

(i) Spatially and temporally varying

Since turbulent eddies of various scales move throughout the field domain, this variant makes the most sense to be implemented as it was done in [34, 35]. In this literature, the field values of f_k are calculated using their respective authors' interpretations of the parameter and how much of the turbulent scales they wish to resolve whilst also considering the grid resolutions their computational resources can handle. Thus, many of these interpretations involve characteristic turbulent length scales and relate them to cell sizes as well as temporal resolution.

Although this category makes good theoretical sense to be an easy choice for implementation, the various definitions of f_k are based on the authors' own judgements and interpretations. For example, cell dimension $-$ often labelled as Δ $-$ is a common parameter that appears in the variety of definitions of f_k and a large number of studies including [36] does not specify whether it is the average, maximum or minimum length of the cell leaving a significant amount of ambiguity. Even when Δ is explicitly defined as done in [35], the reasoning behind the choice is lacking clarity. Thus, such unvalidated definitions introduce model errors as demonstrated in [37].

(ii) Spatially varying and temporally constant

To save computing time and to simplify the model implementation, other studies such as [38, 39] have been done with just spatially varying but temporally constant values of f_k. Similarly to the flaws of the spatially and temporally varying counterparts, the field values of f_k were chosen not based on fully concrete and validated definitions but the authors' individual comprehensions as shown in [37], suffering from similar deficiencies to specially and temporally varying f_k definitions counterparts. This resulted in many studies even defying physics by letting the value go beyond 1 which implies that there is more unresolved turbulence kinetic energy than total.

(iii) Spatially and temporally constant

This implementation is the simplest version of all that is featured in a great number of works of literature including [40, 41]. A spatially and temporally fixed value of f_k is used depending on the domain-wide turbulent scales to be resolved and also the amount of computational power available at hand. The value itself does not impose greater computational power to be used but with more turbulent scales to be resolved, the mesh of the flow field needs to be refined accordingly which ultimately requires more computational power. Apart from these factors, the value of f_k is arbitrarily chosen without any basis.

Although PANS is a young turbulence closure method, a great number of studies and experiments have been conducted using this method and many have realised that the f_k fixed in space and time is the most optimal option due to the absence of commutation errors. Commutation error is a common form of error in bridging models. It arises due to an unphysical buffer layer near the interface of the models that are bridged and the mismatch between log-layers of these models, mainly resulting in significant under-prediction of skin-friction coefficient [42]. As for PANS, when f_k is varying throughout the fluid domain between every consecutive mesh cell, a different turbulence model is implemented in each cell, inducing commutation error at a large number of points.

Moreover, with spatially and temporally fixed f_k, there is no entanglement of numerical and modelling errors as clarified by [43]. In CFD, numerical error, also known as discretisation error, occurs from representing the governing equation such as the PANS equation in discrete space and time. Modelling error is the error from the approximation or assumption that a turbulence model such as the $k-\omega$ SST contains. When f_k is chosen based on spatial and temporal properties, the governing equations are then dependent on the resolutions in space and time resulting in the numerical error being ingrained into modelling error. This is not a desired property as an error analysis cannot be done independently for each type of error.

3.2. PANS $k_u - \omega_u$ SST

For this project, a turbulence model for PANS needs to be chosen and worked with for subsequent stages of the project to be built on and the $k-\omega$ SST was chosen. Thus in this section, the motivation behind this choice is given and its PANS form is derived.

3.2.1. Motivation

Since PANS can utilise existing RANS eddy viscosity models, it makes the most sense to choose the best performing model that is suitable for being transformed into PANS. Additionally, as mentioned in Section 3.1.2, PANS utilises the ratios of turbulence kinetic energy and dissipation rate. Thus, it is an inevitable requirement for the chosen model to cater for these two parameters. With all the advantages of $k-\omega$ SST model above other two-equation models in Section 2.2.5, it is an obvious choice.

3.2.2. Derivation

Derivations of PANS variants of $k-\epsilon$ and $k-\omega$ models are present in [30] and [44] respectively. However for $k-\omega$ SST that is of interest in the project, final result with $k-$equation and $\omega-$equation are often merely stated without the derivation as it was done in [3, 45]. Hence, an extensive derivation of the $k-\omega$ SST equations is done for verification and its procedure is presented in this report.

First, the explicit definitions of PANS parameters are:

$$f_k := \frac{k_u}{k}, \tag{3.6}$$

$$f_\varepsilon := \frac{\varepsilon_u}{\varepsilon} \text{ and} \tag{3.7}$$

$$f_\omega := \frac{\omega_u}{\omega} \tag{3.8}$$

where (3.8) is a new parameter that has not been introduced and is easily derived from the relationship between turbulence kinetic energy k, dissipation rate ϵ and specific dissipation rate ω shown in (2.27) and its unresolved form:

$$\omega_u = \frac{1}{\beta^*}\frac{\varepsilon_u}{k_u} \tag{3.9}$$

where the value of coefficient β^* was concluded not to require an alternation according to a fixed-point analysis done in [46] in which it was found that the value of β^* does not affect the energetics of the turbulence model.

Unresolved turbulence kinetic energy equation

First, (3.6) is applied onto the LHS of (2.29):

$$\rho\frac{\partial k_u}{\partial t} + \rho\overline{u}_j\frac{\partial k_u}{\partial x_j} = f_k\left(\rho\frac{\partial k}{\partial t} + \rho\overline{u}_j\frac{\partial k}{\partial x_j}\right) \tag{3.10}$$

in which k_u is following the progression of the mean velocity field $\overline{u}_j$ that is part of RANS. Thus, further modification is required for it to follow the filtered velocity field of PANS and attention is to be paid to the velocity field parameter.

Following from the form of RANS $k-$equation of (2.29), evolution equation for k_u can be simply expressed as

$$\rho\frac{\partial k_u}{\partial t} + \rho u_j\frac{\partial k_u}{\partial x_j} = P_{ku} - \beta^*\rho\omega_u k_u + T_{ku} \tag{3.11}$$

where the aim of the derivation is to discover an expression for the unresolved turbulence kinetic energy transport term T_{ku}. Substituting the RHS of (2.29) into the RHS of (3.10) and adding a term with instantaneous velocity field u_j and gradient of k_u on both sides,

$$\rho\frac{\partial k_u}{\partial t} + \rho u_j\frac{\partial k_u}{\partial x_j} = f_k\left[P_k - \beta^*\rho\omega k + T_k\right] + \left(u_j - \overline{u}_j\right)\frac{\partial k_u}{\partial x_j} \tag{3.12}$$

is obtained where $(u_j - \overline{u}_j)$ is the resolved velocity fluctuations as defined in (2.4).

Substituting (3.11) into the LHS of (3.12),

$$P_{ku} - \beta^*\rho\omega_u k_u + T_{ku} = f_k\left[P_k - \beta^*\rho\omega k + T_k\right] + \left(u_j - \overline{u}_j\right)\frac{\partial k_u}{\partial x_j} \tag{3.13}$$

from which the source/sink terms that represent the local processes are extracted to formulate

$$\begin{aligned} P_{ku} - \beta^*\rho\omega_u k_u &= f_k\left[P_k - \beta^*\rho\omega k\right] \\ P_k &= \frac{1}{f_k}\left(P_{ku} - \beta^*\rho\omega_u k_u\right) + \beta^*\rho\omega k \\ P_k &= \frac{1}{f_k}\left(P_{ku} - \beta^*\rho\omega_u k_u\right) + \beta^*\rho\frac{\omega_u k_u}{f_\omega f_k} \end{aligned} \tag{3.14}$$

using the relations in (3.6) and (3.8) where

$$P_{ku} = \rho\tau_{ij}\frac{\partial u_j}{\partial x_j}. \tag{3.15}$$

As for the non-local process that is taken up by the transport terms of (3.13):

$$\begin{aligned} T_{ku} &= f_k T_k + \left(u_j - \overline{u}_j\right)\frac{\partial k_u}{\partial x_j} \\ &= f_k\frac{\partial}{\partial x_j}\left[\left(\mu + \sigma_k\mu_t\right)\frac{\partial k}{\partial x_j}\right] + \left(u_j - \overline{u}_j\right)\frac{\partial k_u}{\partial x_j} \\ &= \frac{\partial}{\partial x_j}\left[\left(\mu + \sigma_k\mu_t\right)\frac{\partial k_u}{\partial x_j}\right] + \left(u_j - \overline{u}_j\right)\frac{\partial k_u}{\partial x_j} \\ T_{ku} &= \rho\frac{\partial}{\partial x_j}\left[\left(\nu + \sigma_k\nu_t\right)\frac{\partial k_u}{\partial x_j}\right] + \left(u_j - \overline{u}_j\right)\frac{\partial k_u}{\partial x_j} \end{aligned} \tag{3.16}$$

is setup with substitution of (2.31). Here, an assumption is made whereby at a sufficiently high Re, the resolved fluctuations have no contribution to the SFS energy transport [44]. Thus, (3.16) reduces to

$$T_{ku} = \rho\frac{\partial}{\partial x_j}\left[\left(\nu + \sigma_k\nu_t\right)\frac{\partial k_u}{\partial x_j}\right]. \tag{3.17}$$

With (2.28), (3.6) and (3.8), an expression for the kinematic eddy viscosity ν_t in terms of unresolved fields can be expressed using the ratio of total-to-unresolved kinematic eddy viscosity:

$$\begin{aligned}\frac{\nu_t}{\nu_{tu}} &= \frac{k/\omega}{k_u/\omega_u} \\ &= \frac{k}{k_u}\frac{\omega_u}{\omega} \\ &= \frac{f_\omega}{f_k} \\ \nu_t &= \nu_{tu}\frac{f_\omega}{f_k}. \end{aligned} \tag{3.18}$$

Substituting (3.18) into (3.17), the following is derived:

$$\begin{aligned} T_{ku} &= \rho\frac{\partial}{\partial x_j}\left[\left(\nu + \sigma_k \nu_{tu}\frac{f_\omega}{f_k}\right)\frac{\partial k_u}{\partial x_j}\right] \\ T_{ku} &= \rho\frac{\partial}{\partial x_j}\left[\left(\nu + \sigma_{ku}\nu_{tu}\right)\frac{\partial k_u}{\partial x_j}\right] \end{aligned} \tag{3.19}$$

where

$$\sigma_{ku} = \sigma_k\frac{f_\omega}{f_k}. \tag{3.20}$$

Substituting (3.19) into (3.11), the unresolved turbulence kinetic energy equation for PANS is derived:

$$\rho\frac{\partial k_u}{\partial t} + \rho u_j\frac{\partial k_u}{\partial x_j} = P_{ku} - \beta^*\rho\omega_u k_u + \rho\frac{\partial}{\partial x_j}\left[\left(\nu + \sigma_{ku}\nu_{tu}\right)\frac{\partial k_u}{\partial x_j}\right]. \tag{3.21}$$

It can be observed that the derived equation above is identical in form to the $k-$equation of (2.29) that is a part of the conventional $k-\omega$ SST model. The sole differences from the magnitude of the coefficients are altered due to the implementation of f_k.

Unresolved turbulence specific dissipation rate equation

Starting with the same steps taken for derivation of k_u- equation and starting off with applying (3.8) onto the LHS of (2.30):

$$\rho\frac{\partial \omega_u}{\partial t} + \rho\overline{u}_j\frac{\partial \omega_u}{\partial x_j} = f_\omega\left(\rho\frac{\partial \omega}{\partial t} + \rho\overline{u}_j\frac{\partial \omega}{\partial x_j}\right) \tag{3.22}$$

and substituting (2.30),

$$\begin{aligned} \rho\frac{\partial \omega_u}{\partial t} + \rho u_j\frac{\partial \omega_u}{\partial x_j} &= f_\omega\left[\frac{\gamma}{\nu_t}P_k - \beta\rho\omega^2 + T_\omega + \rho(1-F_1)CD_{k\omega}\right] \\ \rho\frac{\partial \omega_u}{\partial t} + \rho u_j\frac{\partial \omega_u}{\partial x_j} &= \underbrace{f_\omega\frac{\gamma}{\nu_t}P_k}_{A} - \underbrace{f_\omega\beta\rho\omega^2}_{B} + \underbrace{f_\omega T_\omega}_{C} + \underbrace{f_\omega\rho(1-F_1)CD_{k\omega}}_{D} \end{aligned} \tag{3.23}$$

is obtained in which the terms on the RHS have each been assigned an alphabet to assist in further derivation. Additionally, the assumption of resolved fluctuations having no contribution to the SFS energy transport is made again and the term $(u_j - \overline{u}_j)$ is neglected. Starting with A, parameter P_k is defined in (3.14) and ν_t is newly expressed in (3.18). Hence, A can be expanded into

$$\begin{aligned} A &= f_\omega\gamma\frac{f_k}{\nu_{tu}f_\omega}\left[\frac{1}{f_k}\left(P_{ku} - \beta^*\rho\omega_u k_u\right) + \beta^*\rho\frac{\omega_u k_u}{f_\omega f_k}\right] \\ &= \frac{\gamma f_k}{\nu_{tu}}\left[\frac{1}{f_k}\left(P_{ku} - \beta^*\rho\omega_u k_u\right) + \beta^*\rho\frac{\omega_u k_u}{f_\omega f_k}\right] \\ &= \frac{\gamma}{\nu_{tu}}\left(P_{ku} - \beta^*\rho\omega_u k_u\right) + \gamma\beta^*\rho\frac{\omega_u k_u}{f_\omega \nu_{tu}} \\ A &= \frac{\gamma}{\nu_{tu}}P_{ku} - \gamma\beta^*\rho\omega_u^2 + \gamma\beta^*\rho\frac{\omega_u^2}{f_\omega}. \end{aligned} \tag{3.24}$$

Next for the term B, (3.8) is simply implemented to give

$$B = \beta\rho\frac{\omega_u^2}{f_\omega}. \tag{3.25}$$

As for the term C, T_ω of (2.32) is substituted and using similar procedure as to deriving T_{ku} of (3.16) and it is as follows:

$$\begin{aligned}
C &= f_\omega\frac{\partial}{\partial x_j}\left[(\mu+\sigma_\omega\mu_t)\frac{\partial\omega}{\partial x_j}\right]\\
&= \frac{\partial}{\partial x_j}\left[(\mu+\sigma_\omega\mu_t)\frac{\partial\omega_u}{\partial x_j}\right]\\
&= \rho\frac{\partial}{\partial x_j}\left[(\nu+\sigma_\omega\nu_t)\frac{\partial\omega_u}{\partial x_j}\right]\\
C &= \rho\frac{\partial}{\partial x_j}\left[(\nu+\sigma_{\omega u}\nu_{tu})\frac{\partial\omega_u}{\partial x_j}\right],
\end{aligned} \tag{3.26}$$

where

$$\sigma_{\omega u} = \sigma_\omega\frac{f_\omega}{f_k}. \tag{3.27}$$

Moving on to the last term D, the substitution of (2.33) gives:

$$\begin{aligned}
D &= 2(1-F_1)f_\omega\rho\frac{\sigma_{\omega 2}}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial\omega}{\partial x_j}\\
&= 2(1-F_1)\rho\frac{\sigma_{\omega 2}}{\omega}\frac{\partial k}{\partial x_j}\frac{\partial\omega_u}{\partial x_j}\\
&= 2(1-F_1)\rho\sigma_{\omega 2}\frac{f_\omega}{\omega_u}\frac{1}{f_k}\frac{\partial k_u}{\partial x_j}\frac{\partial\omega_u}{\partial x_j}\\
D &= \rho(1-F_1)CD_{k\omega,u},
\end{aligned} \tag{3.28}$$

where

$$CD_{k\omega,u} = \frac{2\sigma_{\omega 2}}{\omega_u}\frac{f_\omega}{f_k}\frac{\partial k_u}{\partial x_j}\frac{\partial\omega_u}{\partial x_j}. \tag{3.29}$$

Putting together the four terms in (3.24), (3.25), (3.26) and (3.28) into (3.23), the final ω_u-equation for PANS is presented as:

$$\begin{aligned}
\rho\frac{\partial\omega_u}{\partial t} + \rho u_j\frac{\partial\omega_u}{\partial x_j} =& \frac{\gamma}{\nu_{tu}}P_{ku} - \left(\gamma\beta^*\rho - \frac{\gamma\beta^*\rho}{f_\omega} + \frac{\beta\rho}{f_\omega}\right)\omega_u^2\\
&+ \rho\frac{\partial}{\partial x_j}\left[(\nu+\sigma_{\omega u}\nu_{tu})\frac{\partial\omega_u}{\partial x_j}\right] + \rho(1-F_1)CD_{k\omega,u},
\end{aligned} \tag{3.30}$$

where equation for F_1 defined in 2.34 is unchanged but its input variable $\arg_1$ of (2.35) is now updated to $\arg_{1,u}$ which is defined as

$$\arg_{1,u} = \min\left[\max\left(\frac{\sqrt{k_u}}{\beta^*\omega_u d}, \frac{500\nu}{d^2\omega_u}\right), \frac{4\rho\sigma_{\omega 2}k_u}{CD_{k\omega,u}\,d^2}\right].$$

Similarly to the k_u-equation, the ω_u-equation is also identical in form when compared to the original $\omega-$equation of (2.30) just with different coefficient values. The k_u- and ω_u-equations of (3.21) and (3.30) ultimately forms the $k-\omega$ SST model for PANS.

4

Data-driven turbulence modelling

In recent years, the number of researches conducted for data-driven turbulence modelling has vastly increased. Many scientists have realised that the improvements in the accuracy of the results are more easily achieved as compared to the conventional method of expanding the insights of the flow physics involved. In this chapter, the background behind the shift in trend and examples of highlighted data-driven methods are presented.

4.1. Role of data in modern physics

Solving complex physical problems involves a large number of assumptions and these assumptions bring about inaccuracies in the solution. Thus, with the help of the abundance of data and the various tools that utilise them, better solutions are achieved instead of having to look for more assumptions to simply physical relations that cause the solution to deviate more from the true solution. Hence, the current trend that reflects this statement and the motivation behind the trend are presented in this section.

4.1.1. Current trend in CFD

Today's trend in science can be observed by looking at prominent research centres, universities and the fields of research that are of their focus. Such observation makes it clear that data-driven turbulence modelling has already gained its place and proved its potential to lead the future of turbulence modelling. The Delft University of Technology has recently opened 24 AI labs one of which is dedicated to fluid mechanics [47]. The University of Michigan has held a symposium on model-consistent data-driven turbulence modelling in the year 2021 [48] and together with NASA, they held a symposium on advances in turbulence modelling in the year 2017 in which data-driven turbulence modelling for RANS was newly placed under spotlight [49]. They are committing to this trend with an upcoming symposium that is dedicated to machine learning for turbulence modelling later in the summer of 2022 [50]. Furthermore, the von Karman Institute for Fluid Dynamics has been regularly hosting lecture series on the topics of data-driven turbulence modelling [51, 52, 53]

4.1.2. Motivation

With multiple large organisations indicated in Section 4.1.1 dwelling in data-driven turbulence modelling, their motivations are of interest. Despite the ever so often mentioned Moore's law that was claimed back in the year 1965 in [54] still being prevalent, it is largely insufficient for highly resolving methods to be applied to a realistically complex model such as a car at a sufficiently high Re. Spalart mentioned in the year 2000 in [55] when Moore's law still had a huge relevance that the application of DNS onto a car on a highway would only be feasible around the year 2080. However, it is inevitable that the consistent improvement in computational power has led to an explosive increase and abundance in HiFi datasets making data-driven turbulence modelling to be highly possible.

4.2. Machine learning

So the motivation behind turbulence modelling using data has been laid out in Section 4.1.2. Now, the role that this data abundance takes up in the fluid mechanics field, especially for CFD, mainly involves ML. ML is a common term that is heavily used in today's technical world. It involves a machine, most often a computer, to learn from a large pool of databases with the goal of attaining the ability to independently make decisions or relationships.

4.2.1. Types of ML

There are three main categories in ML and they are: supervised learning, unsupervised learning and semi-supervised learning [56].

In supervised learning, a pair of inputs and outputs known as a tagged or well-labelled dataset is given to the machine to have one or more functions that connect them discovered while in unsupervised learning, the machine takes a dataset that is not tagged and finds similarities and patterns between the data and cluster the dataset into sub-groups. Semi-supervised learning, also known as reinforcement learning, is about training the machine to make a sequence of decisions. It works on a reward and penalty system whereby the machine is either rewarded or penalised for a decision it makes. The ultimate goal is to maximise the rewards while minimising the penalties. Through this goal, the machine optimises a set of rules for such decisions and they are used in the subsequent environment that the machine is placed.

For the highly complex and high-dimensional nature of flow field problems that are dealt with in CFD, it would take the machine a huge amount of time and computational power for unsupervised and semi-supervised learning. Therefore, supervised learning has been the main category in that numerous ML algorithms for CFD have been developed. In the following sections, some of the more successful supervised learning algorithms that are tailored to CFD are introduced.

4.2.2. Artificial neural networks

Artificial NNs are inspired by the biological NNs that live inside our brains and it consists of several layers with multiple neurons in each layer. Each neuron takes in a certain number of inputs and through an activation function that accommodates these inputs, it outputs a result. Each input is paired with a weight coefficient that sets the amount of significance of the input for a specific activation function and its magnitude is set during the training phase of ML. For typical NNs, there are multiple layers of neurons, including the input and output layers. Between these layers, one or more hidden layers live and the activation functions involved in them are usually not explicit. An example of a deep learning NN with multiple hidden layers is shown in Figure 4.1 taken from [57].

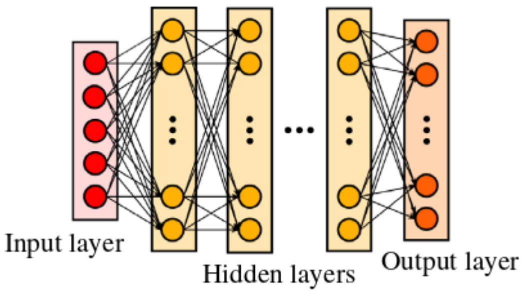


Figure 4.1: Deep neural network [57]

However, for CFD application and especially for prediction of anisotropy tensor that is the very bottleneck for highly modelled turbulence closures such as RANS, such a simple neural network is insufficient regardless of the number of hidden layers due to its disregard for Galilean invariance. Galilean invariance is a fundamental principle which implies that the physics of a flow field is independent of the orientation of the coordinate frame.

To integrate this Galilean invariance into the NN, a tensor basis NN was proposed in [57]. It accommodates this fundamental principle by ensuring that the anisotropy tensor is formulated with isotropic tensors as a basis. An overview of TBNN is shown in Figure 4.2 taken from [57] where an extra tensor input layer, $\boldsymbol{T}^{(n)}$, is present.

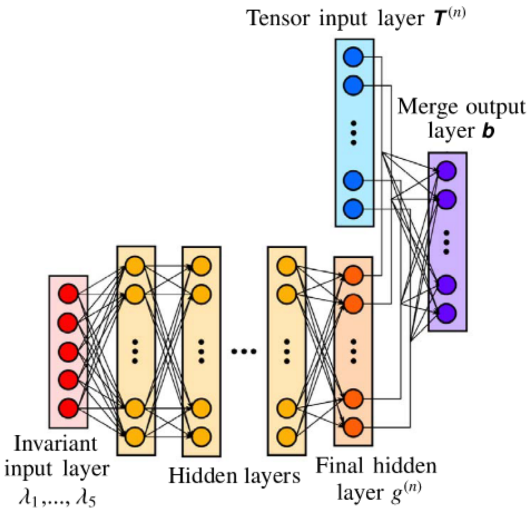


Figure 4.2: Tensor basis neural network [57]

This tensor input layer follows from Pope's finding in [58] which claims that for incompressible flow cases in which non-dimensionalised strain rate tensor S_{ij} and rotation rate tensor R_{ij} are the only variables that the selected eddy viscosity model depends on, the model can be defined in terms of ten isotropic basis tensors:

$$b_{ij}\left(S_{ij},\Omega_{ij}\right)=\sum_{n=1}^{10} g^{(n)}\left(\lambda_1,\cdots,\lambda_5\right) T_{ij}^{(n)} \tag{4.1}$$

where b_{ij} (presented as $\boldsymbol{b}$ in Figure 4.2) abides by the Galilean invariance, $g^{(n)}\left(\lambda_1,\cdots,\lambda_5\right)$ are the ten scalar coefficients that are to be discovered by TBNN in which $(\lambda_1,\cdots,\lambda_5)$ are five scalar tensor invariants which are all traces of some combinations of S_{ij} and Ω_{ij}. In addition, $T_{ij}^{(n)}$ (presented as $\boldsymbol{T}^{(n)}$ in Figure 4.2) are ten base tensors that are also some combinations of $\mathcal{S}_{ij}=S_{ij}/\omega$ and $\mathcal{R}_{ij}=\Omega_{ij}/\omega$ from [58]. For the 2D flow cases of data-driven RANS/PANS which is of focus in this paper, the first three base tensors form a linearly independent basis along with the first two scalar invariant coefficients which are the only non-zero coefficients according to [2] and they are given as

$$\begin{aligned}
&T_{ij}^{(1)}=\mathcal{S}_{ij}, \quad T_{ij}^{(2)}=\mathcal{S}_{ik}\mathcal{R}_{kj}-\mathcal{R}_{ik}\mathcal{S}_{kj},\\
&T_{ij}^{(3)}=\mathcal{S}_{ik}\mathcal{S}_{kj}-\frac{1}{3}\delta_{ij}\mathcal{S}_{mn}\mathcal{S}_{nm},\\
&\lambda_1=\mathcal{S}_{mn}\mathcal{S}_{nm} \quad \text{and} \quad \lambda_2=\mathcal{R}_{mn}\mathcal{R}_{nm}.
\end{aligned} \tag{4.2}$$

It was then concluded in [57] that TBNN had to be set to eight hidden layers with 30 neurons per hidden layer through Bayesian optimisation. Additionally, the artificial NNS are deterministic after they are trained which means that they do not rely on randomness but produce the same results for a fixed input for every run.

4.2.3. Gene expression programming

GEP is, as its name suggests, an ML algorithm that attempts to replicate the way human genetics function to discover an algebraic expression for a specific purpose and it overcomes the deficiencies that older algorithms such as the genetic algorithm introduced in [59] and genetic programming introduced in [60] possess. The deficiencies are related to either a lack of complexity or the difficulty of reproducing the complex structure with modification which involves chromosome functions such as mutation, transposition and gene recombination [61]. GEP on the other hand can reflect phenotype entirely using its feature where any genome modification always gives correct ETs.

A simple example of an ET is shown in Figure 4.3 taken from [62] which is achieved from a gene that is composed of {cos,+,−,∗,x,y,4,2} and it represents the following guessed function:

$$f^{\text{guess}}(x, y) = \cos(2 - y)(x + 4). \tag{4.3}$$

Each ET genetically represents a chromosome and it can be expressed as linear string of fixed length with rules that set GEP apart from the other gene influenced ML algorithms.

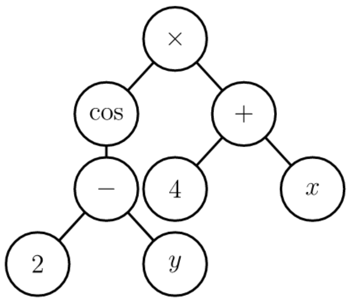


Figure 4.3: Simple ET representing an example chromosome of (4.3) [62]

A linear string consists of two parts: head and tail. With h as the fixed length of the head, the length of the tail t is

$$t = h(n_a - 1) + 1, \tag{4.4}$$

where n_a is the number of arguments the last math operator that the chromosome takes which in the case of this particular chromosome represented by (4.3) and Figure 4.3 is the addition operator (+) thus resulting in

$$h = 6, \;\; n_a = 2 \;\text{ and }\; t = 7,$$

giving a total gene length of $h + t = 13$ and it gives the following linear string: `*c-2y+|4xxyyx2` where `c` represents the cos (cosine) mathematical operator and `|` operator separates the head and tail components. Although there are only two parameters for the tail that should come after the `|` operator, {4, x}, the length of tail is seven which thus introduces a term called "open reading frames" which is the length of the code (the linear string) that is involved in the mathematical operator and the rest of the string is called the "non-coding region" [61]. Thus, for this particular case, the open reading frames is seven as the counting begins from zero and the x in Figure 4.3 is the seventh node.

Another important parameter in GEP is the fitness function, Fit(f^{guess}) that measures how well of a fit a chromosome is. There is no definite expression but is instead chosen based on users' preferences and criteria made for their various purposes. An outline of the procedure is presented in Figure 4.4 taken from [62].

First, the population P^i consisting of several candidate chromosomes that are produced from a given set of mathematical operators, parameters and constants is created and since this is the first generation, $i = 0$. The constants are usually created from Random Numerical Constants as users cannot specify the broad spectrum of possible parameters that GEP needs. This makes GEP a non-deterministic/stochastic process. Then the selection procedure is done based on the fitness function. The selected group of chromosomes are then reproduced based on the various chromosome functions. This is followed by genetic

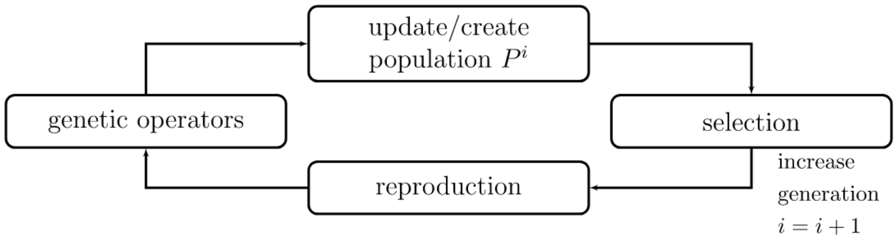


Figure 4.4: GEP algorithm procedure [62]

operators which modify members of each chromosome to determine how well it can represent other existing solutions. Poorly performing variants are discarded and the population is updated before the same set of procedures is applied to the new population.

4.3. Various developed data-driven closure methods

Apart from the difference in the ML algorithms that were used in different data-driven turbulence modelling literature like [57, 62], another difference is the parameters that were subjected to be learnt and reproduced by machine from given HiFi data. [57] attempted to optimise for the entire anisotropic Reynolds stress defined as b_{ij} in (2.24) using (4.1) while [62] attempted to optimise for an additional term that is missing from the already defined base model b_{ij}. Thus, in the following sections, the different data-driven closure terms that close the RS_{ij} term given by Boussinesq eddy viscosity approximation in [10] for RANS turbulence models are introduced.

4.3.1. k-corrective frozen RANS

In [2], a deterministic symbolic regression method called SpaRTA was introduced in optimising for the anisotropy term b_{ij} of Reynolds stress and the turbulence energy production term. Symbolic regression is one of the ML algorithms that are in the same family of algorithms as genetics-inspired algorithms such as genetic programming and GEP. The performance of the symbolic regression used in SpaRTA was largely improved by the use of the Fast Function Extraction technique from [63] in terms of processing speed and the quality of final results.

The data-driven closure model that was used in the literature is the $k-$corrective frozen RANS approach that was built from the work of [64] and it is based on the $k - \omega$ SST RANS model shown in (2.29) and (2.30). It aims to optimise for the model-form error using HiFi dataset in the $k-$equation and the Boussinesq eddy viscosity approximation which has shown its limitations in representing turbulence [12]. It redefines the anisotropy term of Reynolds stress shown in (2.24) as the baseline model b_{ij}^0 and introduces a correction term b_{ij}^Δ that completes the theoretically true anisotropy term resulting in

$$\begin{aligned} b_{ij}^* &= b_{ij}^0 + b_{ij}^\Delta \\ &= -\frac{\nu_t}{k} S_{ij} + b_{ij}^\Delta, \end{aligned} \tag{4.5}$$

where the superscript$-(*)$ notation is used to distinguish from the conventional definition of b_{ij}.

To obtain b_{ij}^Δ, the kinematic eddy viscosity term ν_t of (4.5) is required. This in turn requires ω of (2.28) to be evaluated which is done by solving the conventional $\omega-$equation of the SST model as shown in (2.30). However, in this data-driven approach, another correction term, apart from b_{ij}^Δ, is introduced to correct for the model error in the turbulence energy production term P labelled as the R term which updates the $k - \omega$ SST equations to

$$\rho \frac{\partial (k)}{\partial t} + \rho \overline{u}_j \frac{\partial \left(k \right)}{\partial x_j} = (P_k^* + R) - \beta^* \rho \omega k + T_k \quad \text{and} \tag{4.6}$$

$$\rho \frac{\partial (\omega)}{\partial t} + \rho \overline{u}_j \frac{\partial \left(\omega \right)}{\partial x_j} = \frac{\gamma}{\nu_t} (P_k^* + R) - \beta \rho \omega^2 + T_\omega + \rho (1 - F_1) CD_{k\omega}. \tag{4.7}$$

where computation of P^* is done using limiter defined by Menter in [26] just like in (2.39) and with the addition of newly defined anisotropy term of kinematic eddy viscosity into the Reynolds stress tensors which is

$$P_k^* = \min \left(\rho RS_{ij}^* \frac{\partial \overline{u}_i}{\partial x_j}, 10 \beta^* \rho \omega k \right) \quad \text{and} \tag{4.8}$$

$$RS_{ij}^* = 2k \left(b_{ij}^0 + b_{ij}^\Delta + \frac{1}{3} \delta_{ij} \right) \tag{4.9}$$

The overall procedure for the retrieval of the correction terms is as follows:

1. The HiFi data such as $\overline{u}_j$, k and RS_{ij} are inserted into the $\omega-$equation of 4.7 in which R is first assumed to be 0 and all parameters except ω are known. ω is solved for.
2. The obtained ω is inserted into the $k-$equation solving for $k-$equation model error term, R.
3. Concurrently, ω is used to update ν_t of (2.28).
4. Using HiFi data and the updated ν_t: RS_{ij} and k, b_{ij}^Δ are obtained using (2.25) and (4.5).
5. Using the updated R, $(P_k^* + R)$ term is updated and $\omega-$equation is solved again.
6. This iterative procedure is continued till the values of R and b_{ij}^Δ converge to the user's desired tolerance margin.

These are then used for the aforementioned symbolic regression to obtain algebraic expressions.

4.3.2. Multidimensional GEP driven anisotropy optimisation

Due to the high dimensional nature of CFD, the standard GEP introduced in Section 4.2.3 gives invalid results for data-driven turbulence modelling. In [62], a multi-dimensional GEP was set up to accommodate the tensors of turbulence closures. Using this algorithm, the full anisotropy of the Reynolds stress b_{ij}, shown in (2.24), was selected to be the target variable. An extra term defined in [62] as b_{ij}^x that is supposedly set to zero in a classical $k - \omega$ SST model was subjected to optimisation. Just like TBNN, multi-dimensional GEP also utilised the set of tensors defined in [58] as shown in (4.2) to discover an algebraic expression for the extra term.

4.3.3. Data-driven Stochastic Closure Simulation

In [3], a data-driven PANS approach named DSCS was presented. Using a more elaborate definition of the aforementioned periodic unsteadiness of turbulent flows : "large-scale turbulent fluid mass with a phase-correlated vorticity over its spatial extent" given by [65], [3] furthered this definition by characterising it as an organised component within unsteadiness that is by nature, deterministic which means that the presence of this organised unsteadiness is inevitable regardless of the unsteady flow. To resolve these scales that RANS does not, [3] attempts to use data-trained PANS. It follows the data-driven procedure of [62] whereby GEP, mentioned in Section 4.2.3, is utilised.

Using the triple decomposition, the Reynolds stress tensor, following from (2.12), is partitioned into two parts:

$$\begin{aligned} RS_{ij} &= \overline{u_i' u_j'} \\ &= \overline{(\widetilde{u}_i + u_i'')(\widetilde{u}_j + u_j'')} \\ &= \overline{\widetilde{u}_i \widetilde{u}_j} + \overline{u_i'' u_j''} \\ &= \widetilde{RS}_{ij} + RS_{ij}'' \end{aligned} \tag{4.10}$$

in which the first term $\widetilde{RS}_{ij}$ is given the name "periodic Reynolds stress" and the second term RS''_{ij}, "turbulent Reynolds stress". Using the idea that the periodic Reynolds stress takes up the periodic wave motion, the segregation of length scales associated to the periodic motion and the stochastic motion is deemed feasible. Thus, it aims to take on an alternative approach to the conventional Boussinesq closure of (2.22) wherein all turbulence length scales are modelled. Having defined (2.13) with explicit velocity components of the periodic unsteadiness, $\widehat{u}$ in URANS equations of (2.15) can now be replaced with $(\overline{u} + \widetilde{u})$, explicitly representing the N-S equations for unsteady flow that excludes the stochastic unsteadiness u''_i.

Contrary to the implementation of GEP in [62] that worked on producing an extra term for the anisotropy of Reynolds stress ultimately correcting for steady RANS, this method attempts to use GEP in discovering a new closure for the URANS which resolves the low-frequency periodic unsteadiness that is made up of large turbulent scales. Hence, it introduces URANS' Reynolds stress term following from (2.25) with an extra anisotropy term b^{xt}_{ij} for the new closure given as

$$RS^{URANS}_{ij} = 2k(b_{ij} + b^{xt}_{ij} + \frac{1}{3}\delta_{ij}) \tag{4.11}$$

representing RS''_{ij} and it purely accounts for the stochastic unsteadiness. Now, since this new turbulent closure of (4.11) takes the role of accounting for the stochastic unsteadiness while leaving the resolving of the periodic unsteadiness to the URANS equations, it introduces PANS to tweak $k - \omega$ SST equation. As it was covered in Section 3.2, the PANS form of $k - \omega$ SST solves the unresolved/modelled portion of the turbulent flow and is thus highly relevant for this purpose to be used alongside (4.11) to prevent accounting for the periodic unsteadiness twice. Hence, the PANS form of the Reynolds stress is introduced:

$$RS^{PANS}_{ij} = 2k_u(b_{ij,u} + b^{xt}_{ij,u} + \frac{1}{3}\delta_{ij}), \tag{4.12}$$

representing RS''_{ij} where

$$b_{ij,u} = -\frac{\nu_{tu}}{k_u} S_{ij} \tag{4.13}$$

and ν_{tu} is given by 3.18.

In $k-$corrective frozen method of Section 4.3.1, k field is directly fed into the set of equations from HiFi data. However, in DSCS, the k_u is instead obtained by solving the k_u-equation of PANS. The following procedure is taken up by DSCS:

1. Using the full Reynolds Stress tensor RS_{ij} and the turbulence kinetic energy k from the HiFi dataset, the full anisotropy tensor b_{ij} is obtained.
2. Via FFT or POD, both tensors: RS_{ij} and b_{ij} are split into periodic unsteadiness and stochastic unsteadiness parts. The same is done for the turbulence kinetic energy k to compute the field values for f_k.
3. ω_u, k_u, ν_{tu} are solved for using the PANS $k_u - \omega_u$ SST equations. $b_{ij,u}$ is then calculated using (4.13).
4. Using the stochastic unsteadiness part: b''_{ij} from HiFi dataset as the reference data and $b_{ij,u}$, GEP is applied to obtain $b^{xt}_{ij,u}$ using the base tensors of (4.2).

4.3.4. Potential shortcomings and improvements

With the two closure methods under the scope: $k-$corrective frozen RANS from SpaRTA and DSCS, some of their shortcomings are realised and they are elaborated on in this section.

For the $k-$corrective frozen RANS method that has been presented in Section 4.3.1, one arguable aspect is the addition of the $k-$equation's model error term onto the turbulence kinetic production term P^*_k as shown in (4.6) and (4.7). Since P^*_k is sourced from HiFi dataset, it should not be carried on to the $\omega-$equation but merely stay as a correction term that accounts for $k-$equation's model error. Thus, the suggested improvement would be the exclusion of R term from (4.7), giving:

$$\rho\frac{\partial(\omega)}{\partial t}+\rho\overline{u}_j\frac{\partial\left(\omega\right)}{\partial x_j}=\frac{\gamma}{\nu_t}P_k^*-\beta\rho\omega^2+T_\omega+\rho(1-F_1)CD_{k\omega}. \tag{4.14}$$

Another ambiguous aspect with the same reasoning is the use of Menter's limiter for correcting P_k using P_k^* given by (4.8). This limiter introduces ω into the formulation which is purely a modelled concept of turbulence alongside a constant, β^*, that is entirely empirically decided on. These modelled parameters override the HiFi dataset in some settings and make it less of a "HiFi" dataset. As for the last point, an unsteady case would be an attractive case to further this data-driven method.

Next, for the DSCS method presented in the previous section, the HiFi dataset is not fully utilised. Given k from the HiFi dataset that is used to calculate f_k via FFT or POD, it could have been used to be injected into the $k_u - \omega_u$ PANS equations, together with HiFi velocity field, like it was done in the $k-$corrective frozen RANS method. However, it was only used for obtaining f_k and the full Reynolds stress tensor RS_{ij} from which the anisotropy tensor b_{ij} was calculated. The the periodic and stochastic unsteady components of b_{ij} were extracted using the previously obtained f_k value.

5
High-fidelity data

Data-driven methods are limited by the availability of dataset, specifically those of high-fidelity as it is a uncommon for such a dataset to be at hand for a study to be done on a particular fluid domain with desired flow conditions to be studied. In this chapter, the HiFi data used for the project is explained.

5.1. Dataset selection

The HiFi dataset used for the project is taken from [66] in which LES data with commendable amount of fidelity was produced. The study involved the investigation of combustion dynamics of a V-flame whereby the V-flame holder is shaped just like a triangular prism with a cross-section of an equilateral triangle with one of its vertices facing the inlet as shown in Figure 5.1 taken from [66].

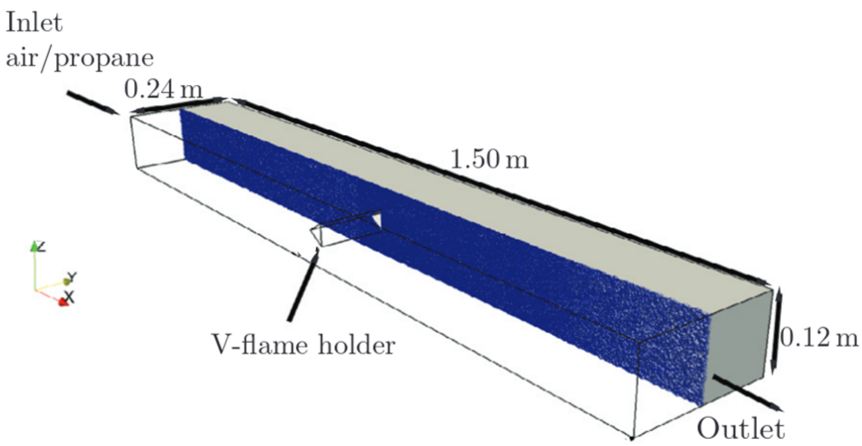


Figure 5.1: Computational V-flame configuration [66]

This V-flame holder is positioned in the middle of the channel, to give enough space and time for inflow to be developed into a fully turbulent flow. In addition, the vertical centre (along $z-$axis) of the equilateral triangle is positioned in the middle of the channel height for a vertically unbiased flow simulation.

The triangular prism is an optimal structure for fluid flow to be studied on, especially with highly modelled method like RANS where it is said by [3] that the most notable shortcomings of the RANS turbulence models come from their poor predictions in separating flows which is most apparent behind a bluff body. Although many bluff bodies such as circular and square cylinders have been researched on, the number of studies done for triangular prisms is significantly lesser, reflecting its little popularity.

However, it is an extremely interesting structure to be studied on due to the investigative advantages it possesses compared to its counterparts.

For example, a CFD simulation around a square cylinder has high instability due to the extreme flow deceleration at the leading edge of the cylinder and the flow separation that occurs at the corners of the leading edge. Additionally at sufficiently high Re, due to this early separation, the flow never reattaches along the stream-wise lengths thus the influence of the trailing edges on the flow instability at the wake is unable to be independently studied.

A circular cylinder on the other hand does not feature sharp corners where sharp flow separation usually occurs at. Therefore, a wake behind such a sharp separation is not studied. Hence, the structural characteristics that triangular prism has allows for a study of an unstable wake that is purely caused by sharp corners of the trailing edges of the structure since its inlet-facing vertex allows for a gradual deceleration of the flow while the flow is still attached.

5.2. Dataset flow field

In this section, the domain of the flow field is introduced alongside the mesh that was used for the LES study in [66]. Additionally, the various settings and parameters that were defined for the selected HiFi dataset is presented.

5.2.1. Flow field domain and mesh

Firstly, the dimensions of the fluid domain can be observed from Figure 5.1. The channel has a length of $1.50\,\text{m}$ along $x-$axis, a height of $0.12\,\text{m}$ along $z-$axis and a width of $0.24\,\text{m}$ along $y-$axis. The equilateral triangle has a side length of $0.82\,\text{m}$ while the triangular prism's width spans the width of the channel. Apart from the inlet and the outlet, the rest of the boundaries in $z-$axis and $y-$axis are defined as walls where no-slip condition is implied. The no-slip condition is applied also for the walls of the triangular prism.

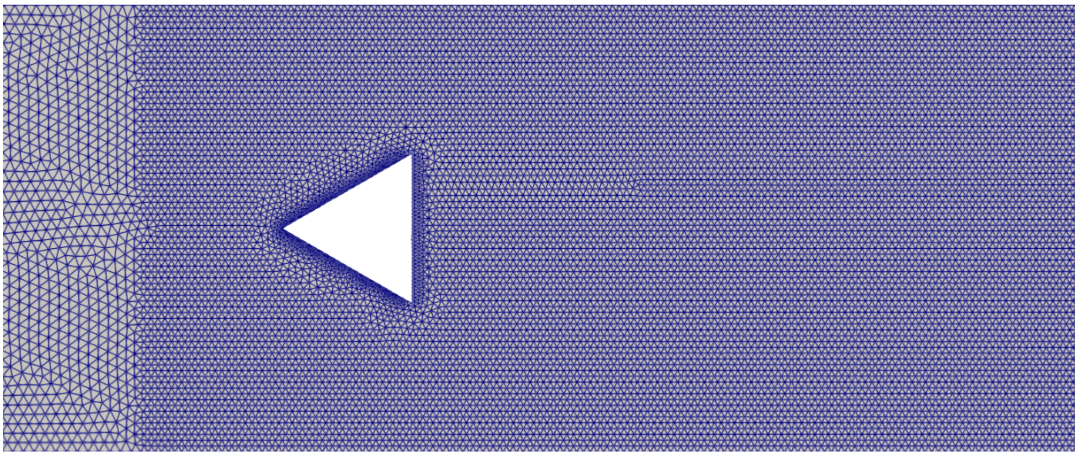


Figure 5.2: Mesh of HiFi dataset

In this domain, an unstructured mesh is applied around the triangular prism as shown in Figure 5.2. As it can be seen form the figure, the mesh is much coarser far upstream of the triangular prism whereas mesh retains a good amount of fineness far downstream since the vortices from the expected flow separations need to be solved and studied. In the region near the walls of the triangular prism, a greater refinement is made to the mesh as shown in Figure 5.3. The mesh for the domain is found to have around 8.1 million points that make up 46.5 million tetrahedral cells.

Such a high level of mesh fineness near the wall is used in order to accurately resolve the eddies in-

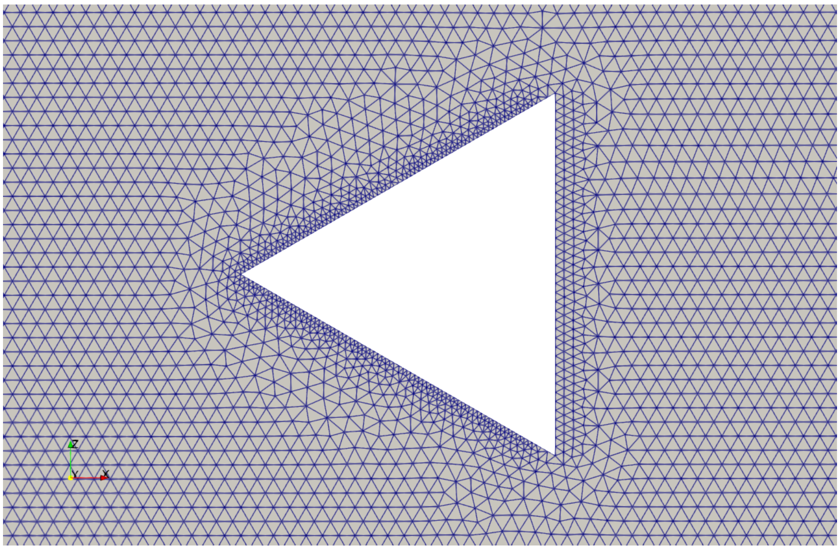


Figure 5.3: Mesh of HiFi dataset near triangular prism

stead of applying wall modelling that induces approximations and thus undesired inaccuracies into the intended HiFi dataset solution. As even the largest of the eddies in this near-wall region is extremely small, y^+ value of less than 1 must be retained which is done in this study hence making it an acceptable HiFi dataset to be used for the data-driven study.

Using this fine mesh, a velocity field of high resolution, much higher than that of a typical RANS solution, was able to be achieved by resolving the smaller turbulence scales as it can be observed in Figure 5.4.

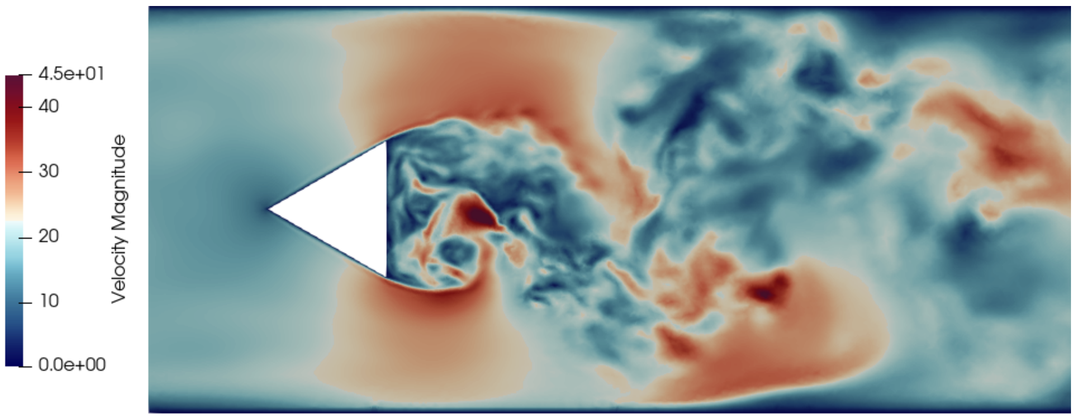


Figure 5.4: Velocity field of HiFi dataset

Consequently, a large spectrum of various turbulence scale sizes are observed in great details in the contours of Q-criterion shown in Figure 5.5 in which extremely small eddies can be observed on the walls of the triangular prism where the extremely high mesh resolution is imposed on.

5.2.2. Flow parameters

Although a combustion dynamics study was conducted in [66], a non-reacting case was evaluated and validated. The calculated flow viscosity is close to that of the atmosphere thus it is deemed as a reasonable dataset for free flow study to be based on. This non-reacting flow is made up of 21.85% of Oxygen (O_2) and 78.85% of Nitrogen (N_2). Using the viscosity relations given by [67], the kinematic

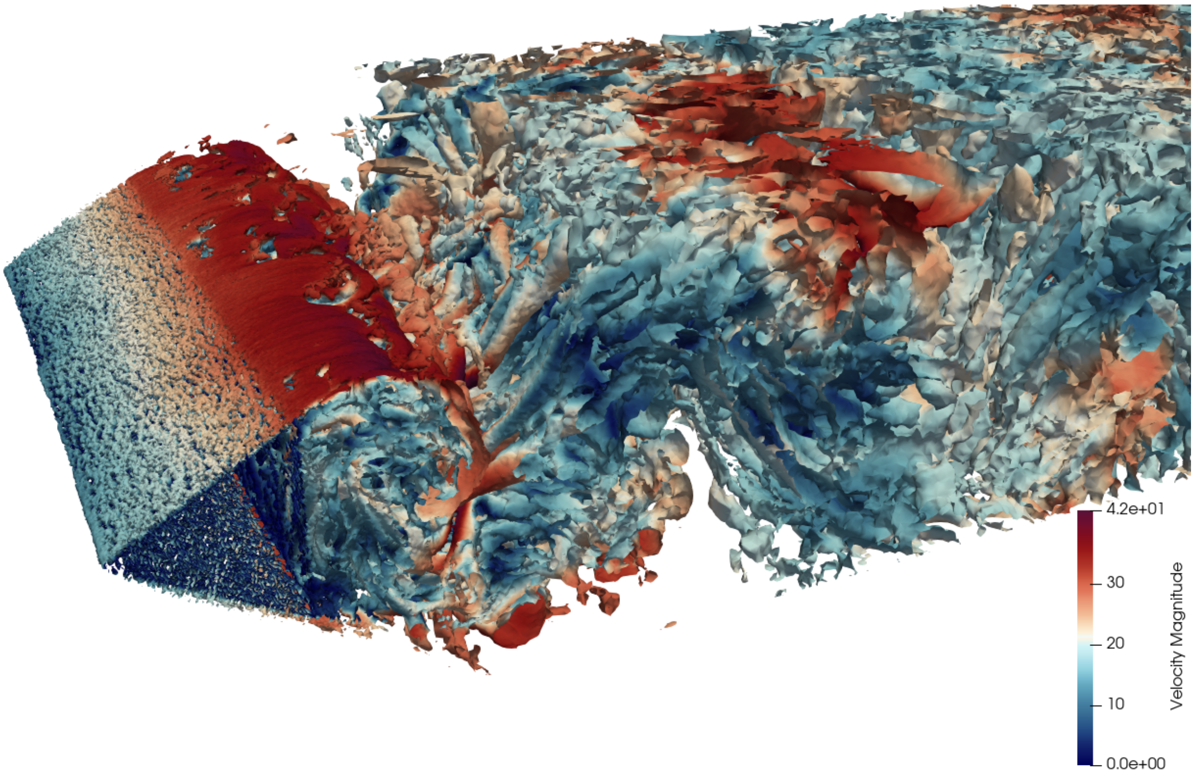


Figure 5.5: Contours of Q-criterion

viscosity ν of the flow is found to be

$$\nu = 1.4822 \times 10^{-5}\,\mathrm{m^2/s}. \tag{5.1}$$

The study was done for $288\,\mathrm{K}$ which is equivalent to $14.85\,°\mathrm{C}$. According to [68], atmospheric air at $15\,°\mathrm{C}$ has a kinematic viscosity of $1.48 \times 10^{-5}\,\mathrm{m^2/s}$ and therefore the two flows can be considered almost identical at least in the study of flow dynamics. With the freestream flow velocity value of $u_\infty = 16.6\,\mathrm{m\,s^{-1}}$ used in literature and kinematic viscosity of (5.1), the following Reynolds number of the flow was calculated:

$$\begin{aligned} Re &= \frac{u\mathcal{L}}{\nu} \\ &= \frac{16.6 \cdot 0.034641}{1.4822 \times 10^{-5}} \\ &= 38796, \end{aligned} \tag{5.2}$$

where $\mathcal{L} = 0.034\,641\,\mathrm{m}$ is the characteristic length which for this fluid domain, is the horizontal length of the equilateral triangle in the direction of the $x-$axis.

5.3. Post-processing

To conduct $k-$corrective study, required flow parameters of the HiFi dataset need to be calculated using the field values of pressure and the three velocity components that are available.

5.3.1. HiFi parameters

As described in Section 4.3.1, the HiFi parameters that are assimilated into the RANS turbulence model are the three velocity components u_i, turbulence kinetic energy k and Reynolds stress tensor RS_{ij}. The difference for this data-driven study is that the flow case is unsteady. Hence for k, unlike (2.18) where infinite averaging is applied as done in Reynolds averaging, a finite averaging is applied. Thus, similarly to (2.14), k is obtained using the following equation

$$k = \frac{1}{2}\widehat{u_i' u_i'}, \tag{5.3}$$

while the the six components of Reynolds stress tensor RS_{ij} for the unsteady flow is given by

$$RS_{ij} = \widehat{u_i' u_j'}. \tag{5.4}$$

The finite averaging is done via triple decomposition as introduced in Section 2.2.3 and will be further explained in detail in Section 7.1. It is worth mentioning that although RS_{ij} is a tensor with nine elements in a 3×3 square matrix $-$ cycling through $i = 1, 2, 3$ and $j = 1, 2, 3$ $-$ it is a symmetrical matrix. Thus for example, RS_{12} is equivalent to RS_{21}, ultimately resulting in just six unique components.

5.3.2. Vortex shedding frequency

In order to use this HiFi dataset in improving the prediction of a flow field via injecting the dataset into PANS, it s crucial for the HiFi dataset to correctly represent the the intended flow field with flow parameters stated in Section 5.2.2. This way the correction terms obtained would truly give PANS an improvement in solving the given flow, driving it to a true solution. Thus, the lift force in $z-$axis on the triangular prism has been calculated by applying the following relation on the walls of the triangular prism:

$$L_z = \sum_{n=0}^{N} p_n A_n \hat{k} \tag{5.5}$$

where p_n and A_n are the pressure and surface area of a single cell, and $\hat{k}$ is a unit vector in the direction of $z-$axis.

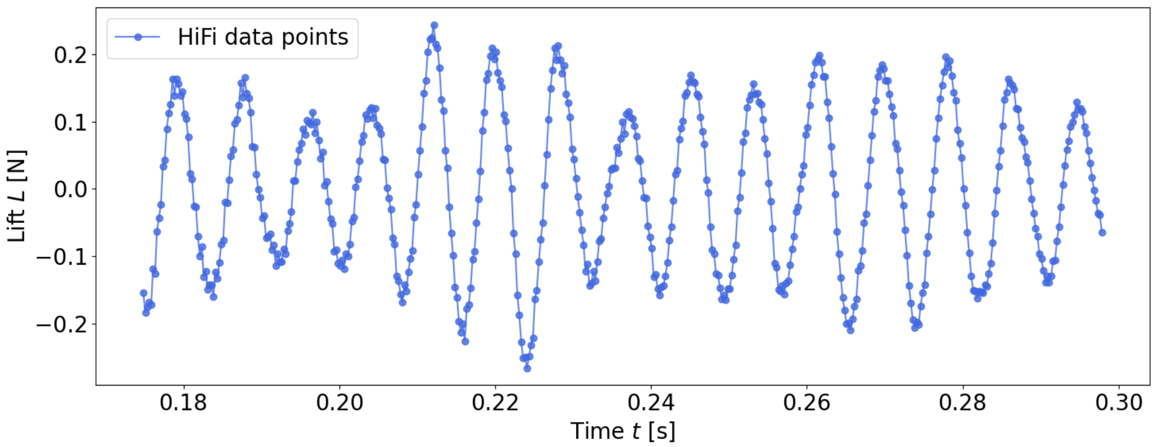


Figure 5.6: Lift of HiFi dataset against time

The calculated lift force is plotted against time as shown in Figure 5.6. It is to be noted that the first point starts at $t \approx 0.17$ as the flow has not yet stabilised prior to this point. It can be confirmed from this plot that the flow field clearly exhibits periodic vortex shedding similar to the von Kármán vortex street shown in Figure 2.3.

Furthermore using this force analysis, the vortex shedding frequency needs to be extracted for a comparison with the experimental value of $f_{vs,exp} = 105\,\mathrm{Hz}$ that is provided by [66]. Using FFT in Scipy[1], a scientific Python library, Figure 5.7 is produced.

[1]https://scipy.org/

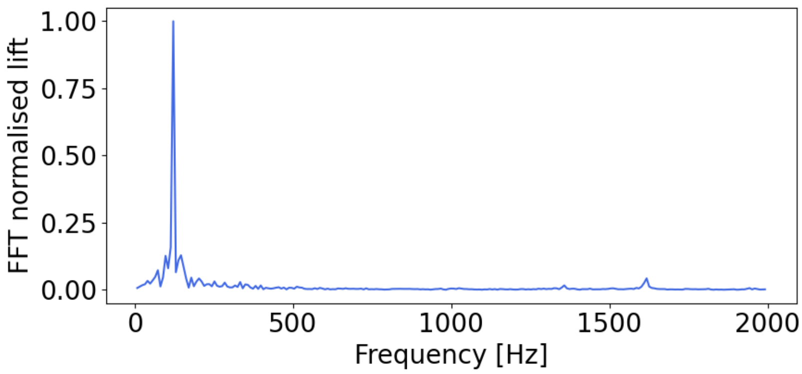


Figure 5.7: Frequency plot of Lift of HiFi dataset

A clear dominant frequency can be observed and the corresponding frequency is found to be 122 Hz which is contains an error bound of ± 5 Hz due to discrete time signal according to [66]. The difference in vortex shedding frequencies of the experimental data and the HiFi LES data is concluded to be an acceptable value. The highlighted parameters of the HiFi dataset are assembled in the Table 5.1.

Parameter	Value
Number of points on mesh	8.1×10^6
Number of tetrahedral cells on mesh	46.5×10^6
Freestream velocity u_∞	$16.6\,\mathrm{m\,s^{-1}}$
Kinematic viscosity ν	$1.48 \times 10^{-5}\,\mathrm{m^2/s}$
Reynolds number Re	38 796
LES vortex shedding frequency $f_{vs,LES}$	122 Hz
Experimental vortex shedding frequency $f_{vs,exp}$	105 Hz

Table 5.1: HiFi dataset parameters

In all, the HiFi dataset is deemed to be a good enough representation of the flow field for a data-driven study to be done. However, this imperfection of the HiFi dataset should always be kept in mind.

6

PANS implementation

In chapter 3, PANS and some of its defining characteristics were introduced, and its $k_u - \omega_u$ SST model was derived starting from the RANS form. In this chapter, after a brief mention of its exact implementation, the mesh created for the computation of PANS is introduced, followed by validation of the newly implemented model and some of the highlighted results.

6.1. Governing equations

The PANS form of the $k - \omega$ SST turbulence model has been derived in Section 3.2.2 resulting in (3.21) and (3.30), the transport differential equations for unresolved k and ω respectively. However, just like how the RANS form of $k - \omega$ SST model that is implemented into various CFD programs such as OpenFOAM[1] and Ansys CFX[2] features slightly different form as demonstrated in [26] compared to the conventional form as in [24], PANS form of $k_u - \omega_u$ SST is also slightly adjusted to better suit computational calculations. The motivation for this difference is essentially the improvement in convergence.

Similarly to P^+ of (2.39), μ_t^+ of (2.36) and $CD_{k\omega}^+$ of (2.40), the same set of terms in PANS $k_u - \omega_u$ SST are adjusted for the CFD softwares and they are:

$$P_{ku}^+ = \min\left(P_{ku}, 10\beta^* k_u \omega_u\right), \tag{6.1}$$

$$\mu_{tu}^+ = \rho\nu_{tu}^+ = \frac{a_1 k_u}{\max\left(a_1\omega_u, W F_2\right)} \quad \text{and} \tag{6.2}$$

$$CD_{k\omega,u}^+ = \max\left(\frac{2\sigma_{\omega 2}}{\omega_u}\frac{f_\omega}{f_k}\frac{\partial k_u}{\partial x_j}\frac{\partial \omega_u}{\partial x_j}, 10^{-20}\right). \tag{6.3}$$

where F_2 is defined in (2.37) and just like F_1, its expression is unchanged but $\arg_2$ of (2.38), the argument that it takes, is updated to $\arg_{2,u}$ and it is

$$\arg_{2,u} = \max\left(2\frac{\sqrt{k_u}}{\beta^*\omega_u d}, \frac{500\nu}{d^2\omega_u}\right).$$

The k_u and ω_u equations are then updated to

$$\rho\frac{\partial k_u}{\partial t} + \rho u_j\frac{\partial k_u}{\partial x_j} = P_{ku}^+ - \beta^*\rho\omega_u k_u + \rho\frac{\partial}{\partial x_j}\left[\left(\nu + \sigma_{ku}\nu_{tu}^+\right)\frac{\partial k_u}{\partial x_j}\right] \quad \text{and} \tag{6.4}$$

[1]https://www.openfoam.com/

[2]https://www.ansys.com/products/fluids/ansys-cfx

$$
\begin{aligned}
\rho\frac{\partial \omega_u}{\partial t}+\rho u_j\frac{\partial \omega_u}{\partial x_j} =& \frac{\gamma}{\nu_{tu}^+}P_{ku}^+ - \gamma\beta^*\rho\omega_u^2+\gamma\beta^*\rho\frac{\omega_u^2}{f_\omega}-\beta\rho\frac{\omega_u^2}{f_\omega} \\
&+\rho\frac{\partial}{\partial x_j}\left[\left(\nu+\sigma_{\omega u}\nu_{tu}^+\right)\frac{\partial \omega_u}{\partial x_j}\right]+\rho(1-F_1)CD_{k\omega,u}^+.,
\end{aligned}
\tag{6.5}
$$

6.2. Meshing

Like any other turbulence model, PANS require a mesh for the fluid domain for the flow to be solved. However, since the purpose of PANS in this project is not to produce the HiFI dataset, a structured mesh that minimises computational effort while still accommodating the various needs such as being wall-resolving is produced. Furthermore, instead of the 3D mesh that is used for the HiFi dataset, a 2D mesh is produced as it was concluded in [3] that "Mean velocity profiles for the two and three-dimensional calculations show an insignificant difference" for PANS computation. First, the size of the domain chosen is divided into some blocks as shown in Figure 6.1.

Figure 6.1: PANS fluid domain blocks

Compared to the mesh for the HiFi dataset shown in Figure 5.2, the length between the inlet and the triangular prism is cut short. This is because in RANS and PANS calculations, in this case, the turbulent flow is assumed to be fully developed from the inlet unlike in LES or DNS. The choice for the various sizes and shapes of the blocks was to accommodate for maximum mesh quality where skewness of the quadrangle structures is kept to a minimum. Additionally, mesh refinement in such a sub-divided domain is much more easily controlled allowing for the sponge layer far downstream to avoid flow reflection. The mesh is shown in Figure 6.2 with a closeup of the near-wall cells in Figure 6.3.

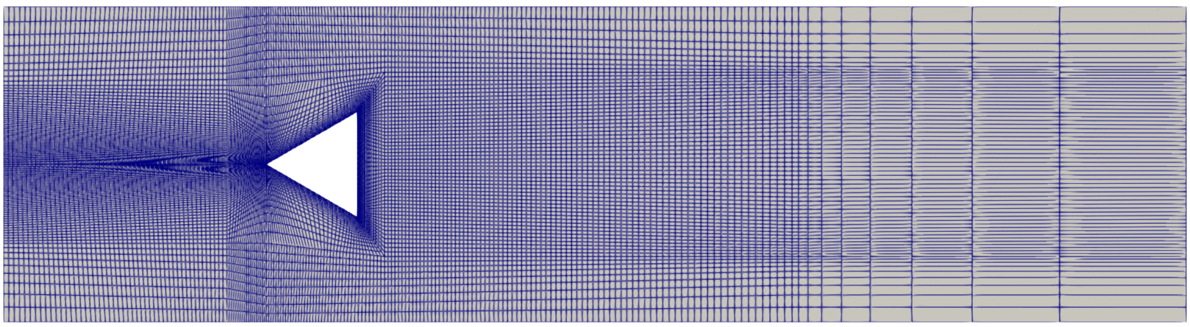


Figure 6.2: PANS mesh

The mesh is generated with attention to three main characteristics and these can be observed from Figure 6.2 and Figure 6.3:

1. $y^+ < 1$ needs to be achieved around the walls of the triangular prism.

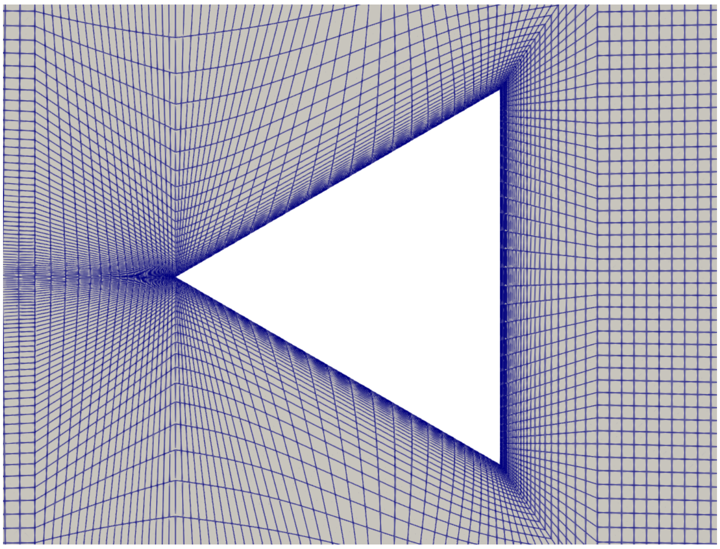


Figure 6.3: PANS mesh near triangular prism

2. Wall distance at other walls is kept coarse to minimise computational effort. It is also done to not introduce additional eddies into the flow as they are not to be studied.
3. A sponge layer is added far downstream to prevent reflection.

The number of points and cells of the mesh are gathered in Table 6.1.

Parameter name	Value
Number of points on mesh	39 480
Number of hexahedral cells on mesh	19 300

Table 6.1: PANS mesh statistics

6.3. OpenFoam implementation

The lines of code used to create the PANS $k_u - \omega_u$ SST is presented in Appendix A wherein both the C and H files are presented. It should be noted that the file is incomplete and only the most significant lines are presented so it does not work independently. The program is developed based on the default RANS $k - \omega$ SST implementation by OpenFOAM v2112 and its structure is followed for consistency's sake. In this section, a pseudo-code that outlines the structure of the program alongside a few snippets of the code that are highlighted are presented.

The case under study is an unsteady case with an incompressible flow assumption and hence for the solver in OpenFOAM, the incompressible PIMPLE algorithm is chosen to be used which combines PISO (Pressure Implicit with Splitting of Operator) and SIMPLE (Semi-Implicit Method for Pressure-Linked Equations) algorithms. The algorithm is used to couple velocity field and pressure field at every time step. The velocity field is then used in the turbulence model, $k_u - \omega_u$ SST in this case, to solve for k_u and ω_u fields. First, the boundary conditions and initial conditions used are introduced in Table 6.2 which is supported by (6.6) and 6.8 given by [69]:

$$k = \frac{3}{2}(0.16u\mathcal{I})^2 \tag{6.6}$$
$$\mathcal{I} = 0.16Re^{-1/8} \tag{6.7}$$
$$\omega = (\beta^*)^{-1/4}\frac{\sqrt{k}}{\ell_t} \tag{6.8}$$
$$\ell_t = 0.07d_h \tag{6.9}$$

where d_h is taken as the height of the inlet for the given domain which has a value of $0.12\,\mathrm{m}$ shown in Figure 5.1.

Field	Inlet		Outlet		Wall		Internal field
	BC	IC	BC	IC	BC	IC	IC
u	fixed	16.6	$\Delta u = 0$	N.A.	no slip	N.A.	16.6
p	$\Delta p = 0$	N.A.	fixed value	1 atm	$\Delta p = 0$	N.A.	1 atm
k	fixed	0.75357 via (6.6)	$\Delta k = 0$	N.A.	N.A.	10^{-8}	0.75357 via (6.6)
ω	fixed	188.68 via (6.8)	$\Delta\omega = 0$	N.A.	N.A.	10^{8}	188.6 via (6.8)
ν_t	fixed	0.0040 via (2.28)	$\Delta\nu_t = 0$	N.A.	N.A.	10^{-10}	0.0040 via (2.28)

Table 6.2: Boundary and initial conditions

Algorithm 1 walks through the summary of the newly implemented turbulence model. The initial field values at $t = 0$ are marked with subscript 0 and intermittent time steps are marked with subscript t.

Algorithm 1 PANS $k_u - \omega_u$ SST

Require: f_k ▷ f_ω is simply an inverse of f_k
  Read u_0, p_0, k_0, ω_0, $\nu_{t,0}$ ▷ Read from $t = 0$ folder
  $k_{u,0} \leftarrow k_0 \times f_k$
  $\omega_{u,0} \leftarrow \omega_0/f_k$
  $\nu_{tu,0} \leftarrow k_{u,0}/\omega_{u,0}$
  while $t \leq t_{final}$ **do**
    procedure PIMPLE SOLVER(u_{t-1}, p_{t-1})
      $u_t \leftarrow$ coupling of u_t and p_t
      $i, j \leftarrow 0, 0$ ▷ i, j: Iteration counters
      while $\omega_{u,i} - \omega_{u,i-1} >$ tolerance$_\omega$ **do**
        $\omega_{u,i} \leftarrow$ Solve via (6.5)
      end while
      while $k_{u,j} - k_{u,j-1} >$ tolerance$_k$ **do**
        $k_{u,j} \leftarrow$ Solve via (6.4)
      end while
      $\omega_{u,t}, k_{u,t} \leftarrow \omega_{u,i}, k_{u,j}$
      $\nu_{tu,t} \leftarrow k_{u,t}/\omega_{u,t}$ ▷ Update unresolved kinematic eddy viscosity
      $\omega_t \leftarrow \omega_{u,t} \times f_k$
      $k_t \leftarrow k_{u,t}/f_k$
    end procedure
    $t \leftarrow t + dt$
  end while

6.4. Validation

Before proceeding to utilise the PANS program created, it had to be validated. As claimed in the literature review done for PANS in chapter 3, PANS should operate as RANS when $f_k = 1.0$. At this value of f_k, all of the flow is modelled by the chosen turbulence closure model which in this project is the $k - \omega$ SST. Thus, some of the flow characteristics, as well as flow parameters are compared between computation results from PANS at f_k and from the conventional RANS SST model. First of all, the dominant frequencies were calculated as it was done in Section 5.3.2 and the result is shown in figure6.4. Peaks at the same frequencies can be observed with the dominant frequency being $f_{vs} = 113.82\,\text{Hz}$.

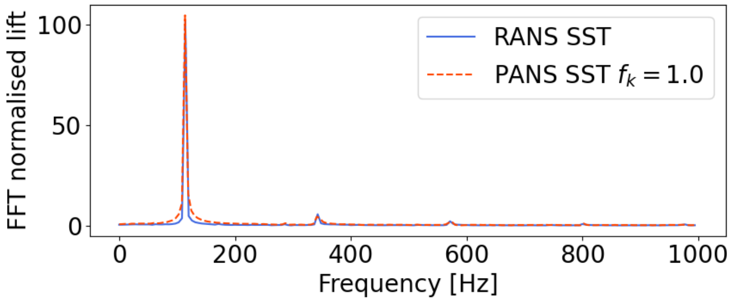


Figure 6.4: Vortex shedding frequency comparison between RANS and PANS SST $f_k = 1.0$

The presence of the small difference is likely to be sourced from the assumption made in Section 3.2.2 whereby it was assumed that at a sufficiently high Re, the resolved fluctuations have no contribution to the SFS energy transport.

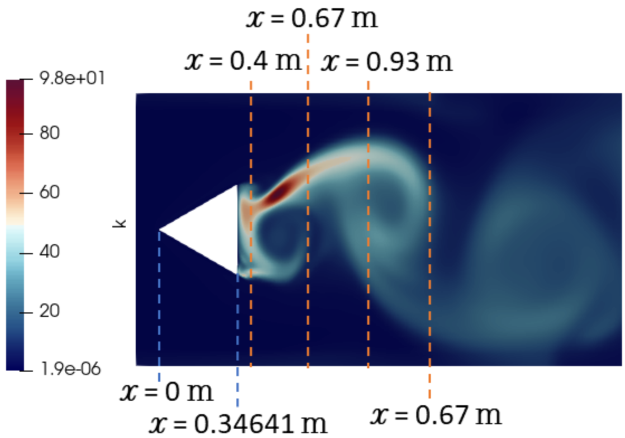


Figure 6.5: Turbulence kinetic energy for PANS at $f_k = 1.0$ marked with various stream-wise positions

Additionally, stream-wise velocity u_x, turbulence kinetic energy k and specific dissipation rate ω are compared and they are presented in Figures 6.6, 6.7 and 6.8 respectively for the stream-wise positions marked with orange dotted lines in Figure 6.5. These positions are relatively close to the triangular prism as compared to the entire length of the domain. The intention was to choose points that have little to no interaction with the top and bottom walls. The mesh is coarse near these walls as shown by Figure 6.2 to save computation time thus dampening out the wall effects such as the additional amount of turbulence kinetic energy. Therefore, to analyse data points that make a fair comparison to the HiFi data in the later part of the project, these stream-wise locations were chosen.

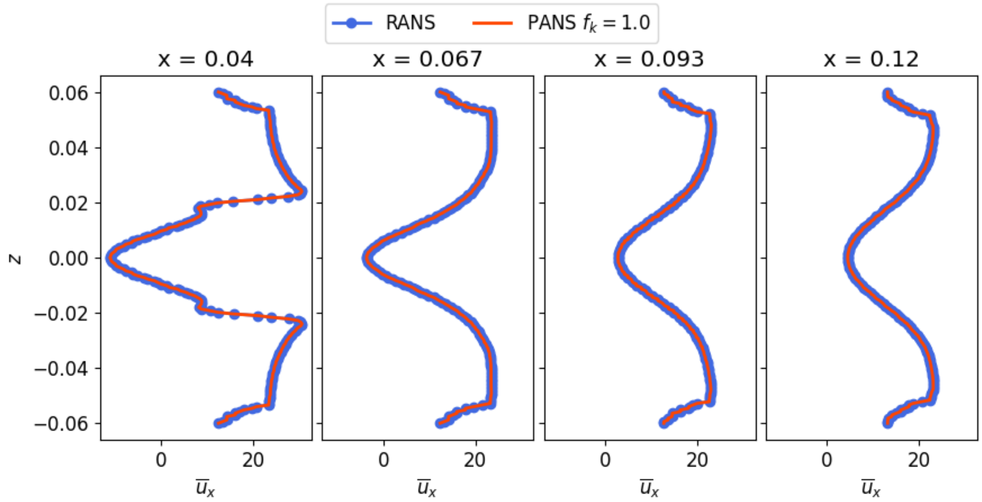


Figure 6.6: Stream-wise velocity comparison between RANS and PANS SST $f_k = 1.0$ at various stream-wise locations

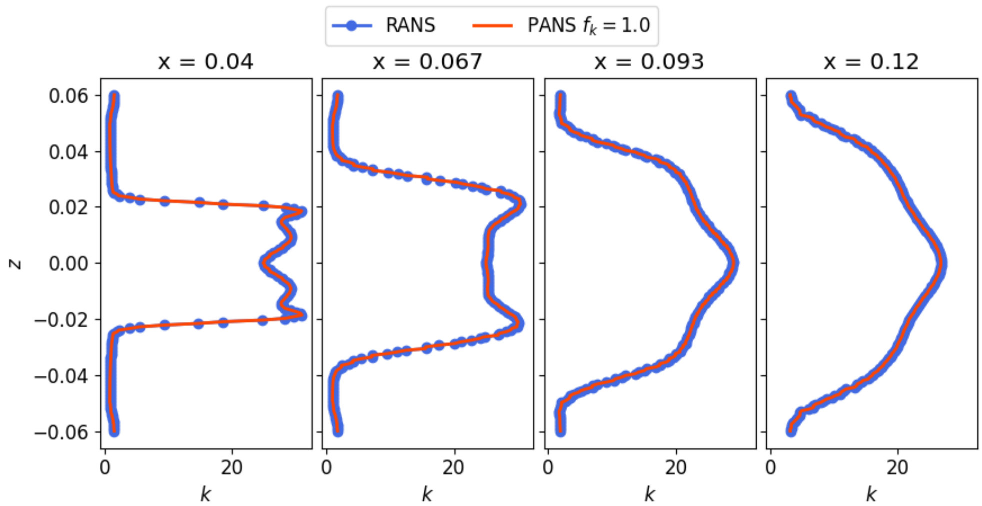


Figure 6.7: Turbulence kinetic energy comparison between RANS and PANS SST $f_k = 1.0$ at various stream-wise locations

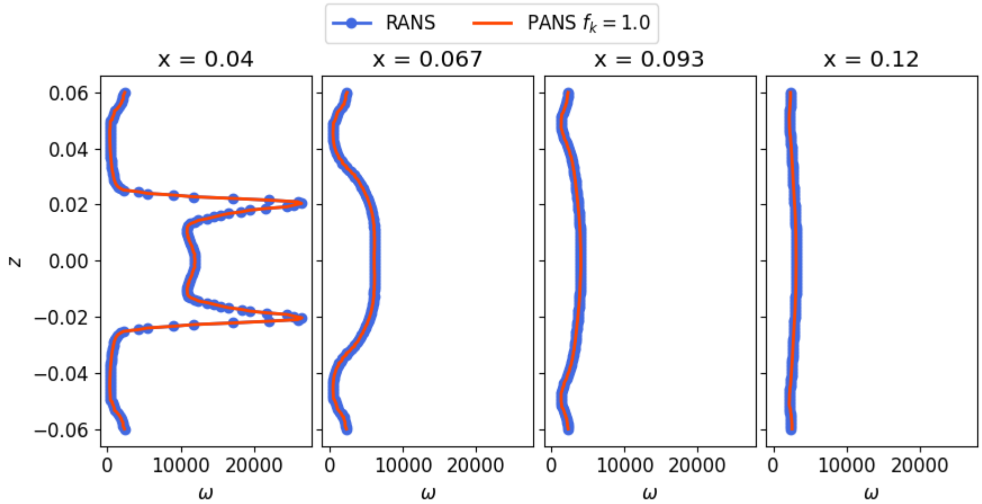


Figure 6.8: Specific dissipation rate comparison between RANS and PANS SST $f_k = 1.0$ at various stream-wise locations

It can be observed that the field values of RANS SST and PANS SST at $f_k = 1.0$ for various stream-wise locations match exactly, validating that the developed PANS model works as expected. This validated PANS model is then be used for further analyses at various values of f_k in Section 6.5.

6.5. Results

Similarly to the analysis of the HiFi data in Section 5.3.2 and the validation that was previously done for the developed PANS model, the vortex shedding frequency f_{vs} was first calculated and the results are shown in Table 6.3. Apart from f_{vs}, the time taken for the computation until stability for each value of f_k is also reported. Stability is considered to be achieved when the periodic solution stops showing large visual changes. After the stability was achieved, 20 flow-through periods were further observed and they were used in computing the vortex shedding frequencies.

f_k	t until stability	f_{vs}
1.0	0.125 s	113.82 Hz
0.8	0.102 s	115.57 Hz
0.6	0.087 s	118.08 Hz

Table 6.3: Time till stability and vortex shedding frequency for PANS at various f_k

It can be observed that as f_k decreases, the vortex shedding frequency approaches 122 Hz, f_{vs} of LES (HiFi) data as shown in Table 5.1. Additionally, the computation takes lesser time to stabilise which is suspected to be due to the smaller value of f_k being able to resolve smaller turbulent scales, especially near the wall where the mesh resolution is extremely high, quickly introducing disturbances to the freestream flow that goes around the triangular prism. It is to be noted that although 0.4 is the next sensible f_k value to be studied, it was not done so as it was claimed in [3] that for a typical RANS mesh resolution, $f_k < 0.44$ does not produce logical results anymore. Furthermore, due to the time constraints of the project, a greater mesh resolution was not used for analyses.

Using the same set of f_k values, analyses at various stream-wise locations for temporal mean values of $\overline{u}_x/$, k and ω were done. These parameters were compared to that of the HiFi values except for ω and they are shown in Figures 6.9, 6.10 and 6.11.

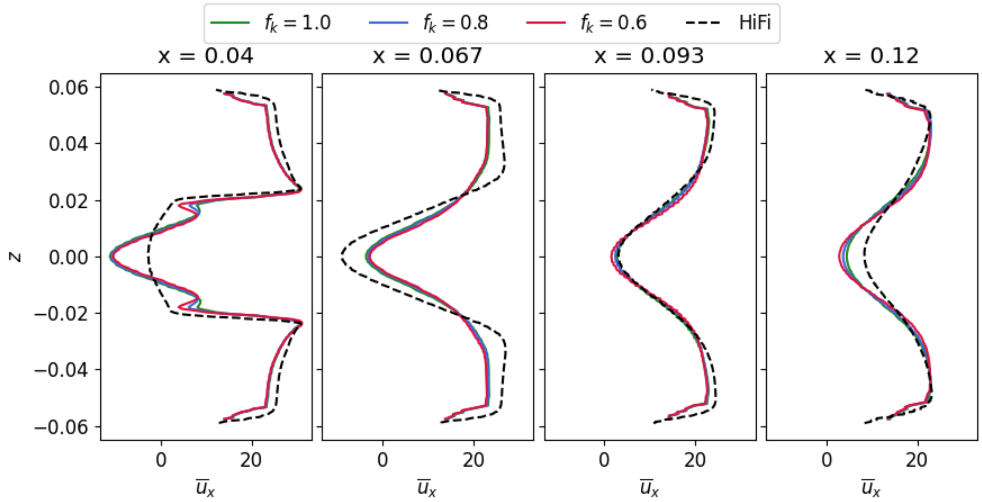


Figure 6.9: Stream-wise velocity comparison between f_k values at various stream-wise locations

Starting with $\overline{u}_x$, at lower values of f_k, the prediction of u_x right behind the prism is improved to a small but significant extent, especially behind the corners of the triangle which takes advantage of the high mesh resolution in this area. However, the solution of the lower f_k gets worse downstream of the fluid domain which was also observed in [3] wherein untrained PANS had a worse prediction for mean stream-wise velocity than untrained URANS. The general characteristics of the solution follow that of typical RANS SST as shown by $f_k = 1.0$.

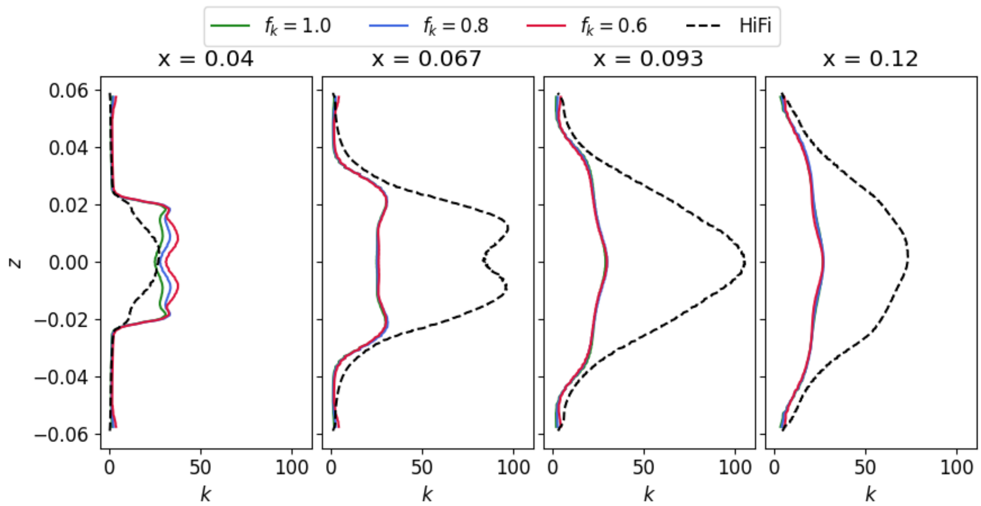


Figure 6.10: Turbulence kinetic energy comparison between f_k values at various stream-wise locations

The prediction for k right behind the prism gets worse for lower f_k, over-predicting to a small extent. However, all the solutions largely under-predict k further downstream with little deviation from one another. This is due to excessive dissipation as shown in Figure 6.11 in which ω is extremely large behind the triangular prism, heavily reducing k.

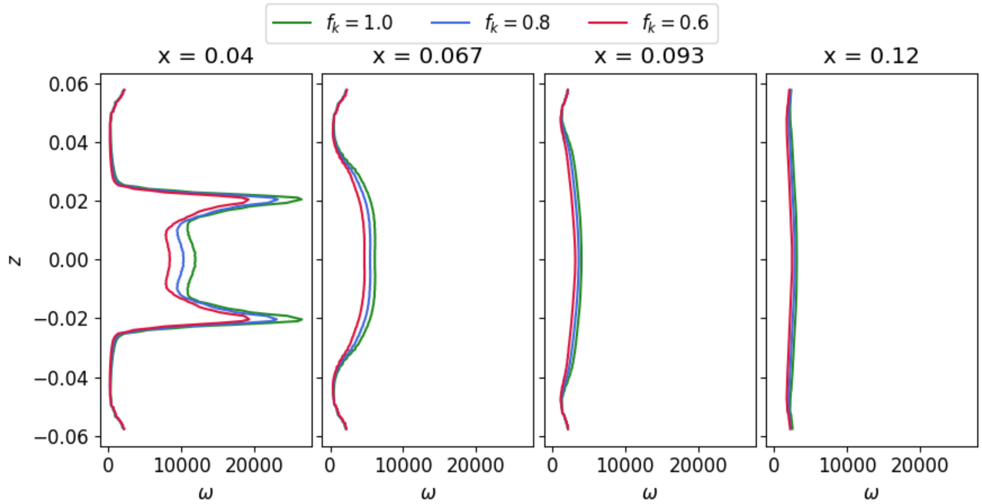


Figure 6.11: Specific dissipation rate comparison between f_k values at various stream-wise locations

Although the results for PANS are unsatisfactory as they deviate from how PANS should perform theoretically, the validation alone is sufficient for a data-driven study that is the main focus of this project. To minimise computational cost and the duration of computation, a mesh of insufficient resolution is used as every small decrement in the value of f_k was found to require a mesh of a much higher resolution in [43]. However, with sufficiently fine mesh, it was proven in [43] that solution of PANS with $f_k = 0$ did indeed approaches the DNS solution at the expense of a huge amount of computational costs.

7

$k-$corrective frozen PANS

This chapter begins with the pre-processing of the HiFi dataset for use in the $k-$corrective frozen PANS method. The implementation of the method into OpenFOAM[1] v2112 is presented alongside feasible verification and validation of the implemented method. The chapter is concluded with an overview of the method and the results obtained.

7.1. Pre-processing

It can be seen from the lift-time plot in Figure 5.6 that the HiFi dataset has varying values of lift amplitudes at every period at different locations in each period. Moreover, although not clearly visible from the plot, the periods have different time values which means that parameters such as the velocity components cannot be simply averaged over the number of available periods.

Thus an averaged period is obtained through "phase-averaging". Here, the lift is used as an example instead of other useful parameters since it is a much better parameter to verify the procedure and validate the results as it clearly shows the sinusoidal behaviour with apparent periods. The following procedure walks through how phase-averaging is done:

1. Roots are extracted from the sinusoidal lift-time plot of Figure 5.7.
2. For every pair of alternating roots that makes up a single sine curve representing a single period, a quadratic function $L(t)$ is found. It takes time as an argument and calculates lift within the root pair.
3. Since the time value of the period is unique for that pair, the function is converted into one for the spherical coordinate system which now has takes phase, $\theta \in [0, 2\pi]$, as its argument giving $L(\theta)$ instead.
4. For a chosen value of resolution $r -$ the number of points in a single period that is much higher than the number of given data points $-$ a fixed set of points in phase coordinates is decided.
5. The lift values in each period are calculated at these points using the previously obtained quadratic function. This allows every r point in each period to be in the exact same locations in a sine curve.
6. Since the points in each period are in the same phase coordinates now, averaging can be applied to get a single phase-averaged period with r number of points. The result is shown in Figure 7.1 in which the horizontal axis is $\theta \in [0, 2\pi \approx 6.28]$.
7. This single phase-averaged period data is converted to the time coordinate. The time value of the period for this phase-averaged data is the average value of all the available periods and this value was found to be $T = 0.008\,256\,\mathrm{s}$.

[1] https://www.openfoam.com/

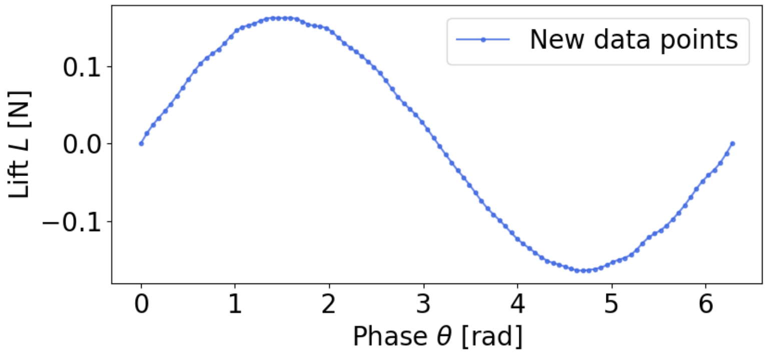


Figure 7.1: Phase averaged lift of HiFi dataset for $r = 100$ data points (example value)

Due to the availability of only 14 full periodic cycles, the phase-averaged lift plot for a single period presented in Figure 7.1 is not fully smooth which is also the case for the three velocity components inevitably. In an optimal setting with a much higher number of cycles, the phase-averaged data should represent a perfect sinusoidal plot. This phase-averaged data of r number of data points representing a single period are used in the frozen$-k$ method.

The averaged data is plotted against the actual data throughout all the available periods in Figure 7.2 for visual verification. With the procedure and the results in lift force verified, the same is done for the three velocity components, turbulence kinetic energy and Reynolds stress tensor as previously presented in (5.3) and (5.4) that are more useful in the frozen$-k$ method.

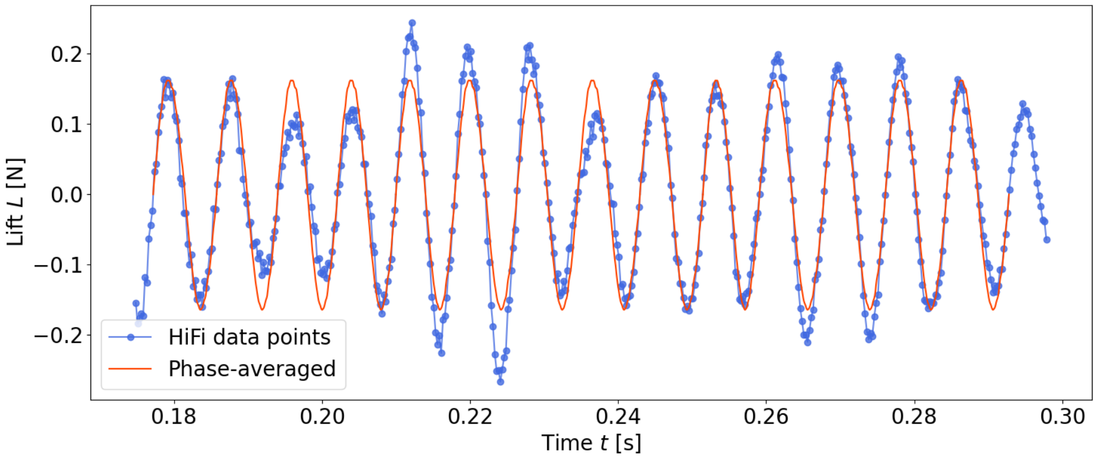


Figure 7.2: Lift and the phase-averaged lift of HiFi dataset against time

So what ultimately was achieved is the triple decomposition of (2.13) that is visualised in Figure 2.2. From the raw data points shown by the blue line in Figure 7.2, the stochastic unsteadiness (the noise) has been removed by averaging all the periods since the partial average of the fluctuation is zero, $\langle u'' \rangle = 0$, whereby the partial averaging represents the averaging the periods in this case. This results in the red line that only includes the mean value ($L = 0$) and the periodic unsteadiness, the sinusoidal fluctuations.

7.2. Frozen$-k$

In this section, a step-by-step procedure of the $k-$corrective frozen PANS method is presented. Using an overview of the method shown in Figure 7.3, each process is elaborated on in detail. This method combines the correction terms of $k-$frozen method of SpaRTA, R and b_{ij}^{Δ} described in Section 4.3.1 and DSCS method of Section 4.3.3 which implements triple decomposition into PANS.

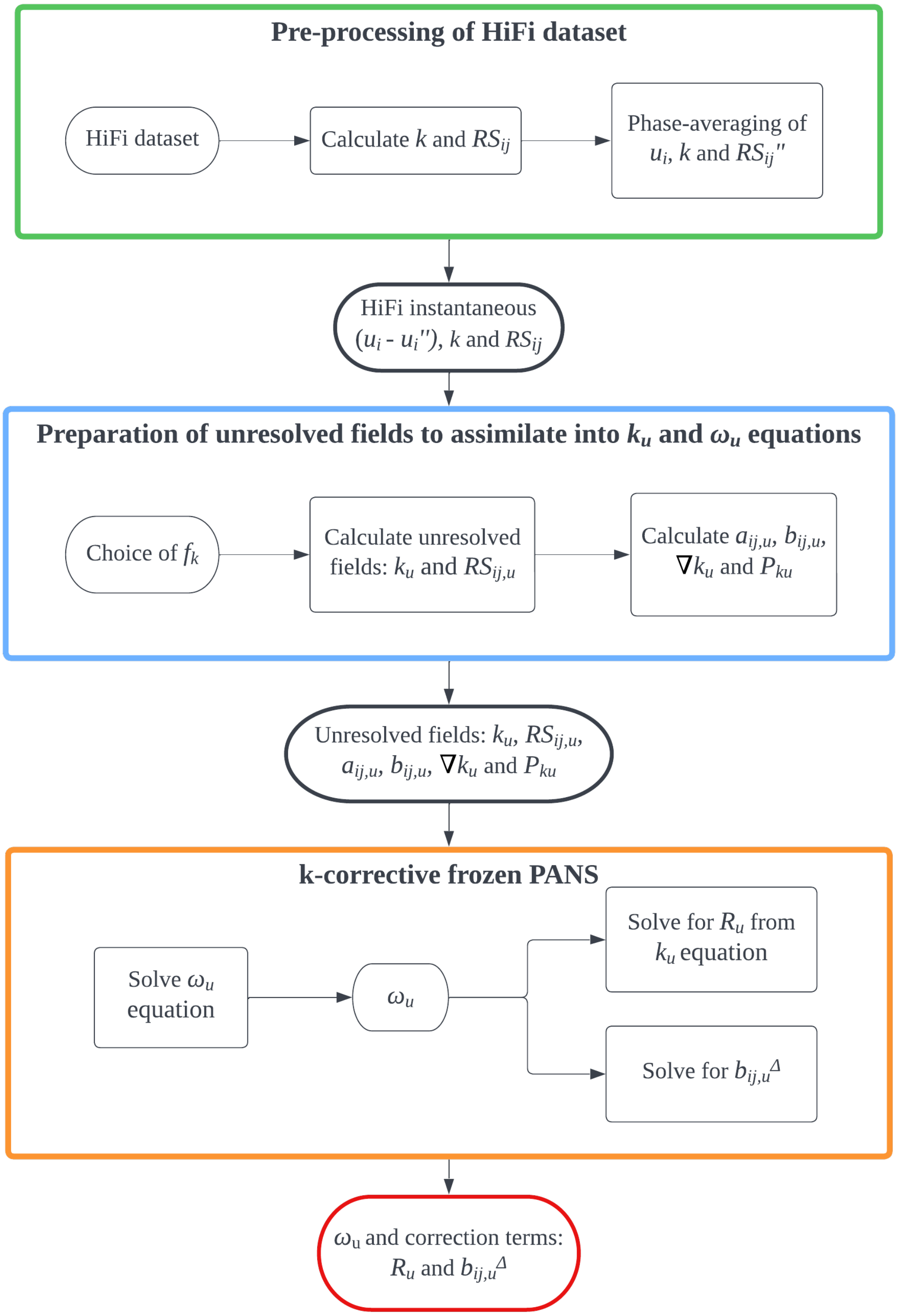


Figure 7.3: Overview of $k-$corrective frozen PANS method

The procedure begins with the pre-processing of the HiFi dataset as described in Section 7.1 and is shown in the green box of Figure 7.3. From the phase-averaging, HiFi instantaneous fields that are required to compute unresolved fields for PANS $k_u - \omega_u$ SST equations are obtained via (5.3) and

(5.4). Similar to the calculation of unresolved turbulence kinetic energy k_u that is done via (3.6), the unresolved Reynolds stress tensor is also achieved using the following equation:

$$RS_{ij,u} = f_k \cdot RS_{ij} \tag{7.1}$$

since the tensor is made up of products of velocity fluctuations just like k. This unresolved Reynolds stress tensor $RS_{ij,u}$ now represents RS''_{ij} following from an assumption that the selected value of f_k coupled with a given mesh is able to exactly drain the turbulent Reynolds stress as given in Section 4.3.3. Using these two parameters, the rest of the required parameters can be calculated:

$$a_{ij,u} = RS_{ij,u} - \frac{2}{3}k_u\delta_{ij} \tag{7.2}$$

$$b_{ij,u} = \frac{a_{ij,u}}{2k_u} \tag{7.3}$$

$$P_{ku} = -RS_{ij,u}\frac{\partial u_i}{\partial x_j} \tag{7.4}$$

with which the preparation part of Figure 7.3 in the blue box is concluded and the unresolved fields are passed on to the orange box where the actual frozen$-k$ is conducted.

As a first step, the ω_u-equation of (6.5) is solved to obtain ω_u $-$ the only unknown in the equation $-$ using the HiFi unresolved fields that were previously calculated together with the other constants that are involved. This equation is presented once more, explicitly pointing out the HiFi fields in **bold**:

$$\begin{aligned}\rho\frac{\partial\omega_u}{\partial t} + \rho\mathbf{u_j}\frac{\partial\omega_u}{\partial x_j} =& \frac{\gamma}{\nu_{tu}^+}\mathbf{P_{ku}} - \gamma\beta^*\rho\omega_u^2 + \gamma\beta^*\rho\frac{\omega_u^2}{f_\omega} - \beta\rho\frac{\omega_u^2}{f_\omega} \\ &+ \rho\frac{\partial}{\partial x_j}\left[\left(\nu + \sigma_{\omega u}\nu_{tu}^+\right)\frac{\partial\omega_u}{\partial x_j}\right] + \rho(1-F_1)CD_{k\omega,u}^+.,\end{aligned} \tag{7.5}$$

where

$$\begin{aligned}\nu_{tu}^+ &= \frac{a_1\mathbf{k_u}}{\max\left(a_1\omega_u, WF_2\right)}, \\ CD_{k\omega,u}^+ &= \max\left(\frac{2\sigma_{\omega 2}}{\omega_u}\frac{f_\omega}{f_k}\nabla\mathbf{k_u}\frac{\partial\omega_u}{\partial x_j}, 10^{-20}\right), \\ W &= \sqrt{2\Omega_{ij}\Omega_{ij}}, \\ \Omega_{ij} &= \frac{1}{2}\left(\frac{\partial\mathbf{u_i}}{\partial x_j} - \frac{\partial\mathbf{u_j}}{\partial x_i}\right), \\ F_1 &= \tanh\left(\arg_{1,u}^4\right), \\ \arg_{1,u} &= \min\left[\max\left(\frac{\sqrt{\mathbf{k_u}}}{\beta^*\omega_u d}, \frac{500\nu}{d^2\omega_u}\right), \frac{4\rho\sigma_{\omega 2}\mathbf{k_u}}{CD_{k\omega,u}^+ d^2}\right], \\ F_2 &= \tanh\left(\arg_{2,u}^2\right) \quad \text{and}\end{aligned}$$

$$\arg_{2,u} = \max\left(2\frac{\sqrt{\mathbf{k_u}}}{\beta^*\omega_u d}, \frac{500\nu}{d^2\omega_u}\right),$$

while ρ, γ, β^*, β, f_ω, f_k, ν, $\sigma_{\omega u}$, $\sigma_{\omega 2}$ and a_1 are constants, and d is, as aforementioned, the distance from the field point to the nearest wall. Additionally, P_{ku} is used instead of P_{ku}^+ since the production term from the HiFi data is considered true value and should not be altered.

The calculated ω_u is then passed on to a modified k_u-equation that was presented in (6.4). Since k_u is a known parameter, the goal is to solve for k_u-equation model error term R_u, similarly to (4.6).

Again implementing **bold** font for HiFi unresolved fields, the modified equation is

$$\rho\frac{\partial \mathbf{k_u}}{\partial t} + \rho\mathbf{u_j}\frac{\partial \mathbf{k_u}}{\partial x_j} = \mathbf{P_{ku}} - \beta^*\rho\omega_u\mathbf{k_u} + \rho\frac{\partial}{\partial x_j}\left[\left(\nu + \sigma_{ku}\nu_{tu}^+\right)\frac{\partial \mathbf{k_u}}{\partial x_j}\right] + R_u$$
$$R_u = \rho\frac{\partial \mathbf{k_u}}{\partial t} + \rho\mathbf{u_j}\frac{\partial \mathbf{k_u}}{\partial x_j} - \mathbf{P_{ku}} + \beta^*\rho\omega_u\mathbf{k_u} - \rho\frac{\partial}{\partial x_j}\left[\left(\nu + \sigma_{ku}\nu_{tu}^+\right)\frac{\partial \mathbf{k_u}}{\partial x_j}\right] \tag{7.6}$$

from which R_u is explicitly solved. While doing so, the other correction term, $b_{ij,u}^{\Delta}$, is calculated from the following equation:

$$b_{ij,u}^{\Delta} = \mathbf{b_{ij,u}} - \left(-\frac{\nu_{tu}^+}{\mathbf{k_u}}S_{ij}\right). \tag{7.7}$$

This term ultimately corrects for the model error in the Boussinesq eddy viscosity assumption in which the error is driven by the anisotropy term that this frozen$-k$ method tries to correct for.

There are two main features of this method that sets it apart from the DSCS method of Section 4.3.3. The first is the inclusion of R_u, the k_u-equation correction term. The second is directly influenced by the first feature. It is the assimilation of HiFi k_u data into the k_u-equation instead of solving the differential equation. Thus, the method is hugely driven by the assumption that a specific value of f_k can correctly drain the stochastic unsteadiness to some extent.

7.3. OpenFOAM implementation

Unlike the standard PANS turbulence model that requires just the initial conditions to be given to the program as described in Section 6.3, for k-corrective frozen PANS, an externally provided HiFi dataset needs to be used at each time step. However, it is not ideal for storage to have the value of the resolution r introduced in Section 7.1 to be large enough to heavily minimise the time step Δt to account for the CFL condition:

$$C = u\frac{dt}{dx} \leq 1. \tag{7.8}$$

This is especially since the mesh carries $y^+ < 1$ on the walls in resolving the near wall flows which drive dx to an extremely small number. Through a trial run of standard RANS/PANS computation on OpenFOAM, it was realised that $dt \approx 1 \times 10^{-8}$ s to abide by (7.8) which would require $r \approx 82\,000$. Computational storage-wise this is extremely undesirable to have such a large number of folders with large data files to be read from. Furthermore, OpenFOAM features adjustable time steps which are hard to track before computation.

Thus, with the value of $r = 166$ that was chosen based on fitting in a time step of $dt_{HiFi} = 5 \times 10^{-5}$ s which results in having clean decimal places for folder names of the calculated average period of $T \approx 0.008\,256\,2$ s (so at $t = 0$ s, $0.000\,05$ s, 0.0001 s, $0.000\,15$ s, $\cdots$, $0.008\,25$ s). Using this temporal resolution, linear interpolation was implemented to easily account for the flexible time steps. From Figure 7.1, linear interpolation was deemed to be sufficient due to the proximity between the adjacent points. After a computational time of $t = 0.008\,25$ s where the time of the last HiFi data containing folder has been passed, the remainder of the division between instantaneous time t and the period T is used to recycle this set of r number of folders. The procedure is elaborated in algorithm 2 and notable snippets of the lines of code is presented in Appendix B.

Additionally, unlike standard PANS implementation, the PIMPLE solver is adjusted since the usage of HiFi velocity data makes the velocity-pressure coupling of the PIMPLE algorithm redundant and this is shown in Appendix C. In algorithm 2, this is given the name “frozen-PIMPLE”.

Algorithm 2 k-corrective frozen PANS $k_u - \omega_u$ SST

Require: f_k ▷ f_ω is simply an inverse of f_k
Require: T ▷ Period of HiFi dataset
  Read u_0, p_0, k_0, ω_0, $\nu_{t,0}$, $RS_{ij,0}$ ▷ Read from $t = 0$ folder
  $k_{u,0} \leftarrow k_0 \times f_k$
  $\omega_{u,0} \leftarrow \omega_0 / f_k$
  $\nu_{tu,0} \leftarrow k_{u,0} / \omega_{u,0}$
  $\mathcal{T} \leftarrow (0, 0.00005, 0.00010, \cdots, 0.00825)$
  while $t \leq t_{final}$ **do**
    procedure PARENT FOLDERS IDENTIFICATION AND INTERPOLATION
      $\theta \leftarrow t \% T$ ▷ Remainder of t/T
      if $\mathcal{T}[-1] < \theta < T$ **then** ▷ Between the time of last available folder and period
        $idx^+ = 0$ ▷ Upper bound index
        $idx^- = -1$ ▷ Lower bound index
      else
        for $k \leftarrow (0, 1, 2, ..., r-1)$ **do** ▷ $r = 83$
          if $\mathcal{T}[k] > \theta$ **then**
            $idx^+ = k$
            $idx^- = k - 1$
            break ▷ Exit for-loop early
          end if
        end for
      end if
      $t^+ \leftarrow \mathcal{T}[idx^+]$
      $t^- \leftarrow \mathcal{T}[idx^-]$
      $\phi = \{u, k, RS_{ij}\}$
      Read ϕ_{t+}, ϕ_{t-} ▷ Read from $t = t^+, t^-$ folder
      $\phi_t \leftarrow (\phi_{t+} - \phi_{t-})/0.0001 \times \theta + \phi_{t-}$ ▷ Linear interpolation
      $k_{u,t} \leftarrow k_t \times f_k$
      $RS_{ij,u,t} \leftarrow RS_{ij,t} \times f_k$
    end procedure
    procedure FROZEN-PIMPLE SOLVER(u_t)
      $i \leftarrow 0$ ▷ i: Iteration counters
      while $\omega_{u,i} - \omega_{u,i-1} >$ tolerance$_\omega$ **do**
        $\omega_{u,i} \leftarrow$ Solve ω_u-equation of (6.5)
      end while
      $\omega_{u,t} \leftarrow \omega_{u,i}$
      $\omega_t \leftarrow \omega_{u,t} \times f_k$
      $R \leftarrow$ Solve via (7.6)
      $\nu_{tu,t} \leftarrow k_{u,t} / \omega_{u,t}$ ▷ Update unresolved kinematic eddy viscosity
      $b^{\Delta}_{ij,u} \leftarrow$ Solve via (7.7)
    end procedure
  end while

7.4. Results and discussion

In this section, the results obtained for the k-corrective frozen PANS method are presented. The corrections, R_u and $b^{\Delta}_{ij,u}$ obtained are plotted and analysed.

The corrections made for $f_k = 0.8$ are presented in Figures 7.4, 7.5 and 7.6. R_u, the model error for the k_u-equation is compared against the unresolved production term P_{ku} derived from Boussinesq approximation and HiFi data in Figure 7.4. It is to be noted that asymmetry is present in the HiFi data to a small extent and although it is not obvious in Figures 6.9, 6.10 and 6.11, this minor asymmetry blows up when R_u is extracted from (7.6) as it can be observed.

In the regions of the wake that are right behind the triangular prism and far downstream at $x = 0.04$ and $x = 0.012$ respectively, the order of magnitudes of R_u stays in line with that of P_{ku} of Boussinesq approximation and HiFi data. At all stream-wise locations, R_u is not limited to just positive nor negative values but it fluctuates between them. At $x = 0.04$, a significant positive value of correction is required at span-wise central point ($z = 0$). The correction quickly switches to negative values as z approaches ± 0.02 which are the span-wise coordinates of the trailing edge corners of the triangle. Right outside these corners, at $|z| > 0.02$, the correction again switches back to a positive value before gradually reducing to 0 henceforth.

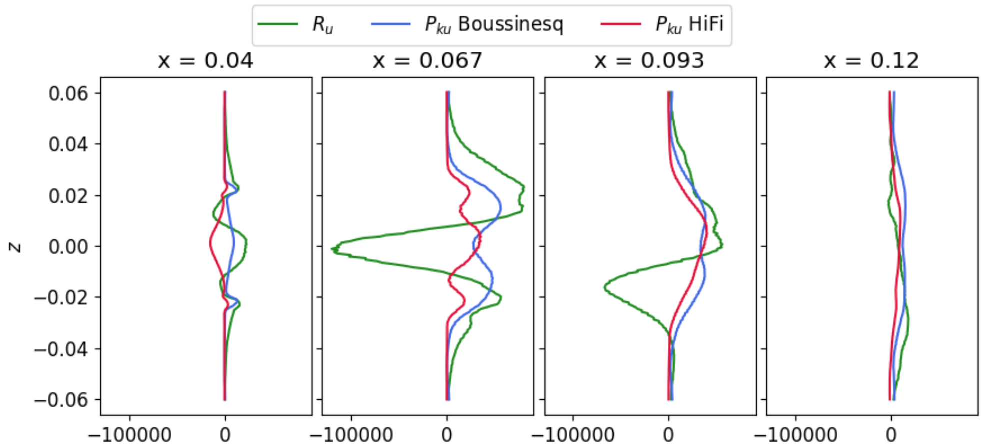


Figure 7.4: R_u compared with $P_{k,u}$ for $f_k = 0.8$

At $x = 0.067$, large magnitudes of R_u are observed. Contrary to R_u at $x = 0.04$, a large negative correction is required at $z = 0$. Once again, the sign of R_u changes as $z = \pm 0.02$ is approached. The positive values of R_u then reduce to 0 towards the ends. The asymmetry is the most visible at $x = 0.093$ wherein the large negative peak in R_u has moved towards the negative value of z. Towards the top side (positive z), a clear transition to positive value can be noticed. Although to a much smaller extent, this transition is also observed towards the bottom side. Nonetheless, due to the involvement of many other terms in the transport equation for k_u, little can be said about the relationship between these parameters. A mere conclusion that can be made is that the k_u-equation requires a significant amount of corrections, especially in the downstream area, in the wake of the triangular prism. Additionally, since R_u is small at $x = 0.12$ even when the difference between k_u of PANS and HiFi is large as shown in Figure 6.10, the correction for b_{ij} was expected to be more significant and this is indeed observed in Figures 7.5 and 7.6. In these figures, $b^{\Delta}_{12,u}$ and $b^{\Delta}_{12,u}$ are presented. They are the normal and shear stress components of the second model error, $b^{\Delta}_{ij,u}$, and they are plotted alongside $b_{ij,u}$ of Boussinesq approximation and HiFi data, generally expressed as

$$b^{\text{Bouss}}_{ij,u} = -\frac{\nu_{tu}}{2k_u} S_{ij} \quad \text{and} \tag{7.9}$$

$$b^{\text{HiFi}}_{ij,u} = \frac{1}{2k^{\text{HiFi}}_u} \left(RS^{\text{HiFi}}_{ij,u} - \frac{2}{3} k^{\text{HiFi}}_u \delta_{ij} \right). \tag{7.10}$$

It can be observed that the addition of $b^{\Delta}_{ij,u}$ onto $b_{ij,u}$ of Boussinesq approximation results in the HiFi $b_{ij,u}$ as intended. Unlike R_u, this model error is concentrated in certain regions in the fluid domain but is distributed throughout the domain, in both stream-wise and span-wise directions. For $b^{\Delta}_{11,u}$ that represents the streamwise normal stress component, the most noticeable trait is behind the centre of the triangular prism at $x = 0.04$ where it is largely negative. The sign of $b_{11,u}$ is incorrectly predicted by Boussinesq approximation resulting in an oppositely shaped plot. Towards the top and bottom walls of the domain ($z = \pm 0.06$), positive values of corrections are observed before converging to a 0 value at the walls. Similarly for $b^{\Delta}_{12,u}$ in Figure 7.6, the plots of Boussinesq approximation's $b_{12,u}$ and that of HiFi are almost a mirror image of each other at $x = 0.04$. Hence, it can be concluded that computation of $b_{ij,u}$ by Boussinesq approximation is the poorest in the near downstream region.

Further downstream, the disparity of the general trend of Boussinesq approximation from the HiFi counterpart reduces but with significant corrections to be made nonetheless. However, the direction of the stress prediction large improves as it can be seen by both blue and red lines being on the same side, except for at the top and bottom walls. Among the presented stream-wise locations, $x = 0.093$ is where prediction by Boussinesq approximation for $b_{ij,u}$ is closest to HiFi dataset. In the far downstream region, represented by $x = 0.12$ plot, a significant amount of corrections is still required. As highlighted previously, the disparity in $b_{ij,u}$ is largely responsible for the poor prediction of k_u for PANS since R_u at $x = 0.12$ is relatively minimal.

Summing up, the largest corrections are required at the centre-line along the span-wise $z-$axis through all points on $x-$axis. $b^{\Delta}_{ij,u}$ seems to have bigger impacts than R_u in the near and far downstream regions, at $x = 0.04$ and $x = 0.12$ respectively. Between these two points, both types of corrections work together to correct for the disparity between PANS and HiFi solutions shown in Figure 6.10.

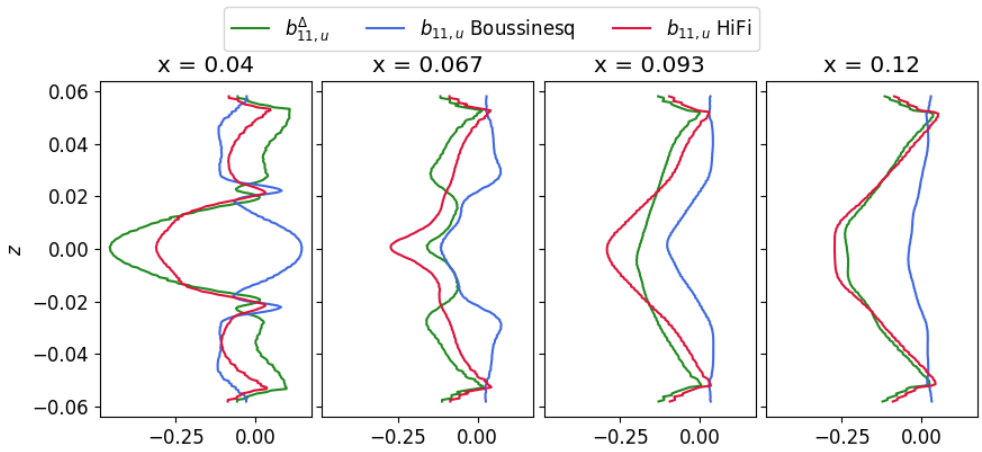


Figure 7.5: $b^{\Delta}_{11,u}$ compared with Boussinesq $b_{11,u}$ and HiFi $b_{11,u}$ for $f_k = 0.8$

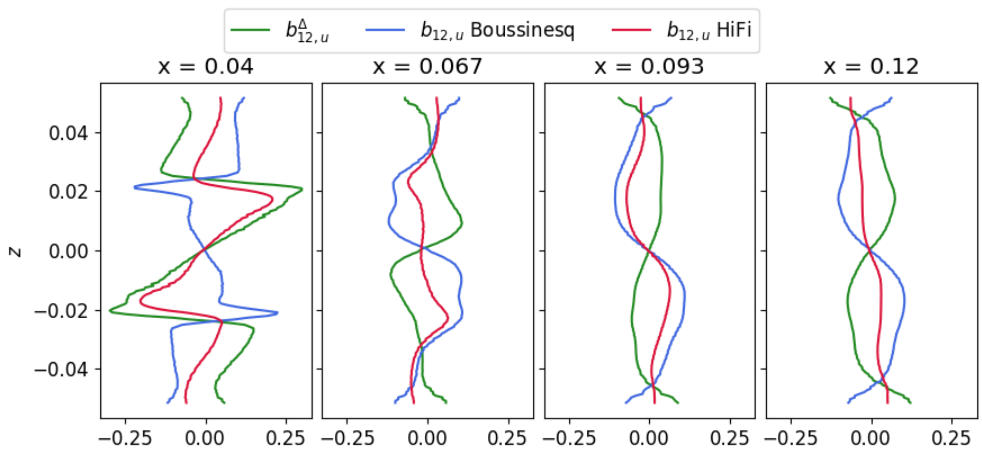


Figure 7.6: $b^{\Delta}_{12,u}$ compared with Boussinesq $b_{12,u}$ and HiFi $b_{12,u}$ for $f_k = 0.8$

Similarly, the corrections for $f_k = 0.6$ are shown in Figures 7.7, 7.8 and 7.9 of R_u, $b^{\Delta}_{11,u}$ and $b^{\Delta}_{12,u}$ respectively. Although all three variables show the same trends in both stream-wise and span-wise directions as the corrections computed for $f_k = 0.8$, differences exist. However, these differences are not clear from these plots.

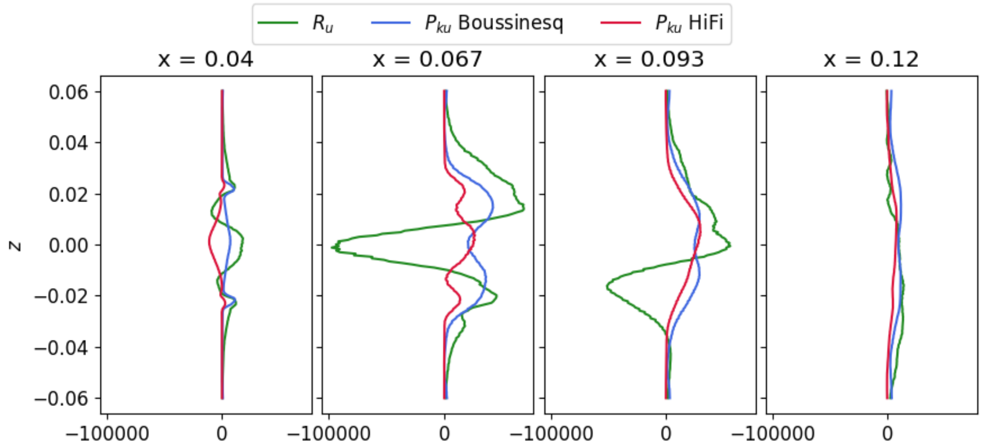


Figure 7.7: R_u compared with $P_{k,u}$ for $f_k = 0.6$

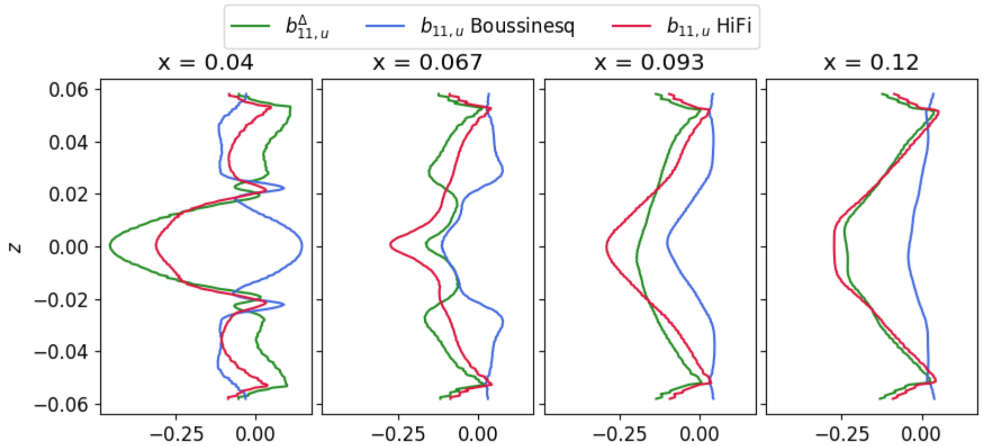


Figure 7.8: $b^{\Delta}_{11,u}$ compared with Boussinesq $b_{11,u}$ and HiFi $b_{11,u}$ for $f_k = 0.6$

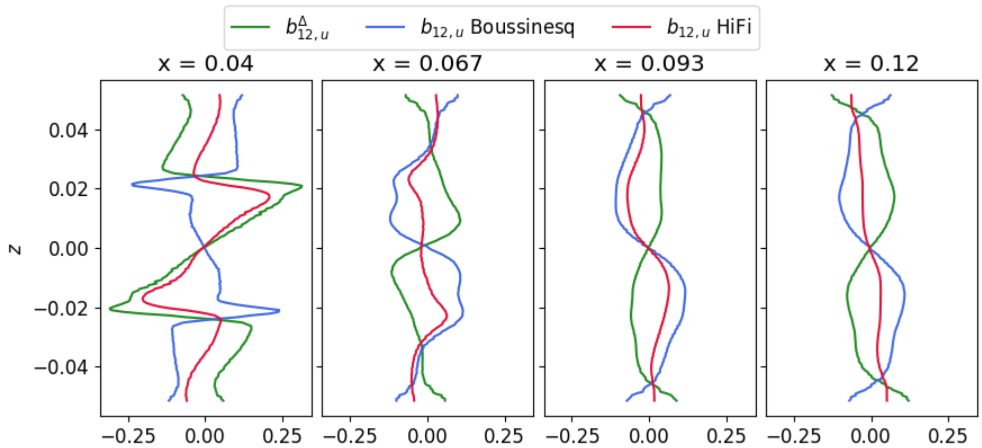


Figure 7.9: $b^{\Delta}_{12,u}$ compared with Boussinesq's $b_{12,u}$ and HiFi $b_{12,u}$ for $f_k = 0.6$

The absolute values of the corrections, $|R_u|$, $|b^{\Delta}_{11,u}|$ and $|b^{\Delta}_{12,u}|$, are thus plotted for three different values of f_k in Figures 7.10, 7.11 and 7.12 respectively for an obvious comparison. Since the general trends of the corrections in the domain are presented above, the absolute values are presented for clearer distinctions between the corrections of various f_k values.

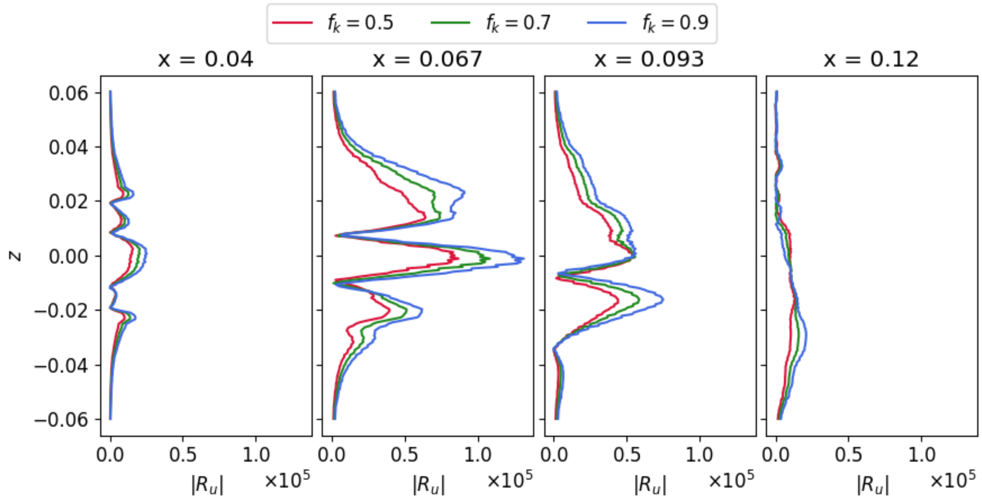


Figure 7.10: $|R_u|$ compared between different f_k values

In general, significantly less correction for k_u-equation, R_u, is required with decreasing f_k (meaning smaller portion of the flow is modelled) as shown by Figure 7.10. Although this is an expected behaviour since smaller values of the parameters in the k_u-equation are involved in solving the equation, it proves that even with smaller model errors, PANS can be corrected to achieve the HiFi solution and that the variation in f_k is indeed directly related to outcome of the corrections. This is a crucial result since solving smaller fraction of the k_u-equation does not necessarily result in smaller correction as seen at $x, z = (0.12, 0.01)$ in which there still is somewhat a linear relationship between f_k and R_u. Furthermore, smaller corrections, especially the smaller fluctuations through the peaks and troughs, are beneficial in terms of stability for training the machine learning algorithm which is one of the suggested steps to take next.

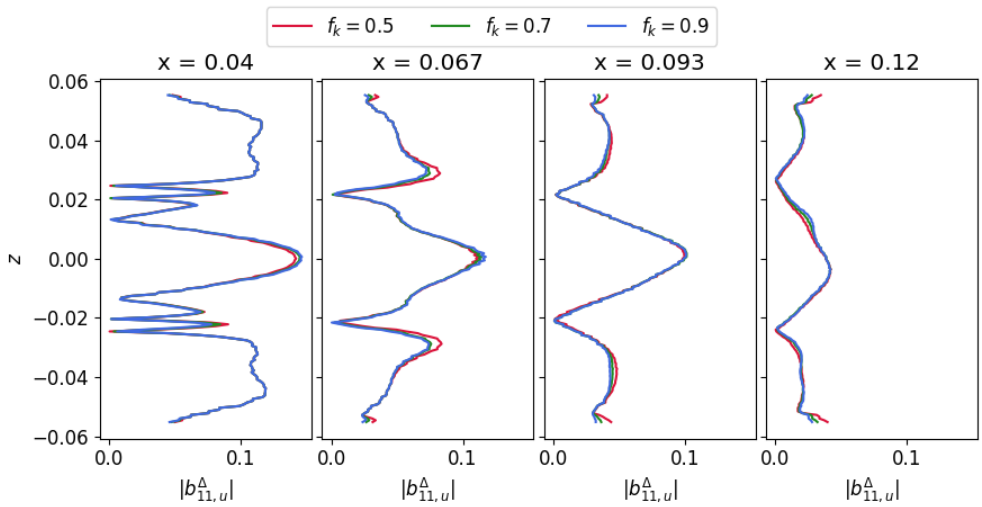


Figure 7.11: $|b^{\Delta}_{11,u}|$ compared between different f_k values

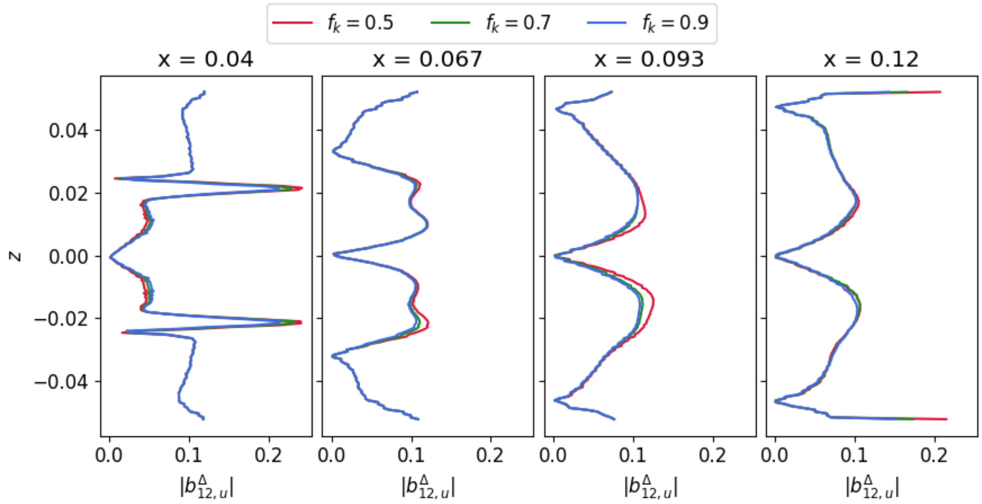


Figure 7.12: $|b^{\Delta}_{12,u}|$ compared between different f_k values

For both the stream-wise normal stress and shear stress components of $b^{\Delta}_{ij,u}$ shown in Figure 7.11 and Figure 7.12, no large deviations between the three values of f_k is shown. Obvious differences come from region $x, z = (0.067, \pm 0.025)$ and $x, z = (0.093, \pm 0.01)$, which are the regions behind the two trailing edge corners of the triangular prism, lower values of f_k require slightly more corrections. These regions share a similarity where they are high vorticity regions due to interactions with walls. The reason behind the difference comes from the difference in definitions of the $b^{\text{Bouss}}_{ij,u}$ and $b^{\text{HiFi}}_{ij,u}$ shown in (7.9) and (7.10) respectively. The former has its numerator, ν_{tu}, derived from ν_t through product of f_k^2 as deduced from (3.18) while the latter's numerator is derived by multiplying f_k as shown in (7.1). Hence, the shear stress component for the Boussinesq approximation is further reduced by an additional product of f_k compared to the HiFi counterpart, resulting in a larger deviation with decreasing f_k.

8

Conclusion and recommendations

8.1. Conclusion

Data-driven turbulence modelling has shown its effectiveness and has seen success in different applications involving various methods and flow settings as covered in Chapter 4. PANS has also been proved in [43] to perform as it is theoretically intended where it approaches DNS solution as f_k is reduced. In this project, data-driven PANS was studied and performed to augment both approaches to CFD. From the perspective of data-driven studies, it is always preferred to have a more accurate solution to start with. As for PANS, although it has the potential to give DNS quality results, too much computational cost is incurred and thus a data-driven approach can bridge PANS to HiFi solution with minimal computational cost. Using this goal and the gaps found in existing literature, a research objective was first established in Chapter 1 alongside the main research question which was:

> *"How can $k-$corrective frozen RANS and DSCS be combined to be implemented into the $k_u - \omega_u$ SST PANS turbulence model for improvement in the prediction of turbulent flows around triangular prism?"*

In answering this question, the characteristics of the mentioned data-driven methods: $k-$corrective frozen RANS method and DSCS were studied in Sections 4.3.1 and 4.3.3 respectively. Additionally, their drawbacks and potential improvements were discussed in Section 4.3.4.

A revised version of the existing $k-$corrective frozen RANS method which was implemented in SpaRTA method [2] was attempted to be appended to DSCS. Unlike other data-driven methods, the frozen RANS approach aims to obtain two model error terms, one for the $k-$equation of a typical two-equation RANS turbulence model labelled as R and one for the Boussinesq approximation of the anisotropy tensor term labelled as b^{Δ}_{ij}. These terms are recovered using the HiFi dataset which is injected into the RANS turbulence model. However, the frozen RANS was implemented into a steady flow case where flow variables are fixed in time.

DSCS attempted to extract a model error for the anisotropy term of Boussinesq approximation from PANS in an unsteady flow case. It takes advantage of the idea behind PANS whereby the turbulence model handles only the unresolved portion of the flow thus injecting just a fraction of the Reynolds stress tensor into the chosen turbulence model while the other part of the fraction is taken up by the PANS equation. However, DSCS does not fully utilise the HiFi dataset nor correct the $k-$equation's model error.

Therefore, combining these two methods' favourable traits was deemed ideal to make up for each other's shortcomings and this was attempted in the project. Triple decomposition was used to extract the periodic unsteadiness component from the flow velocity of the HiFi dataset and this was injected into PANS to derive correction terms: R_u and $b^{\Delta}_{ij,u}$ which are the model error of unresolved $k-$equation and normalised unresolved anisotropy tensor term of Boussinesq approximation at every time step of an unsteady flow case.

There are multiple stages in a typical data-driven turbulence modelling including injection of the extracted correction terms back into the turbulence model, finding a suitable machine learning algorithm and training it to discover some properties of the flow physics. However, the project's focus was on developing a $k_u - \omega_u$ SST turbulence model for PANS in the OpenFOAM program and incorporating the model with the frozen RANS and DSCS methods to obtain the aforementioned unresolved correction terms.

8.2. Recommendations for future work

During and after the project, it was realised that many other studies need to be done to fill the voids of the project to enhance this data-driven PANS method into a complete and usable one to make it a reliable mainstream model.

Firstly, a machine learning study of obtaining optimal values for f_k is a crucial study to be done. Whether it is for fixed f_k or only spatially varying f_k or both spatially and temporally varying f_k, it is important to know the relationship it has with a specific mesh in a fluid domain for a specific flow case. Although it has been proved that smaller values of f_k require finer meshes, a suitable mesh is only discovered after rounds of trial and error as it was done in [43]. Furthermore, all the equations that relate f_k to cell sizes and characteristic scales of turbulence have insufficient theories to back them up and they are empirically chosen. A data-driven study that discovers a general equation for f_k would be extremely useful.

A natural follow-up to this thesis project is to test how the correction terms help the PANS turbulence model in approaching the HiFi solution by injecting the R_u and $b^{\Delta}_{ij,u}$ terms into k_u-equation and ω_u equation. Although it has been proven in [2] that the $k-$corrective frozen RANS method does indeed recover the HiFi solution in a steady flow case, the method has not yet been tested in an unsteady flow case which is a lot trickier with the injection of correction terms.

Lastly, a 3D study for the $k-$corrective frozen PANS method with sufficient computational budge would be a useful extension to the project. Although DSCS has proved that 2D analyses are sufficient for PANS, the combined method of the project deviates from DSCS a fair amount thus the statement should be reaffirmed for this specific method. Furthermore, in [43] where the solution of PANS has successfully converged to DNS solution, a 3D fluid domain was used. However, it is expected that a substantial increase in computational cost will be incurred.

References

[1] Charles G. Speziale. "Computing non-equilibrium turbulent flows with time-dependent rans and vles". In: *Fifteenth International Conference on Numerical Methods in Fluid Dynamics*. Vol. 490. Series Title: Lecture Notes in Physics. Springer Berlin Heidelberg, 1997, pp. 123–129. ISBN: 978-3-540-63054-8. DOI: 10.1007/BFb0107089. URL: http://link.springer.com/10.1007/BFb0107089 (visited on 06/02/2022).

[2] Martin Schmelzer, Richard P. Dwight, and Paola Cinnella. "Discovery of Algebraic Reynolds-Stress Models Using Sparse Symbolic Regression". In: *Flow, Turbulence and Combustion* 104.2 (Mar. 2020), pp. 579–603. ISSN: 1386-6184, 1573-1987. DOI: 10.1007/s10494-019-00089-x. URL: http://link.springer.com/10.1007/s10494-019-00089-x (visited on 11/01/2021).

[3] Chitrarth Lav, Richard D. Sandberg, and Jimmy Philip. "A framework to develop data-driven turbulence models for flows with organised unsteadiness". In: *Journal of Computational Physics* 383 (Apr. 2019), pp. 148–165. ISSN: 00219991. DOI: 10.1016/j.jcp.2019.01.022. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999119300476 (visited on 11/01/2021).

[4] Michael Leschziner. *Statistical Turbulence Modelling for Fluid Dynamics - Demystified*. en. United Kingdom: Imperial College Press, 2016. ISBN: 978-1-78326-661-6. URL: https://doi.org/10.1142/p997.

[5] Osborne Reynolds. "IV. On the dynamical theory of incompressible viscous fluids and the determination of the criterion". en. In: *Philosophical Transactions of the Royal Society of London. (A.)* 186 (Dec. 1895), pp. 123–164. ISSN: 0264-3820, 2053-9231. DOI: 10.1098/rsta.1895.0004. URL: https://royalsocietypublishing.org/doi/10.1098/rsta.1895.0004 (visited on 11/11/2021).

[6] A Favre. "Statistical equations of turbulent gases(Statistical equations for compressible gas, discussing turbulent quantities separated into fluctuating and macroscopic parts)". In: (1969).

[7] H.E. Fiedler. "Coherent structures in turbulent flows". en. In: *Progress in Aerospace Sciences* 25.3 (Jan. 1988), pp. 231–269. ISSN: 03760421. DOI: 10.1016/0376-0421(88)90001-2. URL: https://linkinghub.elsevier.com/retrieve/pii/0376042188900012 (visited on 11/21/2021).

[8] A. K. M. F. Hussain and W. C. Reynolds. "The mechanics of an organized wave in turbulent shear flow". In: *Journal of Fluid Mechanics* 41.2 (Apr. 13, 1970), pp. 241–258. ISSN: 0022-1120, 1469-7645. DOI: 10.1017/S0022112070000605. URL: https://www.cambridge.org/core/product/identifier/S0022112070000605/type/journal_article (visited on 07/16/2022).

[9] F.X. Trias, A. Gorobets, and A. Oliva. "Turbulent flow around a square cylinder at Reynolds number 22,000: A DNS study". In: *Computers & Fluids* 123 (Dec. 2015), pp. 87–98. ISSN: 00457930. DOI: 10.1016/j.compfluid.2015.09.013. URL: https://linkinghub.elsevier.com/retrieve/pii/S0045793015003254 (visited on 11/16/2021).

[10] J. Boussinesq. "Essai sur la théorie des eaux courantes". fr. In: *Mémoires présentés par divers savants à l'Académie des Sciences* 23 (1877), pp. 1–680.

[11] Aidan Wimshurst. *[CFD] Eddy Viscosity Models for RANS and LES*. Fluid Mechanics 101, Youtube. 2021. URL: https://www.youtube.com/watch?v=SVYXNICeNWA.

[12] François G. Schmitt. "About Boussinesq's turbulent viscosity hypothesis: historical remarks and a direct evaluation of its validity". In: *Comptes Rendus Mécanique* 335.9 (Sept. 2007), pp. 617–627. ISSN: 16310721. DOI: 10.1016/j.crme.2007.08.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S1631072107001386 (visited on 11/14/2021).

[13] Nicholas M. Hamilton and Raúl B. Cal. "Characteristic shapes of the normalized Reynolds stress anisotropy tensor in the wakes of wind turbines with counter-rotating rotors". In: 17th International Symposium on Applications of Laser Techniques to Fluid Mechanics (July 2014).

[14] P. R. Spalart and S. R. Allmaras. "A One-Equation Turbulence Model for Aerodynamic Flows". In: *Recherche Aerospatiale* 1 (1994), pp. 5–21.

[15] Georgi Kalitzin et al. "Near-wall behavior of RANS turbulence models and implications for wall functions". In: *Journal of Computational Physics* 204.1 (Mar. 2005), pp. 265–291. ISSN: 00219991. DOI: 10.1016/j.jcp.2004.10.018. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999104004164 (visited on 06/25/2022).

[16] L. Eça et al. "A manufactured solution for a two-dimensional steady wall-bounded incompressible turbulent flow". In: *International Journal of Computational Fluid Dynamics* 21.3 (Mar. 2007), pp. 175–188. ISSN: 1061-8562, 1029-0257. DOI: 10.1080/10618560701553436. URL: http://www.tandfonline.com/doi/abs/10.1080/10618560701553436 (visited on 06/01/2022).

[17] Michael L. Shur et al. "Turbulence Modeling in Rotating and Curved Channels: Assessing the Spalart-Shur Correction". In: *AIAA Journal* 38.5 (May 2000), pp. 784–792. ISSN: 0001-1452, 1533-385X. DOI: 10.2514/2.1058. URL: https://arc.aiaa.org/doi/10.2514/2.1058 (visited on 06/01/2022).

[18] W.P Jones and B.E Launder. "The prediction of laminarization with a two-equation model of turbulence". In: *International Journal of Heat and Mass Transfer* 15.2 (Feb. 1972), pp. 301–314. ISSN: 00179310. DOI: 10.1016/0017-9310(72)90076-2. URL: https://linkinghub.elsevier.com/retrieve/pii/0017931072900762 (visited on 06/25/2022).

[19] Kuei-Yuan Chien. "Predictions of Channel and Boundary-Layer Flows with a Low-Reynolds-Number Turbulence Model". In: *AIAA Journal* 20.1 (Jan. 1982), pp. 33–38. ISSN: 0001-1452, 1533-385X. DOI: 10.2514/3.51043. URL: https://arc.aiaa.org/doi/10.2514/3.51043 (visited on 06/01/2022).

[20] B E Launder and B I Sharma. "APPLICATION OF THE ENERGY-DISSIPATION MODEL OF TURBULENCE TO THE CALCULATION OF FLOW NEAR A SPINNING DISC". In: 1.2 (1974), p. 7.

[21] E. R. Van Driest. "On Turbulent Flow Near a Wall". In: *Journal of the Aeronautical Sciences* 23.11 (Nov. 1956), pp. 1007–1011. ISSN: 1936-9956. DOI: 10.2514/8.3713. URL: https://arc.aiaa.org/doi/10.2514/8.3713 (visited on 06/25/2022).

[22] F. R. Menter. "Zonal Two Equation k-[omega] Turbulence Models for Aerodynamic Flows". In: *AIAA Journal* 32.8 (July 1993), pp. 1598–1605. ISSN: 0001-1452, 1533-385X. DOI: 10.2514/3.12149. URL: https://arc.aiaa.org/doi/10.2514/3.12149 (visited on 06/01/2022).

[23] David C. Wilcox. "Formulation of the k-w Turbulence Model Revisited". In: *AIAA Journal* 46.11 (Nov. 2008), pp. 2823–2838. ISSN: 0001-1452, 1533-385X. DOI: 10.2514/1.36541. URL: https://arc.aiaa.org/doi/10.2514/1.36541 (visited on 11/14/2021).

[24] F. R. Menter. "Two-equation eddy-viscosity turbulence models for engineering applications". In: *AIAA Journal* 32.8 (Aug. 1994), pp. 1598–1605. ISSN: 0001-1452, 1533-385X. DOI: 10.2514/3.12149. URL: https://arc.aiaa.org/doi/10.2514/3.12149 (visited on 02/27/2022).

[25] Johan Kok. "Resolving the dependence on freestream values for the k-ω turbulence model". In: *AIAA journal* 38 (July 2000). DOI: 10.2514/3.14547.

[26] F R Menter, M Kuntz, and R Langtry. "Ten Years of Industrial Experience with the SST Turbulence Model". In: *Heat and Mass Transfer* (2003), p. 9.

[27] Antti Hellsten. "Some improvements in Menter's k-[omega] SST turbulence model". In: *29th AIAA, Fluid Dynamics Conference*. 29th AIAA, Fluid Dynamics Conference. Albuquerque,NM,U.S.A.: American Institute of Aeronautics and Astronautics, June 15, 1998. DOI: 10.2514/6.1998-2554. URL: https://arc.aiaa.org/doi/10.2514/6.1998-2554 (visited on 02/27/2022).

[28] Florian Menter and Thomas Esch. "ELEMENTS OF INDUSTRIAL HEAT TRANSFER PREDICTIONS". In: (2001), p. 11.

[29] Stephen B. Pope. *Turbulent flows*. Cambridge University Press, 2000.

[30] Sharath S. Girimaji. “Partially-Averaged Navier-Stokes Model for Turbulence: A Reynolds-Averaged Navier-Stokes to Direct Numerical Simulation Bridging Method”. In: *Journal of Applied Mechanics* 73.3 (May 1, 2006), pp. 413–421. ISSN: 0021-8936, 1528-9036. DOI: 10.1115/1.2151207. URL: https://asmedigitalcollection.asme.org/appliedmechanics/article/73/3/413/469970/PartiallyAveraged-NavierStokes-Model-for (visited on 11/01/2021).

[31] M. Germano. “Turbulence: the filtering approach”. In: *Journal of Fluid Mechanics* 238 (May 1992), pp. 325–336. ISSN: 0022-1120, 1469-7645. DOI: 10.1017/S0022112092001733. URL: https://www.cambridge.org/core/product/identifier/S0022112092001733/type/journal_article (visited on 11/11/2021).

[32] Arnab Chakraborty and Hv Warrior. “Study of turbulent flow past a square cylinder using partially-averaged Navier–Stokes method in OpenFOAM”. In: *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science* 234.14 (July 2020), pp. 2821–2832. ISSN: 0954-4062, 2041-2983. DOI: 10.1177/0954406220910176. URL: http://journals.sagepub.com/doi/10.1177/0954406220910176 (visited on 12/04/2021).

[33] Khlaed S. Abdol-Hamid and Sharath S. Girimaji. “A Two-Stage Procedure Toward the Efficient Implementation of PANS and Other Hybrid Turbulence Models”. In: *NASA* (Nov. 2004).

[34] *Assessment of Partially Averaged Navier Stokes (PANS) Multiscale Model in Transonic Turbulent Separated Flows*. Vol. Volume 2: Fora, Parts A and B. Fluids Engineering Division Summer Meeting. July 2007, pp. 1451–1459. DOI: 10.1115/FEDSM2007-37630. eprint: https://asmedigitalcollection.asme.org/FEDSM/proceedings-pdf/FEDSM2007/42894/1451/2669964/1451_1.pdf. URL: https://doi.org/10.1115/FEDSM2007-37630.

[35] Alaa Elmiligui et al. “Numerical Study of Flow Past a Circular Cylinder Using RANS, Hybrid RANS /LES and PANS Formulations”. In: *22nd Applied Aerodynamics Conference and Exhibit*. 22nd Applied Aerodynamics Conference and Exhibit. Providence, Rhode Island: American Institute of Aeronautics and Astronautics, Aug. 16, 2004. ISBN: 978-1-62410-025-3. DOI: 10.2514/6.2004-4959. URL: https://arc.aiaa.org/doi/10.2514/6.2004-4959 (visited on 06/02/2022).

[36] Chi-Su Song and Seung-O Park. “Numerical simulation of flow past a square cylinder using Partially-Averaged Navier–Stokes model”. In: *Journal of Wind Engineering and Industrial Aerodynamics* 97.1 (Jan. 2009), pp. 37–47. ISSN: 01676105. DOI: 10.1016/j.jweia.2008.11.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167610508001785 (visited on 11/01/2021).

[37] M. Klapwijk, T. Lloyd, and G. Vaz. “On the accuracy of partially averaged Navier–Stokes resolution estimates”. In: *International Journal of Heat and Fluid Flow* 80 (Dec. 2019), p. 108484. ISSN: 0142727X. DOI: 10.1016/j.ijheatfluidflow.2019.108484. URL: https://linkinghub.elsevier.com/retrieve/pii/S0142727X19307131 (visited on 12/07/2021).

[38] Hosein Foroutan and Savas Yavuzkurt. “A partially-averaged Navier–Stokes model for the simulation of turbulent swirling flow with vortex breakdown”. In: *International Journal of Heat and Fluid Flow* 50 (2014), pp. 402–416. ISSN: 0142-727X. DOI: https://doi.org/10.1016/j.ijheatfluidflow.2014.10.005. URL: https://www.sciencedirect.com/science/article/pii/S0142727X14001301.

[39] Abdelkader Frendi, Abhijit Tosh, and Sharath Girimaji. “Flow Past a Backward-Facing Step: Comparison of PANS, DES and URANS Results with Experiments”. In: *International Journal for Computational Methods in Engineering Science and Mechanics* 8.1 (Dec. 19, 2006), pp. 23–38. ISSN: 1550-2287, 1550-2295. DOI: 10.1080/15502280601006207. URL: http://www.tandfonline.com/doi/abs/10.1080/15502280601006207 (visited on 01/28/2022).

[40] Dasia A. Reyes, Jacob M. Cooper, and Sharath S. Girimaji. “Characterizing velocity fluctuations in partially resolved turbulence simulations”. In: *Physics of Fluids* 26.8 (Aug. 2014), p. 085106. ISSN: 1070-6631, 1089-7666. DOI: 10.1063/1.4892080. URL: http://aip.scitation.org/doi/10.1063/1.4892080 (visited on 12/04/2021).

[41] J.M. Ma et al. “A low Reynolds number variant of partially-averaged Navier-Stokes model for turbulence”. In: *International Journal of Heat and Fluid Flow* 32.3 (June 2011), pp. 652–669. ISSN: 0142727X. DOI: 10.1016/j.ijheatfluidflow.2011.02.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S0142727X11000166 (visited on 01/20/2022).

[42] Fujihiro Hamba. "Log-layer mismatch and commutation error in hybrid RANS/LES simulation of channel flow". In: *International Journal of Heat and Fluid Flow* 30.1 (Feb. 2009), pp. 20–31. ISSN: 0142727X. DOI: 10.1016/j.ijheatfluidflow.2008.10.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S0142727X08001549 (visited on 07/18/2022).

[43] F S Pereira. "Modeling and simulation of transitional Taylor-Green vortex flow with partially averaged Navier-Stokes equations". In: (May 2021), p. 33.

[44] Sunil Lakshmipathy and Sharath Girimaji. "Partially-averaged Navier-Stokes method for turbulent flows: k-w model implementation". In: *44th AIAA Aerospace Sciences Meeting and Exhibit*. 44th AIAA Aerospace Sciences Meeting and Exhibit. Reno, Nevada: American Institute of Aeronautics and Astronautics, Jan. 9, 2006. ISBN: 978-1-62410-039-0. DOI: 10.2514/6.2006-119. URL: https://arc.aiaa.org/doi/10.2514/6.2006-119 (visited on 11/01/2021).

[45] Tomáš Kořínek, Tomáš Tisovský, and Karel Fraňa. "Prediction of heat transfer from a circular cylinder in a cross-flow at a low sub-critical Reynolds number with the Partially-Averaged Navier-Stokes method". In: *International Journal of Thermal Sciences* 169 (Nov. 2021), p. 106977. ISSN: 12900729. DOI: 10.1016/j.ijthermalsci.2021.106977. URL: https://linkinghub.elsevier.com/retrieve/pii/S1290072921001435 (visited on 03/29/2022).

[46] Sharath S. Girimaji, Eunhwan Jeong, and Ravi Srinivasan. "Partially Averaged Navier-Stokes Method for Turbulence: Fixed Point Analysis and Comparison With Unsteady Partially Averaged Navier-Stokes". In: *Journal of Applied Mechanics* 73.3 (May 1, 2006), pp. 422–429. ISSN: 0021-8936, 1528-9036. DOI: 10.1115/1.2173677. URL: https://asmedigitalcollection.asme.org/appliedmechanics/article/73/3/422/469965/Partially-Averaged-NavierStokes-Method-for (visited on 11/01/2021).

[47] *TU Delft launches new series of TU Delft AI Labs*. Jan. 28, 2021. URL: https://www.leiden-delft-erasmus.nl/en/news/tu-delft-launches-new-series-of-tu-delft-ai-labs (visited on 06/05/2022).

[48] *Symposium on Model-Consistent Data-driven Turbulence Modeling*. June 2021. URL: http://turbgate.engin.umich.edu/symposium/index21.html (visited on 06/05/2022).

[49] *UMich/NASA Symposium on Advances in Turbulence Modeling*. July 2017. URL: http://turbgate.engin.umich.edu/symposium/ (visited on 06/05/2022).

[50] *2022 Symposium on Turbulence Modeling: Roadblocks, and the Potential for Machine Learning*. July 2022. URL: https://turbmodels.larc.nasa.gov/turb-prs2022.html (visited on 06/05/2022).

[51] *Data-Driven Fluid Mechanics: Combining First Principles and Machine Learning*. Feb. 2020. URL: https://www.datadrivenfluidmechanics.com/index.php (visited on 06/05/2022).

[52] *Hands on Machine Learning for Fluid Dynamics*. Feb. 2022. URL: https://www.vki.ac.be/index.php/events-ls/events/eventdetail/534/-/hands-on-machine-learning-for-fluid-dynamics (visited on 06/05/2022).

[53] *Optimization Methods for Computational Fluid Dynamics*. May 2022. URL: https://www.vki.ac.be/index.php/events-ls/events/eventdetail/527/-/online-lecture-series-optimization-methods-for-computational-fluid-dynamics (visited on 06/05/2022).

[54] Gordon E Moore. "Cramming more components onto integrated circuits". In: 38.8 (1965), p. 4.

[55] P R Spalart. "Strategies for turbulence modelling and simulations". In: (2000), p. 12.

[56] Steven L. Brunton, Bernd R. Noack, and Petros Koumoutsakos. "Machine Learning for Fluid Mechanics". In: *Annual Review of Fluid Mechanics* 52.1 (Jan. 5, 2020), pp. 477–508. ISSN: 0066-4189, 1545-4479. DOI: 10.1146/annurev-fluid-010719-060214. URL: https://www.annualreviews.org/doi/10.1146/annurev-fluid-010719-060214 (visited on 11/01/2021).

[57] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance". In: *Journal of Fluid Mechanics* 807 (Nov. 25, 2016), pp. 155–166. ISSN: 0022-1120, 1469-7645. DOI: 10.1017/jfm.2016.615. URL: https://www.cambridge.org/core/product/identifier/S0022112016006157/type/journal_article (visited on 11/01/2021).

[58] S. B. Pope. "A more general effective-viscosity hypothesis". In: *Journal of Fluid Mechanics* 72.2 (Nov. 1975), p. 331. ISSN: 0022-1120, 1469-7645. DOI: 10.1017/S0022112075003382. URL: http://www.journals.cambridge.org/abstract_S0022112075003382 (visited on 11/01/2021).

[59] Melanie Mitchell. *An Introduction to Genetic Algorithms*. en. The MIT Press, 1996. ISBN: 0−262−13316−4.

[60] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. en. The MIT Press, 1992. ISBN: 9780262111706.

[61] Cândida Ferreira. "Gene Expression Programming: A New Adaptive Algorithm for Solving Problems". In: (2001), p. 22.

[62] Jack Weatheritt and Richard Sandberg. "A novel evolutionary algorithm applied to algebraic modifications of the RANS stress–strain relationship". In: *Journal of Computational Physics* 325 (Nov. 2016), pp. 22–37. ISSN: 00219991. DOI: 10.1016/j.jcp.2016.08.015. URL: https://linkinghub.elsevier.com/retrieve/pii/S0021999116303643 (visited on 11/01/2021).

[63] Trent McConaghy. "FFX: Fast, Scalable, Deterministic Symbolic Regression Technology". In: *Genetic Programming Theory and Practice IX*. Ed. by Rick Riolo, Ekaterina Vladislavleva, and Jason H. Moore. Series Title: Genetic and Evolutionary Computation. New York, NY: Springer New York, 2011, pp. 235–260. ISBN: 978-1-4614-1769-9. DOI: 10.1007/978-1-4614-1770-5_13. URL: http://link.springer.com/10.1007/978-1-4614-1770-5_13 (visited on 11/19/2021).

[64] J. Weatheritt and R.D. Sandberg. "The development of algebraic stress models using a novel evolutionary algorithm". In: *International Journal of Heat and Fluid Flow* 68 (Dec. 2017), pp. 298–318. ISSN: 0142727X. DOI: 10.1016/j.ijheatfluidflow.2017.09.017. URL: https://linkinghub.elsevier.com/retrieve/pii/S0142727X17303223 (visited on 06/08/2022).

[65] A. K. M. F. Hussain. "Coherent structures—reality and myth". In: *Physics of Fluids* 26.10 (1983), p. 2816. ISSN: 00319171. DOI: 10.1063/1.864048. URL: https://aip.scitation.org/doi/10.1063/1.864048 (visited on 06/09/2022).

[66] Abdulla Ghani et al. "LES of longitudinal and transverse self-excited combustion instabilities in a bluff-body stabilized turbulent premixed flame". In: *Combustion and Flame* 162.11 (Nov. 2015), pp. 4075–4083. ISSN: 00102180. DOI: 10.1016/j.combustflame.2015.08.024. URL: https://linkinghub.elsevier.com/retrieve/pii/S0010218015002989 (visited on 02/09/2022).

[67] C. R. Wilke. "A Viscosity Equation for Gas Mixtures". In: *The Journal of Chemical Physics* 18.4 (Apr. 1950), pp. 517–519. ISSN: 0021-9606, 1089-7690. DOI: 10.1063/1.1747673. URL: http://aip.scitation.org/doi/10.1063/1.1747673 (visited on 06/15/2022).

[68] *Viscosity of Air, Dynamics and Kinematic*. 2000. URL: hhttps://www.engineersedge.com/physics/viscosity_of_air_dynamic_and_kinematic_14483.htm (visited on 06/15/2022).

[69] Florian R. Menter. *Improved two-equation k - [omega] turbulence models for aerodynamic flows*. National Aeronautics and Space Administration, Ames Research Center ; National Technical Information Service, distributor Moffett Field, Calif. : Springfield, Va, 1992.

A

$k_u - \omega_u$ SST PANS OpenFOAM v2112 implementation

A.1. Main .C file

```
#include "PANSkOmegaSST.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam{
namespace RASModels{

// * * * * * * * * * * * * Private Member Functions  * * * * * * * * * * * * //

template<class BasicTurbulenceModel>
tmp<volScalarField> PANSkOmegaSST<BasicTurbulenceModel>::PANSkOmegaSST::F1
(
    const volScalarField& CDkOmega
) const
{
    tmp<volScalarField> CDkOmegaPlus = max
    (
        CDkOmega,
        dimensionedScalar("1.0e-10", dimless/sqr(dimTime), 1.0e-10)
    );

    tmp<volScalarField> arg1 = min
    (
        min
        (
            max
            (
                (scalar(1)/this->betaStar_)*sqrt(kU_)/(omegaU_*this->y_),
                scalar(500)*(this->mu()/this->rho_)/(sqr(this->y_)*omegaU_)
            ),
            (4*this->alphaOmega2_*(fK_/fOmega_))*kU_
            /(CDkOmegaPlus*sqr(this->y_))
        ),
        scalar(10)
    );

    return tanh(pow4(arg1));
}

template<class BasicTurbulenceModel>
tmp<volScalarField>
PANSkOmegaSST<BasicTurbulenceModel>::PANSkOmegaSST::F2() const
{
```

```
    tmp<volScalarField> arg2 = min
    (
        max
        (
            (scalar(2)/this->betaStar_)*sqrt(kU_)/(omegaU_*this->y_),
            scalar(500)*(this->mu()/this->rho_)/(sqr(this->y_)*omegaU_)
        ),
        scalar(100)
    );

    return tanh(sqr(arg2));
}

template<class BasicTurbulenceModel>
tmp<volScalarField>
PANSkOmegaSST<BasicTurbulenceModel>::PANSkOmegaSST::F3() const
{
    tmp<volScalarField> arg3 = min
    (
        150*(this->mu()/this->rho_)/(omegaU_*sqr(this->y_)),
        scalar(10)
    );

    return 1 - tanh(pow4(arg3));
}

template<class BasicTurbulenceModel>
void PANSkOmegaSST<BasicTurbulenceModel>::correctNut
(
    const volScalarField& S2
    // const volScalarField& F2
)
{
    this->nut_ = this->a1_*kU_/max(this->a1_*omegaU_, this->b1_*F23()*sqrt(S2));

}

// * * * * * * * * * * * * Protected Member Functions  * * * * * * * * * * * //

template<class BasicTurbulenceModel>
void PANSkOmegaSST<BasicTurbulenceModel>::correctNut()
{
    // correctNut(2*magSqr(symm(fvc::grad(this->U_))), this->F23());
    correctNut(2*magSqr(symm(fvc::grad(this->U_))));
}

template<class BasicEddyViscosityModel>
tmp<volScalarField::Internal> PANSkOmegaSST<BasicEddyViscosityModel>::GbyNu
(
    const volScalarField::Internal& GbyNu0,
    const volScalarField::Internal& F2,
    const volScalarField::Internal& S2
) const
{
    return min
    (
        GbyNu0,
        (this->c1_/this->a1_)*this->betaStar_*omegaU_()
       *max(this->a1_*omegaU_(), this->b1_*F2*sqrt(S2))
    );
}

template<class BasicTurbulenceModel>
tmp<fvScalarMatrix> PANSkOmegaSST<BasicTurbulenceModel>::Qsas
(
    const volScalarField::Internal& S2,
    const volScalarField::Internal& gamma,
    const volScalarField::Internal& beta
) const
{
    return tmp<fvScalarMatrix>
```

```
    (
        new fvScalarMatrix
        (
            omegaU_,
            dimVolume*this->rho_.dimensions()*omegaU_.dimensions()/dimTime
        )
    );
}

// * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * //

template<class BasicTurbulenceModel>
PANSkOmegaSST<BasicTurbulenceModel>::PANSkOmegaSST
():

    fEpsilon_
    (
        dimensioned<scalar>::getOrAddToDict
        (
            "fEpsilon",
            this->coeffDict_,
            1.0
        )
    ),

    fK_
    (
        IOobject
        (
            IOobject::groupName("fK", alphaRhoPhi.group()),
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        this->mesh_
    ),

    fOmega_
    (
        IOobject
        (
            "fOmega",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        fEpsilon_/fK_
    ),

    kU_
    (
        IOobject
        (
            IOobject::groupName("kU", alphaRhoPhi.group()),
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        this->mesh_
    ),

    omegaU_
    (
        IOobject
        (
            IOobject::groupName("omegaU", alphaRhoPhi.group()),
            this->runTime_.timeName(),
            this->mesh_,
```

```
                IOobject::MUST_READ,
                IOobject::AUTO_WRITE
            ),
            this->mesh_
    )

// * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * //

template<class BasicTurbulenceModel>
void PANSkOmegaSST<BasicTurbulenceModel>::correct()
{
    volScalarField CDkOmega
    (
        (2*this->alphaOmega2_*(fOmega_/fK_))*
        (fvc::grad(kU_) & fvc::grad(omegaU_))/omegaU_
    );

    {
        volScalarField::Internal betaL
        (
            gamma*this->betaStar_ - (gamma *this->betaStar_/fOmega_)
            + (beta/fOmega_)
        );

        // Unresolved turbulent frequency equation
        tmp<fvScalarMatrix> omegaUEqn
        (
            fvm::ddt(alpha, rho, omegaU_)
          + fvm::div(alphaRhoPhi, omegaU_)
          - fvm::laplacian(alpha*rho*DomegaUEff(F1), omegaU_)
         ==
            alpha()*rho()*gamma*GbyNu0
          - fvm::SuSp((2.0/3.0)*alpha()*rho()*gamma*divU, omegaU_)
          - fvm::Sp(alpha()*rho()*betaL*omegaU_(), omegaU_)
          - fvm::SuSp
            (
                alpha()*rho()*(F1() - scalar(1))*CDkOmega()/omegaU_(),
                omegaU_
            )
          + Qsas(S2(), gamma, beta)
          + fvOptions(alpha, rho, omegaU_)
        );
        solve(omegaUEqn);
;     }

    // Unresolved turbulent kinetic energy equation
    tmp<fvScalarMatrix> kUEqn
    (
        fvm::ddt(alpha, rho, kU_)
      + fvm::div(alphaRhoPhi, kU_)
      - fvm::laplacian(alpha*rho*DkUEff(F1), kU_)
     ==
        alpha()*rho()*min(G, (this->c1_*this->betaStar_)*kU_()*omegaU_())
      - fvm::SuSp((2.0/3.0)*alpha()*rho()*divU, kU_)
      - fvm::Sp(alpha()*rho()*this->betaStar_*omegaU_, kU_)
      + fvOptions(alpha, rho, kU_)
    );
    solve(kUEqn);

    // Calculation of total Turbulent kinetic energy and Frequency
    this->k_ = kU_/fK_;
    this->omega_ = omegaU_/fOmega_;
}

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

} // End namespace RASModels
} // End namespace Foam
```

A.2. Header .H file

```
#ifndef PANSkOmegaSST_H
#define PANSkOmegaSST_H

#include "kOmegaSST.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam{
namespace RASModels{

/*---------------------------------------------------------------------------*\
                     Class PANSkOmegaSST Declaration
\*---------------------------------------------------------------------------*/

template<class BasicTurbulenceModel>
class PANSkOmegaSST
:

    // PANS coefficients
        dimensionedScalar fEpsilon_;
        volScalarField fK_;
        volScalarField fOmega_;

    // Fields
        volScalarField kU_;
        volScalarField omegaU_;

    //- Destructor
        virtual ~PANSkOmegaSST() = default;


    // Member Functions

        //- Return the effective diffusivity for unresolved k
        tmp<volScalarField> DkUEff(const volScalarField& F1) const
        {
            return tmp<volScalarField>
            (
                new volScalarField
                (
                    "DkUEff",
                    (fOmega_/fK_)*this->alphaK(F1)*this->nut_ + this->nu()
                )
            );
        }

        //- Return the effective diffusivity for unresolved omega
        tmp<volScalarField> DomegaUEff(const volScalarField& F1) const
        {
            return tmp<volScalarField>
            (
                new volScalarField
                (
                    "DomegaUEff",
                    (fOmega_/fK_)*this->alphaOmega(F1)*this->nut_ + this->nu()
                )
            );
        }

        //- Return the unresolved turbulence kinetic energy
        virtual tmp<volScalarField> kU() const
        {
            return kU_;
        }

        //- Return the turbulence kinetic energy dissipation rate
        virtual tmp<volScalarField> omegaU() const
        {
            return omegaU_;
```

```cpp
        }

        //- Solve the turbulence equations and correct the turbulence viscosity
        virtual void correct();

};

} // End namespace RASModels
} // End namespace Foam

#ifdef NoRepository
    #include "PANSkOmegaSST.C"

#endif
#endif
```

B

$k-$corrective frozen PANS implementation

B.1. Main .C file

```
#include "frozenInterpPANSkOmegaSST.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam{
namespace RASModels{

// * * * * * * * * * * * * Private Member Functions  * * * * * * * * * * * * //

template<class BasicTurbulenceModel>
tmp<volScalarField> frozenInterpPANSkOmegaSST<BasicTurbulenceModel>::
    frozenInterpPANSkOmegaSST::F1
(
    const volScalarField& CDkOmega
) const
{
    tmp<volScalarField> CDkOmegaPlus = max
    (
        CDkOmega,
        dimensionedScalar("1.0e-10", dimless/sqr(dimTime), 1.0e-10)
    );

    tmp<volScalarField> arg1 = min
    (
        min
        (
            max
            (
                (scalar(1)/betaStar_)*sqrt(kU_LES_)/(omegaU_*y_),
                scalar(500)*(this->mu()/this->rho_)/(sqr(y_)*omegaU_)
            ),
            (4*alphaOmega2_*(fK_/fOmega_))*kU_LES_
            /(CDkOmegaPlus*sqr(y_))
        ),
        scalar(10)
    );

    return tanh(pow4(arg1));
}

template<class BasicTurbulenceModel>
tmp<volScalarField>
frozenInterpPANSkOmegaSST<BasicTurbulenceModel>::frozenInterpPANSkOmegaSST::F2() const
```

```
{
    tmp<volScalarField> arg2 = min
    (
        max
        (
            (scalar(2)/betaStar_)*sqrt(kU_LES_)/(omegaU_*y_),
            scalar(500)*(this->mu()/this->rho_)/(sqr(y_)*omegaU_)
        ),
        scalar(100)
    );

    return tanh(sqr(arg2));
}

template<class BasicTurbulenceModel>
tmp<volScalarField>
frozenInterpPANSkOmegaSST<BasicTurbulenceModel>::frozenInterpPANSkOmegaSST::F3() const
{
    tmp<volScalarField> arg3 = min
    (
        150*(this->mu()/this->rho_)/(omegaU_*sqr(y_)),
        scalar(10)
    );

    return 1 - tanh(pow4(arg3));
}

template<class BasicTurbulenceModel>
void frozenInterpPANSkOmegaSST<BasicTurbulenceModel>::correctNut
(
    const volScalarField& S2
    // const volScalarField& F2
)
{
    this->nut_ = a1_*kU_LES_/max(a1_*omegaU_, b1_*F23()*sqrt(S2));
}

// * * * * * * * * * * * * Protected Member Functions  * * * * * * * * * * * //

template<class BasicEddyViscosityModel>
tmp<volScalarField::Internal> frozenInterpPANSkOmegaSST<BasicEddyViscosityModel>::GbyNu
(
    const volScalarField::Internal& GbyNu0,
    const volScalarField::Internal& F2,
    const volScalarField::Internal& S2
) const
{
    return min
    (
        GbyNu0,
        (c1_/a1_)*betaStar_*omegaU_()
       *max(a1_*omegaU_(), b1_*F2*sqrt(S2))
    );
}

template<class BasicTurbulenceModel>
tmp<fvScalarMatrix> frozenInterpPANSkOmegaSST<BasicTurbulenceModel>::Qsas
(
    const volScalarField::Internal& S2,
    const volScalarField::Internal& gamma,
    const volScalarField::Internal& beta
) const
{
    return tmp<fvScalarMatrix>
    (
        new fvScalarMatrix
        (
            omegaU_,
            dimVolume*this->rho_.dimensions()*omegaU_.dimensions()/dimTime
        )
    );
```

```
}

// * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * //

////////////// Custom function to aid in interpolation //////////////

class customClass
{
public:
  const float period_;
  const float timeStep_;
  std::vector<double> times_hifi_;

  customClass();
  ~customClass();

};

customClass::customClass()
:
period_(0.00825617),
timeStep_(1e-4),
times_hifi_(arange<double>(0, round_up(period_,4), round_up(timeStep_,4)))
{}

customClass::~customClass(){}

customClass myClass;

template<class BasicTurbulenceModel>
frozenInterpPANSkOmegaSST<BasicTurbulenceModel>::frozenInterpPANSkOmegaSST
():
    eddyViscosity<RASModel<BasicTurbulenceModel>>
    (),

    fEpsilon_
    (
        dimensioned<scalar>::getOrAddToDict
        (
            "fEpsilon",
            this->coeffDict_,
            1.0
        )
    ),

    fK_
    (
        IOobject
        (
            IOobject::groupName("fK", alphaRhoPhi.group()),
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::MUST_READ, //MUST_READ,
            IOobject::AUTO_WRITE
        ),
        this->mesh_
    ),

    fOmega_
    (
        IOobject
        (
            "fOmega",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::NO_WRITE
        ),
        fEpsilon_/fK_
    ),

```

```
    //========================== LES fields ==============================
    k_LES_
    (
        IOobject
        (
            IOobject::groupName("k_LES", U.group()),
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::MUST_READ,
            IOobject::NO_WRITE
        ),
        this->mesh_
    ),

    kU_LES_
    (
        IOobject
        (
            IOobject::groupName("kU_LES", U.group()),
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        k_LES_ * fK_
    ),

    tauij_LES_
    (
        IOobject
        (
            "tauij_LES",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::MUST_READ,
            IOobject::NO_WRITE
        ),
        this->mesh_
    ),
    tauijU_LES_
    (
        IOobject
        (
            "tauijU_LES",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::NO_WRITE
        ),
        tauij_LES_ * fK_
    ),
    aijU_LES_
    (
        IOobject
        (
            "aijU_LES",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::NO_WRITE
        ),
        tauijU_LES_ - ((2.0/3.0)*I)*kU_LES_
    ),
    bijU_LES_
    (
        IOobject
        (
            "bijU_LES",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
```

```
            IOobject::NO_WRITE
        ),
        aijU_LES_ / 2.0 / (kU_LES_ + this->kMin_)
    ),
    PkULES_
    (
        IOobject
        (
            "PkULES",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::NO_WRITE
        ),
        this->mesh_,
        dimensionedScalar("PkULES", dimensionSet(0,2,-3,0,0,0,0), 0.0)
    ),

    //========================== Unknown fields - MUST be written
        ============================
    omega_
    (
        IOobject
        (
            IOobject::groupName("omega", alphaRhoPhi.group()),
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        this->mesh_
    ),
    omegaU_
    (
        IOobject
        (
            IOobject::groupName("omegaU", alphaRhoPhi.group()),
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::MUST_READ,
            IOobject::AUTO_WRITE
        ),
        this->mesh_
    ),

    kUDeficit_
    (
        IOobject(
                  "kUDeficit",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        this->mesh_,
        dimensionedScalar("kUDeficit", dimensionSet(0,2,-3,0,0,0,0), 0.0)
    ),

    bijUDelta_
    (
        IOobject
        (
            "bijDelta",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        0.0*symm(fvc::grad(this->U_))/omegaU_
    ),

```

```
    //========================== Misc ===========================

    gradU_
    (
        IOobject
        (
            "gradU",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::NO_WRITE
        ),
        fvc::grad(this->U_)
    ),
    gradkU_LES_
    (
        IOobject
        (
            "gradkU_LES",
            this->runTime_.timeName(),
            this->mesh_,
            IOobject::NO_READ,
            IOobject::NO_WRITE
        ),
        fvc::grad(kU_LES_)
    ),
    gradomegaU_
    (
       IOobject
       (
           "gradomegaU",
           this->runTime_.timeName(),
           this->mesh_,
           IOobject::NO_READ,
           IOobject::NO_WRITE
       ),
       this->mesh_,
       dimensionedVector("gradomegaU", dimensionSet(0,-1,-1,0,0,0,0), Zero)
    )
{}

// * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * //

template<class BasicTurbulenceModel>
void frozenInterpPANSkOmegaSST<BasicTurbulenceModel>::correct()
{
    volScalarField& omegaU_ = this->omegaU_;
    volScalarField& k_LES_ = this->k_LES_;
    volScalarField& kU_LES_ = this->kU_LES_;

    ///////////////// RE-READ FIELDS FOR NEW TIMESTEP ///////////////////

    double currentTime_(this->runTime_.value());
    int lowerIndex_(std::get<0>(searchBounds(myClass.period_, currentTime_, myClass.
        times_hifi_)));
    int upperIndex_(std::get<1>(searchBounds(myClass.period_, currentTime_, myClass.
        times_hifi_)));
    double preTime_(myClass.times_hifi_[lowerIndex_]);
    double postTime_(myClass.times_hifi_[upperIndex_]);
    double remainTime_(fmod(currentTime_,myClass.period_) - preTime_);
    volScalarField k_LES_pre_
    (
        IOobject
        (
            IOobject::groupName("k_LES", U.group()),
            name(preTime_),
            this->mesh_,
            IOobject::MUST_READ,
            IOobject::NO_WRITE
        ),
        this->mesh_
```

```
    );

    volScalarField k_LES_post_
    (
        IOobject
        (
            IOobject::groupName("k_LES", U.group()),
            name(postTime_),
            this->mesh_,
            IOobject::MUST_READ,
            IOobject::NO_WRITE
        ),
        this->mesh_
    );

    // k_LES_ = (k_LES_post_ - k_LES_pre_) / (postTime_ - preTime_) * (currentTime_ -
        preTime_) + k_LES_pre_;
    k_LES_ = (k_LES_post_ - k_LES_pre_) / myClass.timeStep_ * remainTime_ + k_LES_pre_;
    kU_LES_ = k_LES_ * fK_;

    volSymmTensorField tauij_LES_pre_
    (
        IOobject
        (
            "tauij_LES",
            name(preTime_),
            this->mesh_,
            IOobject::MUST_READ,
            IOobject::NO_WRITE
        ),
        this->mesh_
    );

    volSymmTensorField tauij_LES_post_
    (
        IOobject
        (
            "tauij_LES",
            name(postTime_),
            this->mesh_,
            IOobject::MUST_READ,
            IOobject::NO_WRITE
        ),
        this->mesh_
    );

    tauij_LES_ =(tauij_LES_post_ - tauij_LES_pre_) / myClass.timeStep_ * remainTime_ +
        tauij_LES_pre_;
    tauijU_LES_ = tauij_LES_ * fK_;
    aijU_LES_ = tauijU_LES_ - ((2.0/3.0)*I)*kU_LES_;
    bijU_LES_ = aijU_LES_ / 2.0 / (kU_LES_ + this->kMin_);
    gradkU_LES_ = fvc::grad(kU_LES_);

    ///////////////// END OF RE-READ ////////////////////

    // Production term from HiFi dataset
    PkULES_ = -tauijU_LES_ && tgradU();

    volScalarField CDkOmega
    (
        (2*this->alphaOmega2_*(fOmega_/fK_))*
        (fvc::grad(kU_LES_) & fvc::grad(omegaU_))/omegaU_
    );

    {
        volScalarField::Internal gamma(this->gamma(F1));
        volScalarField::Internal beta(this->beta(F1));
        volScalarField::Internal betaL
        (
            gamma*betaStar_ - (gamma *betaStar_/fOmega_)
            + (beta/fOmega_)
```

```
        );

        // Unresolved Turbulent frequency equation
        tmp<fvScalarMatrix> omegaUEqn
        (
            fvm::ddt(alpha, rho, omegaU_)
          + fvm::div(alphaRhoPhi, omegaU_)
          - fvm::laplacian(alpha*rho*DomegaUEff(F1), omegaU_)
         ==
            alpha()*rho()*gamma*
            (
                   PkULES_ *omegaU_()/kU_LES_() // omega/k = 1/nut
                )
          - fvm::SuSp((2.0/3.0)*alpha()*rho()*gamma*divU, omegaU_)
          - fvm::Sp(alpha()*rho()*betaL*omegaU_(), omegaU_)
          - fvm::SuSp
            (
                alpha()*rho()*(F1() - scalar(1))*CDkOmega()/omegaU_(),
                omegaU_
            )
          + Qsas(S2(), gamma, beta)
          + fvOptions(alpha, rho, omegaU_)
        );
        solve(omegaUEqn);
;     }

    // kUDeficit_ refers to R_u term

    kUDeficit_ =  fvc::ddt(alpha, rho*kU_LES_)
                + fvc::div(alphaRhoPhi, kU_LES_)
                - fvc::laplacian(alpha*rho*DkUEff(F1), kU_LES_)
                - alpha()*rho()*PkULES_
                + (2.0/3.0)*alpha()*rho()*divU*kU_LES_ // Incompressible fluid divU = 0
                + alpha()*rho()*this->betaStar_*omegaU_*kU_LES_;

    // Calculation of Turbulent kinetic energy and Frequency
    omega_ = omegaU_/fOmega_;

    // Calculate bijUDelta, the model correction term for RST equation
    bijUDelta_ = bijU_LES_ + nut / kU_LES_ * symm(fvc::grad(this->U_));

}
} // End namespace RASModels
} // End namespace Foam
```

B.2. Header .H file

```
#ifndef frozenInterpPANSkOmegaSST_H
#define frozenInterpPANSkOmegaSST_H

#include "kOmegaSST.H"

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

namespace Foam{
namespace RASModels{

// * * * * * * * * * * * * * Custom Function(s)  * * * * * * * * * * * * //

//// Similar to numpy.arange()
template<typename T>
std::vector<T> arange(T start, T stop, T step)
{
    std::vector<T> values;
    for (T value = start; value < stop; value += step)
        values.push_back(value);
    return values;
}

```

```
//// Round double to certain decimal places
double round_up(double value, int decimal_places)
{
    const double multiplier = std::pow(10.0, decimal_places);
    return std::ceil(value * multiplier) / multiplier;
}

//// Search for the smallest time in that is larger than the instantaneous time
std::tuple<int,int> searchBounds(double per, double val, std::vector<double> vec)
{
  double remain;
  int lowerBoundIndex = 0;
  int upperBoundIndex = 0;
  remain = fmod(val, per);
  if (remain > vec.back() && remain < per)
  {
    Info << "situation B" << endl;
    lowerBoundIndex = vec.size() - 1;
    upperBoundIndex = 0;
  }
  else
  {
    for(std::size_t i = 0; i < vec.size(); ++i)
      {
        // remain = remainder(val, per); // not absolute remainder but scaled withrespect to
            the divider
        // Info << "vec.back():" << vec.back() << endl;
        if (vec[i] > remain)
        {
          lowerBoundIndex = i-1;
          upperBoundIndex = i;
          break;
        }
      }
  }
  return {lowerBoundIndex, upperBoundIndex};
}

/*---------------------------------------------------------------------------*\
                     Class PANSkOmegaSST Declaration
\*---------------------------------------------------------------------------*/

template<class BasicTurbulenceModel>
class frozenInterpPANSkOmegaSST
:

protected:

    // PANS coefficients
        dimensionedScalar fEpsilon_;
        volScalarField fK_;
        volScalarField fOmega_;

    // LES fields
        volScalarField k_LES_;
        volScalarField kU_LES_;
        volSymmTensorField tauij_LES_;
        volSymmTensorField tauijU_LES_;
        volSymmTensorField aijU_LES_;
        volSymmTensorField bijU_LES_;
        volScalarField PkULES_;

    // Fields to solve for
        volScalarField omega_;
        volScalarField omegaU_;
        volScalarField kUDeficit_;
        volSymmTensorField bijUDelta_;
        volSymmTensorField aijUDelta_;

    // Gradients
        volTensorField gradU_;
```

```
        volVectorField gradkU_LES_;
        volVectorField gradomegaU_;

    // Member Functions

        //- Return the effective diffusivity for unresolved k
        tmp<volScalarField> DkUEff(const volScalarField& F1) const
        {
            return tmp<volScalarField>
            (
                new volScalarField
                (
                    "DkUEff",
                    (fOmega_/fK_)*this->alphaK(F1)*this->nut_ + this->nu()
                )
            );
        }

        //- Return the effective diffusivity for unresolved omega
        tmp<volScalarField> DomegaUEff(const volScalarField& F1) const
        {
            return tmp<volScalarField>
            (
                new volScalarField
                (
                    "DomegaUEff",
                    (fOmega_/fK_)*this->alphaOmega(F1)*this->nut_ + this->nu()
                )
            );
        }

        //- Return the unresolved turbulence kinetic energy
        virtual tmp<volScalarField> kU_LES() const
        {
            return kU_LES_;
        }

        //- Return the turbulence kinetic energy dissipation rate
        virtual tmp<volScalarField> omegaU() const
        {
            return omegaU_;
        }
};
} // End namespace RASModels
} // End namespace Foam

#ifdef NoRepository
    #include "frozenInterpPANSkOmegaSST.C"

#endif
#endif
```

C

Frozen-PIMPLE algorithm implementation

```
#include "fvCFD.H"
#include "dynamicFvMesh.H"
#include "singlePhaseTransportModel.H"
#include "turbulentTransportModel.H"
#include "pimpleControl.H"
#include "CorrectPhi.H"
#include "fvOptions.H"
#include "localEulerDdtScheme.H"
#include "fvcSmooth.H"

// Similar to numpy.arange()
template<typename T>
std::vector<T> arange(T start, T stop, T step)
{
    std::vector<T> values;
    for (T value = start; value < stop; value += step)
        values.push_back(value);
    return values;
}

// Round double to certain decimal places
double round_up(double value, int decimal_places)
{
    const double multiplier = std::pow(10.0, decimal_places);
    return std::ceil(value * multiplier) / multiplier;
}

// Search for the smallest time in that is larger than the instantaneous time
std::tuple<int,int> searchBounds(double per, double val, std::vector<double> vec)
{
  double remain;
  int lowerBoundIndex = 0;
  int upperBoundIndex = 0;
  remain = fmod(val, per);
  if (remain > vec.back() && remain < per)
  {
    Info << "situation B" << endl;
    lowerBoundIndex = vec.size() - 1;
    upperBoundIndex = 0;
  }
  else
  {
    for(std::size_t i = 0; i < vec.size(); ++i)
      {
        // remain = remainder(val, per); // not absolute remainder but scaled withrespect to
            the divider
```

```
        // Info << "vec.back():" << vec.back() << endl;
        if (vec[i] > remain)
        {
          lowerBoundIndex = i-1;
          upperBoundIndex = i;
          break;
        }
      }
  }
  return {lowerBoundIndex, upperBoundIndex};
}

class customClass
{
public:
  const float period_;
  const float timeStep_;
  double currentTime_;
  int lowerIndex_;
  int upperIndex_;
  double preTime_;
  double postTime_;
  double remainTime_;

  customClass();
  ~customClass();
};

customClass::customClass()
:
period_(0.00825617),
timeStep_(5e-5),
currentTime_(0.0),
lowerIndex_(0),
upperIndex_(0),
remainTime_(0.0)
{}
customClass::~customClass()
{}

customClass myClass;
auto times_hifi = arange<double>(0, round_up(myClass.period_,5), round_up(myClass.timeStep_
    ,5));

// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

int main(int argc, char *argv[])
{
    // Initialise the preTime and postTime
    myClass.preTime_ = times_hifi[0];
    myClass.postTime_ = times_hifi[1];

    // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

    while (runTime.run())
    {
        #include "readDyMControls.H"

        if (LTS)
        {
            #include "setRDeltaT.H"
        }
        else
        {
            #include "CourantNo.H"
            #include "setDeltaT.H"
        }

        ++runTime;

        myClass.currentTime_ = runTime.value();
```

```
        myClass.lowerIndex_ = std::get<0>(searchBounds(myClass.period_, myClass.currentTime_,
             times_hifi));
        myClass.upperIndex_ = std::get<1>(searchBounds(myClass.period_, myClass.currentTime_,
             times_hifi));
        myClass.preTime_ = times_hifi[myClass.lowerIndex_];
        myClass.postTime_ = times_hifi[myClass.lowerIndex_+1];
        myClass.remainTime_ = fmod(myClass.currentTime_,myClass.period_) - myClass.preTime_;

        // --- Pressure-velocity PIMPLE corrector loop
        while (pimple.loop())
        {
            if (pimple.firstIter() || moveMeshOuterCorrectors)
            {
                mesh.controlledUpdate();

                if (mesh.changing())
                {
                    MRF.update();

                    if (correctPhi)
                    {
                        // Calculate absolute flux
                        // from the mapped surface velocity
                        phi = mesh.Sf() & Uf();

                        #include "correctPhi.H"

                        // Make the flux relative to the mesh motion
                        fvc::makeRelative(phi, U);
                    }

                    if (checkMeshCourantNo)
                    {
                        #include "meshCourantNo.H"
                    }
                }
            }

            volVectorField U_LES_pre
            (
                IOobject
                (
                    "U_LES",
                    name(myClass.preTime_),
                    mesh,
                    IOobject::MUST_READ,
                    IOobject::NO_WRITE
                ),
                mesh
            );

            volVectorField U_LES_post
            (
                IOobject
                (
                    "U_LES",
                    name(myClass.postTime_),
                    mesh,
                    IOobject::MUST_READ,
                    IOobject::NO_WRITE
                ),
                mesh
            );

            U_LES = (U_LES_post - U_LES_pre) / myClass.timeStep_ * myClass.remainTime_ +
                U_LES_pre;
        }
    }
}
```