Machine Learning-Augmented Turbulence Models for the Simulation of Two-Phase Shear Flows

by

Luc Bertolotti

Thesis Supervisors | Dr. Richard Jefferson-Loveday
Dr. Stephen Ambrose

The Gas Turbine and Transmissions Research Centre
Department of Mechanical, Materials and Manufacturing Engineering
The University of Nottingham

Dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mechanical Engineering

August 2022

To Mémé, who guided and encouraged me to look for answers.

"I am an old man now, and when I die and go to Heaven there are two matters on which I hope enlightenment. One is quantum electrodynamics and the other is turbulence of fluids. About the former, I am really rather optimistic."

Sir Horace Lamb

Acknowledgements

Firstly, I would like to express my gratitude to my supervisors Dr. Richard Jefferson-Loveday and Dr. Stephen Ambrose who gave me the opportunity to carry out this PhD project. I would like to thank them for their constant support, guidance, and precious advice throughout the past few years.

I am also thankful to Prof. Carol Eastwick, head of G2TRC, for her ongoing help and support anytime we needed them; Dr. Evgenia Korsukova for her suggestions and collaboration; Dr. Jung Hoon Kim, Dr. David Hann, and Ahmed Abou Sherif for their experimental work, it has been a pleasure working with them on the Cornerstone project.

Thanks go to Rolls-Royce plc and the EPSRC for their support under the Prosperity Partnership Grant Cornerstone[1].

I am grateful to Dr. Mirco Magnini and Dr. Federico Municchi for their invaluable training, advice, and tips on OpenFOAM.

I would like to thank Dr. Colin Bannister, manager of the University of Nottingham's HPC service, which was used to perform most of my CFD calculations.

A special thanks goes to Benedikt, Gabriele, Ryan, William, Zak, and to all my lab mates, for their support beyond friendship and the interesting conversations; to my former classmate, then roommate, but most importantly valuable friend Dr Philippe Sohouenou, and our all-nighters working on our theses or debating on existential issues; to my dear friends, Benjo, Guillaume, Martin, and Théo, who were always happy to discuss about the math, and helped maintain my (online) social life alive especially during the many lockdowns.

Finally, my thanks go to my parents, who have always encouraged me to pursue my dreams and been giving me the tools to achieve my projects; and to my Juliette, for her love, her kindness, her patience, for so many years now.

[1]Mechanical Engineering Science to Enable Aero Propulsion Futures", Grant Ref: EP/R004951/1.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and the acknowledgements.

Luc Bertolotti

September 2022

Abstract

This dissertation focuses on the investigation of co-current two-phase stratified gas-liquid shearing flows with a sharp interface for industrial applications and more specifically for the study of flows present in aero-engine bearing chambers. Predicting the behaviour of shear flows in the different parts of an aero-engine, such as bearing chambers, is crucial. As a matter of fact, in the context of the thermal management of the bearing chamber, methodologies to predict the oil film thickness distribution must be investigated in order to optimise the system lubrication and prevent from oil coking or degradation. The thickness distribution of wavy films is largely analysed in the industry, but turbulent two-phase flows remain very challenging to predict depending on the case. Carrying out experiments is often too expensive and computational fluid dynamics (CFD) still struggles with two-phase flow prediction especially when it involves the modelling of a sharp interface. Many CFD methods are employed to predict the oil film thickness distribution and interface velocity. However, the vastly used standard Reynolds-averaged Navier-Stokes equations (RANS) turbulence models are derived from semi-empirical methods of turbulence damping, which are inaccurate for wavy films with high gradients of velocity across the interface, thus impacting flow modelling in bearing chambers.

With the objective of improving the current RANS models from scale-resolving methods, high-fidelity simulations were carried out in this dissertation,

using quasi-direct numerical simulation (qDNS) with OpenFOAM. The results of various high-fidelity simulation cases were employed to inform the interfacial turbulence in the widely used standard Wilcox's RANS $k-\omega$ turbulence model. Two flow configurations based on experimental works exploring stratified flows in horizontal channels were investigated, namely the thick-film and thin-film configurations. Simplified 3D periodic versions of the channels were designed. Channels were filled with two distinct gaseous and liquid phases. Shearing flows and interface waviness were triggered by the strong interfacial velocity gradients, as the gaseous phase velocity was set at much larger values than the liquid velocity. Numerical results were compared with experiments in terms of mean velocity and turbulent energy and Reynolds stress.

First, a preliminary study was conducted in order to choose the most optimal methods for the investigated two-phase flows. Part of this preliminary work involved the domain simplification by an analysis of the fluctuation velocity auto-correlations in the periodic directions of the computational domain. Secondly, a methodology of a proof of concept was established and performed in order to demonstrate the RANS $k-\omega$ turbulence model capability to be driven by high-fidelity data for the prediction of accurate behaviours of two-phase shearing flows, in the thick-film configuration. Finally, a high-fidelity data-driven neural network was implemented in machine learning models with PyTorch and coupled with the Wilcox's model through python scripts in order to inform and improve the interfacial turbulence of RANS simulations. This was accomplished by following the methodology of the proof of concept on new cases, in the thin film configuration. While the type of corrections predicted by the implemented machine learning models presented in this dissertation applies to the specific Wilcox's turbulence model, it was proven that machine learning

can be used effectively to assist and enhance CFD abilities to predict two-phase stratified flows with averaged models without increasing computational costs significantly as opposite to high-fidelity simulations or hybrid models. In the context of industrial studies, one can imagine the future development of new CFD methods involving couplings between averaged turbulence models and machine learning models for quick analyses and accurate results.

Keywords: Aero-engine, Bearing chamber, Two-phase flow, Computational Fluid Dynamics, Quasi-Direct Numerical Simulation, Interfacial Turbulence, Neureal Network, Machine Learning.

Table of contents

List of figures **xvii**

List of tables **xxvii**

Nomenclature **xxix**

1 Introduction **1**

1.1 Context of the research 1

1.2 Problem statement 5

1.3 Contribution to knowledge and objectives 6

1.4 Thesis layout 10

2 Background knowledge on two-phase flows and numerical methods **13**

2.1 General knowledge on two-phase flows 13

2.2 Numerical modelling of two-phase flows 15

2.2.1 The Volume of Fluid method 19

2.2.2 The Euler-Euler method 23

2.3 Additional numerical methods for CFD simulations 26

2.3.1 OpenFOAM 26

2.3.2 Finite volume spatial discretisation method 27

2.3.3 Discretisation schemes 28

2.3.4 Turbulence modelling 31

2.3.5 Solution methods 43

2.4 Introduction to machine learning 49
2.4.1 Machine learning process and neural network 49
2.4.2 Data normalisation 52
2.4.3 Loss function and training quality 54
2.5 Remarks 58

3 Literature review: two-phase shear flows and ML applications in CFD 59

3.1 Existing work on two-phase shear-flows 59
3.1.1 Experimental work 60
3.1.2 Numerical work 69
3.2 Existing applications of machine learning in CFD 78
3.2.1 ML applications in single-phase flows 80
3.2.2 ML applications in two-phase flows 84
3.2.3 Concluding remarks 84

4 High-fidelity simulation of a stratified flow in a channel 87

4.1 Preliminary study: comparison of the VOF and Euler-Euler methods 87
4.1.1 Simulation setup and methodology 88
4.1.2 Numerical methods 92
4.1.3 Results and comparison between the VOF and Euler-Euler methods 96
4.2 Further analyses using the VOF method 103
4.2.1 Domain geometry 104
4.2.2 Quasi-DNS and small turbulent scales 104
4.2.3 Mesh resolution 106
4.2.4 Results 107

5 Proof of concept: implementation of a data-driven turbulence model 123

5.1 Informing the $k-\omega$ model with high-fidelity data 124
5.1.1 Performance of the standard $k-\omega$ model in two-phase shear flows 124
5.1.2 Methodology 125
5.2 Data-driven thick-film flow turbulence modelling 131
5.2.1 Application to the smooth interface flow regime 132
5.2.2 Limitations of the frozen film correction: application to wavy-films 137
5.3 Turbulence correction prediction by a machine learning model . 140
5.3.1 Structure of the ML model 141
5.3.2 Model training 142
5.3.3 ML model predictions 145
5.4 Conclusions and discussions 146

6 Application to thin-film two-phase channel flows 151
6.1 Simulation setup and creation of the training dataset 152
6.1.1 Geometry and flow characteristics 152
6.1.2 Quasi-DNS simulations and comparison with experiments 157
6.2 Frozen correction field method 164
6.2.1 Process description 164
6.2.2 Implementation of the machine learning model M2 . . . 166
6.2.3 Model M2 prediction results and discussions 170
6.3 Adaptive correction field method 175
6.3.1 Process description 175
6.3.2 Implementation of a new machine learning model: model M3 176
6.3.3 Model M3 prediction results and discussions 181
6.3.4 Test of model M3 in an open channel 188
6.4 Conclusions and discussions 190

7 Conclusions and future work **193**

7.1 Conclusions 193

7.1.1 Summary of the work 193

7.1.2 Key contributions 199

7.2 Recommendations for future improvements 200

List of publications **207**

References **209**

Appendix A Code implementations **229**

A.1 C++ pieces of code 229

A.1.1 Field objects implemented in the solver `interFoam` . . . 229

A.1.2 Function objects coded in the OpenFOAM simulation control dictionary `controlDict` 229

A.1.3 Coded source terms in the `fvOptions` dictionary 235

A.1.4 Modified turbulence model 237

A.2 Python scripts 245

A.3 PyTorch model notebooks 248

Appendix B Additional figures **259**

B.1 Additional figures from chapter 5 259

B.2 Additional figures from chapter 6 261

List of figures

1.1 Rolls-Royce Ultrafan aero-engine [109] and rear bearing chamber location 4
1.2 Schematic of a bearing chamber cross section 4
1.3 Schematic of a zoomed view of a bearing chamber cross-section 6

2.1 Schematics of different gas/liquid two-phase flow configurations 15
2.2 Illustration of the volume of fluid method applied to a stratified flow with a sharp interface 20
2.3 Examples of a 2D co-located grid (left) and of a 2D staggered grid (right) 28
2.4 Convective flux F across a control volume of volume centroid P and face centroid f 30
2.5 Classification of the main turbulence simulation methods. Adapted from [114] 33
2.6 Standard law of the wall with sublayers ranges 40
2.7 Flowchart of the PIMPLE algorithm process 44
2.8 Representation of a feed-forward neural network containing three inputs, two hidden layers of six and five neurons and two outputs. 52

3.1 Main gas-liquid co-current flow configurations in channels with liquid in red and gas in white 61
3.2 General scheme of the experiments of Fabre et al. [35] 65
3.3 Mean axial velocity profiles measured in [35] 65

3.4 General scheme of the experiments of Kim et al. [76] 66
3.5 Image processing procedure for the detection of the interfaces developed by Hann et al. [54] and performed in [76] 67
3.6 Kim et al. [76] raw PIV image of the air flow field (top left), velocity vector field obtained using the adaptive PIV algorithm (bottom left), corresponding mask (centre left), averaged velocity profiles for different flow regimes probed vertically in the channel (top right), and instantaneous velocity profile (bottom right). Superficial velocity: 3.6 m/s (air), 0.019 m/s (water). 68
3.7 Oil film thickness distribution in an aero-engine's bearing chamber by Bristot et al. [15] for $B = 100$ (left) and $B = 10$ (right) . 71
3.8 Mixture velocity profiles of the corrected and standard RANS models obtained by Frederix et al. [41] compared with Fabre et al. experiments [35] 74
3.9 Mean streamwise velocity profile above the interface obtained by Fulgosi et al. [43] 76
3.10 Example of a comparison obtained replacing the source term in the SA model within the boundary layer of the NACA 0012 airfoil at 8 degrees using training data from the flat plate solutions [135] 81
3.11 Difference in the true Spalart-Allmaras source term and the machine learning prediction in a global view (left) and a zoomed view on the tail (right) [135] 81
3.12 Skin friction prediction for a case example of a flow over a bump. The magenta lines represent predictions using an ensemble of machine-learned models trained on different combinations of the inverse solutions. LES is represented in blue and the standard Wilkox's $k - \omega$ in green. [118] 83

4.1 2D schematic of the simplified channel with boundary conditions (side view) 89

4.2 Turbulent kinetic energy spectrum measured in the gaseous phase 94
4.3 Mean and instantaneous velocity profiles obtained with the two multiphase methods 97
4.4 Mean axial velocity profiles 98
4.5 Axial velocity profiles 99
4.6 Turbulent kinetic energy profiles 100
4.7 Contours (x, z) of vorticity magnitude with the VOF (top) and Euler-Euler method (bottom) on a plane located above the interface at $y = 0.040$ m 101
4.8 Contours (x, z) of vorticity magnitude with the VOF (top) and Euler-Euler method (bottom) on a plane located above the interface at $y = 0.045$ m 101
4.9 3D isosurfaces of Q for $Q > 4\ 500$ with contours of vorticity magnitude with the VOF method (top) and with the Euler-Euler model (bottom) 102
4.10 Schematic of the simplified domain's geometry and boundary conditions 105
4.11 Root mean square error of the mean axial velocity (top left), turbulent kinetic energy (top right), and Reynolds stress (bottom) between Fabre et al. experiments [35] and the VOF simulations, against the mesh density 109
4.12 Mesh convergence analysis for the mean axial velocity (top left), turbulent kinetic energy (top right), and Reynolds stress (bottom) 110
4.13 Mean axial velocity (top left), TKE (top right), shear stress in the liquid (bottom left) and in the gas (bottom right) 111
4.14 Mesh representation corresponding to the refinement level s1 . 112
4.15 Spatial autocorrelations of the x, y and z-components of the fluctuation velocity measured in the centre of the gaseous phase in the flow direction (top) and in cross-flow direction (bottom) . 114

4.16 Autocorrelation contour map $R_{xx}(\Delta x, \Delta t)$ of the axial fluctuation velocity in the gaseous phase measured in the streamwise direction for $50\Delta t = 0.05$ s. 115
4.17 Energy spectrum measured in the gaseous phase in qDNS compared with the LES measurements 116
4.18 Isosurfaces of Q-criterion and contours of vorticity magnitude on a 3D view of the cyclic channel using the refinement level s1. The Q-criterion threshold is 100 times higher in the gaseous phase than the liquid phase 117
4.19 Contours of vorticity magnitude and liquid phase fraction on a 3D view of the cyclic channel with the refinement level s1 . . . 119
4.20 Contours of vorticity magnitude on a cross-flow plane (y, z) duplicated 4 times for a total width of 0.1 m (left) and contours of vorticity magnitude with velocity vector field on a single cyclic domain in the cross-stream direction with the refinement level s1 120

5.1 Mean axial velocity (left), TKE (centre), and absolute shear stress (right) profiles comparison between Fabre et al. experiments [35], qDNS predictions with mesh s1, and standard RANS $k-\omega$ model predictions 125
5.2 Specific turbulence dissipation rate (left), turbulence dissipation (centre), and TKE (right) profiles comparison between qDNS predictions with mesh s1, and standard RANS $k-\omega$ model predictions 129
5.3 Correction source term S_ω calculated and time-averaged in qDNS (black cross) and with additional spatial averaging, mapped on the RANS mesh (green line) 130

5.4 Smooth interface regime (left column) compared with the two-dimensional wavy interface regime (right column) in terms of liquid volume fraction (top row), instantaneous axial velocity in $m \cdot s^{-1}$ (centre row), and vorticity magnitude in s^{-1} (bottom row) 133
5.5 Specific turbulence dissipation budget across the interface 134
5.6 Mean axial velocity (left), TKE (centre), and cross Reynolds stress profiles comparison between the qDNS, the standard $k-\omega$ model (stand.), and the corrected $k-\omega$ model (corr.) predictions in the closed channel configuration and smooth interface regime 136
5.7 Mean axial velocity (left), TKE (centre), and cross Reynolds stress profiles comparison between the qDNS, the standard $k-\omega$ model (stand.), and the corrected $k-\omega$ model (corr.) predictions in the open channel configuration and smooth interface regime . 137
5.8 Mean axial velocity (left), TKE (centre), and cross Reynolds stress profiles comparison between the qDNS, the standard $k-\omega$ model (stand.), and the corrected $k-\omega$ model (corr.) predictions in the closed channel configuration and wavy interface regime . 138
5.9 Mean axial velocity (left), TKE (centre), and cross Reynolds stress profiles comparison between the qDNS, the standard $k-\omega$ model (stand.), and the corrected $k-\omega$ model (corr.) predictions in the open channel configuration and wavy interface regime . . 139
5.10 Adjustment of the correction field S_ω according to the interface position. S_ω profile is represented in green, the liquid film is represented in light red 140
5.11 Histogram of the inputs and output used in the training of the model M1 143
5.12 Training and validation loss (left) and accuracy (right) against the number of epochs obtained during the training of the ML model M1 144

5.13 Profile predictions obtained with the ML-informed $k-\omega$ model (blue line), the former corrected $k-\omega$ model (green dash line), the standard $k-\omega$ model (orange point line) and the qDNS (black line) in the closed channel configuration, smooth interface regime 145

5.14 Mean axial velocity (left), TKE (centre), and cross Reynolds stress profiles predictions compared between the qDNS, the standard $k-\omega$ model (stand.), the corrected $k-\omega$ model (corr.) and the corrected $k-\omega$ model without correction in the liquid denoted as "New", in the closed channel configuration and wavy interface regime 147

6.1 Map of flow patterns identified by Andritsos and Hanratty [5]: smooth interface (smooth), 2D waves (2D), and large amplitude waves (LA), overlaid with the experimental conditions of Hann and Kim [54][76] (marked symbols) identifying wave patterns: stratified smooth (SS), 2D small amplitude (2D), and 3D small amplitude(3D) 153

6.2 Spatial autocorrelations of the x, y and z-components of the fluctuation velocity measured in the centre of the gaseous phase in the flow direction (top) and in cross-flow direction (bottom) . 155

6.3 Range of mean axial velocity and correction source term profiles obtained in qDNS in cases "1." ($U_{b,l} = 0.008$ m/s), "2." ($U_{b,l} = 0.019$ m/s), and "3." ($U_{b,l} = 0.031$ m/s) 158

6.4 Q-criterion isosurfaces with axial velocity contours & vorticity magnitude contour map obtained in qDNS, case 1.e 158

6.5 qDNS simulations performed at the liquid film velocity 0.008 m/s 160

6.6 Mean axial velocity profiles obtained in qDNS and compared with the experiments in case 1.e (left), 2.c (centre), and 3.d (right) 162

6.7 Absolute value of the Reynolds stress profiles obtained in qDNS and compared with the experiments in case 1.e (left), 2.c (centre), and 3.d (right) . . . 162
6.8 Mean axial fluctuation velocity profiles obtained in qDNS and compared with the experiments in case 1.e (left), 2.c (centre), and 3.d (right) . . . 163
6.9 Mean vertical fluctuation velocity profiles obtained in qDNS and compared with the experiments in case 1.e (left), 2.c (centre), and 3.d (right) . . . 163
6.10 Flowchart of the process of a RANS simulation carried out with the M2-informed $k-\omega$ turbulence model and the frozen correction field method . . . 165
6.11 Histogram of the inputs and output used in the training of the ML model M2 . . . 167
6.12 Training and validation loss (left) and accuracy (right) against the number of epoches obtained during the training of the ML model M2 . . . 170
6.13 Predictions of model M2 in test cases 1.e (left), 2.c (centre), and 3.d (right) . . . 171
6.14 Comparison of the mean axial velocity profiles obtained in qDNS, in RANS using the M2-informed $k-\omega$ model, and in RANS using the standard $k-\omega$ model in cases 1.e (left), 2.c (centre), and 3.d (right) . . . 172
6.15 Comparison of the absolute values of Reynolds stress profiles obtained in qDNS, in RANS using the M2-informed $k-\omega$ model, and in RANS using the standard $k-\omega$ model in cases 1.e (left), 2.c (centre), and 3.d (right) . . . 173

6.16 Comparison of the TKE profiles obtained in qDNS, in RANS using the M2-informed $k-\omega$ model, and in RANS using the standard $k-\omega$ model in cases 1.e (left), 2.c (centre), and 3.d (right) 173
6.17 Flowchart of the process of a RANS simulation carried out with the M3-informed $k-\omega$ turbulence model and the adaptive correction field method 176
6.18 Histogram of the inputs and output used in the training of the ML model M3 178
6.19 Illustration of the qDNS dataset in terms of mean axial velocity (m/s) and log10 correction source term (s^{-2}) including the test cases 1.e, 2.c, and 3.d highlighted in red 179
6.20 Training and validation loss (left) and accuracy (right) against the number of epochs obtained during the training of the ML model M3 181
6.21 Predictions of model M3 in test cases 1.e (left), 2.c (centre), and 3.d (right) 182
6.22 Evolution of the corrected axial velocity (first row) and predicted (second row) source using the M3-informed $k-\omega$ model 184
6.23 Comparison of the mean axial velocity profiles obtained in qDNS, in RANS using the M3-informed $k-\omega$ model, and in RANS using the standard $k-\omega$ model in cases 1.e (left), 2.c (centre), and 3.d (right) 185
6.24 Comparison of the absolute value of Reynolds stress profiles obtained in qDNS, in RANS using the M3-informed $k-\omega$ model, and in RANS using the standard $k-\omega$ model in cases 1.e (left), 2.c (centre), and 3.d (right) 186

6.25 Comparison of the TKE profiles obtained in qDNS, in RANS using the M3-informed $k-\omega$ model, and in RANS using the standard $k-\omega$ model in cases 1.e (left), 2.c (centre), and 3.d (right) 186
6.26 Comparison of the Reynolds stress (left) and TKE (right) profiles obtained in qDNS, in RANS with the M3-informed $k-\omega$ model and the standard $k-\omega$ model, for case 1.e using linear scales . . 188
6.27 Comparison of the mean axial velocity (left), absolute value of Reynolds stress (centre), and TKE (right) profiles obtained in qDNS and in RANS using the M3-informed $k-\omega$ model and the standard $k-\omega$ model cases 3.d with a slip BC at the top wall 189

7.1 Two-phase flow in periodic section of bearing chamber with boundary conditions (BC) 204

B.1 Specific turbulence dissipation rate (left) and turbulence dissipation rate (right) profiles comparison between the qDNS, the standard $k-\omega$ model (stand.), and the corrected $k-\omega$ model (corr.) predictions in the closed channel configuration and wavy interface regime 259
B.2 profiles comparison between the qDNS, the standard $k-\omega$ model (stand.), and the corrected $k-\omega$ model (corr.) predictions in the open channel configuration and smooth interface regime . . 260
B.3 Specific turbulence dissipation rate (left) and turbulence dissipation rate (right) profiles comparison between the qDNS, the standard $k-\omega$ model (stand.), and the corrected $k-\omega$ model (corr.) predictions in the closed channel configuration and smooth interface regime 260
B.4 profiles comparison between the qDNS, the standard $k-\omega$ model (stand.), and the corrected $k-\omega$ model (corr.) predictions in the open channel configuration and wavy interface regime 261

B.5 Mean axial velocity profiles for $U_{b,g} = 3.1$ m/s and using $U_{b,l} =$ 0.008 m/s (left), $U_{b,l} = 0.019$ m/s (centre), and $U_{b,l} = 0.031$ m/s (right) 261
B.6 Mean axial velocity profiles for $U_{b,g} = 3.6$ m/s and using $U_{b,l} =$ 0.008 m/s (left), $U_{b,l} = 0.019$ m/s (centre), and $U_{b,l} = 0.031$ m/s (right) 262
B.7 Mean axial velocity profiles for $U_{b,g} = 4.2$ m/s and using $U_{b,l} =$ 0.008 m/s (left), $U_{b,l} = 0.019$ m/s (centre), and $U_{b,l} = 0.031$ m/s (right) 262
B.8 Mean axial velocity profiles for $U_{b,g} = 4.7$ m/s and using $U_{b,l} =$ 0.008 m/s (left), $U_{b,l} = 0.019$ m/s (centre), and $U_{b,l} = 0.031$ m/s (right) 263
B.9 Mean axial velocity profiles for $U_{b,g} = 5.2$ m/s and using $U_{b,l} =$ 0.008 m/s (left), $U_{b,l} = 0.019$ m/s (centre), and $U_{b,l} = 0.031$ m/s (right) 263
B.10 qDNS simulations performed at the liquid film velocity 0.019 m/s 264
B.11 qDNS simulations performed at the liquid film velocity 0.031 m/s 265

List of tables

4.1 Case description based on Fabre et al. [35] 90
4.2 Characteristics of the six meshes 108

6.1 Flow regimes description and names 154
6.2 Description of the flows conditions of the 15 qDNS cases 156
6.3 M2 training data statistics before standardisation 168
6.4 Standardised data statistics used for the training of model M2 . 168
6.5 RMSEs of the standard ("Stand.") and the ML-informed ("M2") models 174
6.6 M3 training data statistics before standardisation 180
6.7 Standardised data statistics used for the training of model M3 . 180
6.8 Simulation times of the M3-informed model 183
6.9 RMSEs of the standard ("Stand.") and M3-informed ("M3") models and evolution ("Evol.") of RMSEs in comparison with results of M3 187
6.10 RMSEs of the M3-informed model ("M3") compared to the standard model ("Stand.") in the open channel configuration . . 190

Nomenclature

Roman Symbols

A_0 Amplitude of the quantity A

$\mathbf{A}$ Vectorial field of the quantity A

A' Fluctuation of the quantity A

$\overline{A}$ Time average of the quantity A

A_x, A_y, A_z Resp. X, Y, Z components of the vector $\mathbf{A}$

B Turbulence damping factor

C_α Artificial compression coefficient

C_D Drag coefficient

D_h Hydraulic diameter (m)

F_D Drag force (N)

F_S Surface tension force (N)

H Channel height (m)

h Interface height (m)

k Turbulent kinetic energy ($\mathrm{m}^2 \cdot \mathrm{s}^{-2}$)

p	Pressure (Pa)
Re	Reynolds number
Re_h	Reynolds number based on the hydraulic diameter
S	Rate of strain tensor (s^{-1})
$\mathbf{S}$	Symmetric component of $\nabla\mathbf{U}$ (s^{-1})
S_τ	Strouhal number
T	Time period (s)
t	Time (s)
T_c	Turnover time (s)
T_comp	Simulation duration (s)
U	Velocity ($m{\cdot}s^{-1}$)
U_b	Bulk velocity ($m{\cdot}s^{-1}$)
u_c	Compression velocity ($m{\cdot}s^{-1}$)
u^*	Shear velocity ($m{\cdot}s^{-1}$)

Greek Symbols

α	Phase volume fraction
μ	Dynamic viscosity (Pl)
μ_t	Turbulent dynamic viscosity
ν	Kinematic viscosity ($m^2 \cdot s^{-1}$)

ν_t	Turbulent kinematic viscosity ($m^2 \cdot s^{-1}$)
$\mathbf{\Omega}$	Antisymmetric component of $\nabla \mathbf{U}$ (s^{-1})
ω	Specific turbulence dissipation rate (s^{-1})
Ω	Vorticity magnitude (s^{-1})
ρ	Fluid density ($kg{\cdot}m^{-3}$)
$\boldsymbol{\tau}$"	Reynolds stress tensor ($m^2 \cdot s^{-2}$)
τ_{ij}	Reynolds stress tensor coordinates ($m^2 \cdot s^{-2}$)
τ_w	Wall shear stress (s^{-1})
ε	Turbulence dissipation rate ($m^2{\cdot}s^{-3}$)

Superscripts

j	Superscript index

Subscripts

g	Gas
i	Subscript index
i	Phase number
i	Liquid
b	Bulk
s	Superficial
w	Wall

Acronyms / Abbreviations

BC Boundary Condition

BSL Baseline Model

CFD Computational Fluid Dynamics

CNN Convolution Neural Network

CV Control Volume

DNS Direct Numerical Simulation

FEM Finite Element Method

FFNN Feed-Forward Neural Network

FVM Finite Volume Method

G2TRC Gas Turbine and Transmissions Research Centre

GNU GNU's Not Unix operating system

GPL General Public License

GPU Graphics Processing Unit

HPC High Performance Computing

IATA The International Air Transport Association

LDA Laser Doppler Anemometry

LES Large Eddy Simulation

ML Machine Learning

MLP Multi-Layer Perceptron (neural network)

NN Neural Network

OpenFOAM Open Field Operation and Manipulation

PCG Preconditioned Conjugate Gradient

PIC Particle-In-Cell

PINN Physics-Informed Neural Network

PISO Pressure-Implicit with Operators Splitting

PIV Particle Image Velocimetry

QDNS Quasi-Direct Numerical Simulation

RANS Reynolds Averaged Navier-Stokes equations

SIMPLE Semi-Implicit Method for Pressure Linked Equations

SPH Smoothed-Particle Hydrodynamics

SST Shear Stress Transport model

TD Turbulence Damping

TKE Turbulent Kinetic Energy

TVD Total Variation Diminishing

URANS Unsteady Reynolds Averaged Navier-Stokes equations

VMS Variational Multiscale approach

VOF Volume of Fluid

Chapter 1

Introduction

1.1 Context of the research

In the last decades, the interest for flow simulation has grown along with the computational power and resources available. It has become common in fluid mechanics to use Computational Fluid Dynamics (CFD) as a tool to reproduce the physical behaviours of fluids. Those behaviours are obtained by either directly solving the Navier-Stokes equations and/or modelling the flow. On the one hand, high-fidelity simulations, such as direct numerical simulations (DNS), are solving the Navier-Stokes equations. On the other hand, lower fidelity simulations can be performed using flow models, such as the commonly used Reynolds Average Navier-Stokes equations (RANS) models. Those lower fidelity simulations are computationally less expensive than high fidelity simulations and are widely employed in the industry to help with creating and improving engineering designs and processes. On the contrary, high-fidelity simulations allow for a better understanding of physical behaviours as accurate solutions can be obtained for numerous cases that would not be easily set up experimentally. Moreover, CFD provides detailed data for entire studied

domains. In this thesis, it was aimed to use high-fidelity simulations results to inform and improve lower fidelity simulation models, and more particularly a RANS turbulence model: the $k-\omega$ model. One way to inform those models is to train with high fidelity data a machine learning model capable of providing appropriate corrections for the RANS model.

CFD models are designed to fit known behaviours of the flow. They are based on the Navier-Stokes equations that describe the flow motion and the parameters within are tuned by using empirical data. High-fidelity simulations try to reproduce reality with the most accuracy and depend on how well the equations describing flow motion are implemented, discretised, and how appropriate they are for this purpose. For high-fidelity simulations, increasing the number of discrete volumes in the domain normally leads to more accurate results as it helps capturing and solving the smallest scales of the turbulence. However, computational resources often limit the level of discretisation of the domain. That is why a compromise between domain discretisation refinement and accuracy of the results must be found. Large eddy simulations (LES) can be used as high-fidelity methods to resolve the largest scales of the turbulence until a certain threshold under which a subgrid model is applied and models the smaller scales. A quasi-direct numerical Simulation (qDNS) will employ a coarser mesh resolution than DNS comparable to LES, and without any subgrid model, assuming the smallest turbulent scales impact the solution marginally. The recommendations for appropriate mesh generation in LES and qDNS are provided in chapter 4

In this thesis, two-phase flow simulations are investigated in an industrial context and more particularly two-phase stratified flows with a large scale sharp

interface, where high shear occurs between the two phases. The interfacial shearing is due to the high velocity gradients between the two phases, one usually being liquid and the other gaseous. Two-phase flows are encountered in aero-engine cores, gearboxes and more especially in bearing chambers where wavy films can be initiated in a shear driven air-oil flow. Figure 1.1 shows the location of the rear bearing chamber in a Rolls-Royce Ultrafan aero-engine. The flow of interest is circulating in the external portion of the chamber in the wall area and contains oil and air represented in red and white respectively in figure 1.2.

It goes without saying that the urgent necessity of reducing our emissions directly involves the aviation industry, which accounted for 2.5% of the total CO_2 emissions in 2018 [47], and contributed to global warming with a share of 3.5% through radiative forcing [81]. Those emissions have doubled since 1987 at an analogous rate as total carbon emission. The research developed in this thesis falls within the Cornerstone project, coming from the ongoing partnership between Rolls-Royce plc and the universities of Nottingham, Oxford, along with Imperial College London and Queen's University of Belfast. This partnership was made with the objectives to improve aero-engine efficiency, reduce noise and carbon emissions in new and future designs. These objectives can be qualified as key targets as aviation has radically changed its ways to design aero-engines over the last 50 years in order to improve their performance and now to meet present and future requirements in terms of carbon emissions. The International Air Transport Association (IATA) has for instance fixed the environmental goal of reducing aviation's net CO_2 emissions to half of what they were in 2005 and make flying net zero by 2050 [68]. In order to achieve this goal, aviation has been responding by designing more efficient engines,

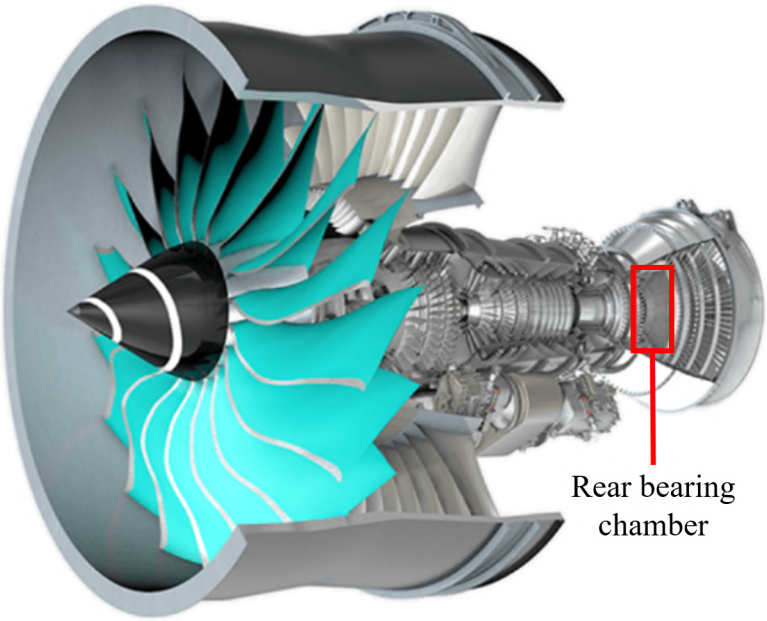


Figure 1.1 Rolls-Royce Ultrafan aero-engine [109] and rear bearing chamber location

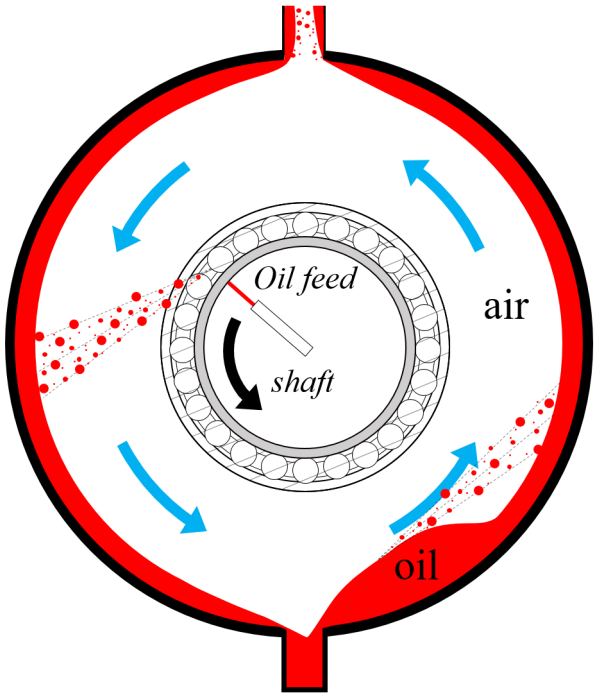


Figure 1.2 Schematic of a bearing chamber cross section

identifying weight savings, using sustainable aviation fuels (SAF), or the use of new propulsion technologies (hydrogen).

1.2 Problem statement

With the aim of improving their performance, future aero-engines are designed to increase the shaft speed and reliability. Bearing chambers have the fundamental role of distributing and collecting the oil in the engine for lubrication purposes. Additionally, the bearing chamber is the location in which the cooling of the lubricating oil occurs, as the air mixes with it [19]. As a consequence, air-driven and sprayed liquid oil flows are observed in the rotating chamber. Therefore, it is essential to be able to predict the oil film distribution in the bearing chamber to prevent from oil coking and deterioration.

The realisation of numerical predictions of two-phase shear flows is a challenging task. As mentioned previously, CFD models are mostly employed for flow simulations in the industry, as experiments and high fidelity simulations are time consuming and expensive to carry out. However, current CFD models like the averaged RANS models ($k-\omega$, $k-\varepsilon$) tend to overestimate the turbulence intensity at the interface between the two phases due to the high gradients of velocities from one phase to another in the shearing zone around the interface [32] [101]. Modelling interfacial turbulence has indeed become a real challenge in stratified flows [41] [141] and more specifically in the prediction of the oil thickness distribution in aero-engine bearing chambers. Moreover, it is especially difficult to fully understand interfacial turbulence as experimental measurements are exceptionally hard to carry out in the region of interest, as wavy interfaces add more constraints to the process [9]. Figure 1.3 highlights

the flow of interest in a bearing chamber showing the sharp interface of the air-driven oil film. In order to improve the prediction of the air shear-driven oil film thickness distributions in aero-engine bearing chambers and optimise the process of improving the engine performance, the inability of the current RANS models to produce accurate interfacial turbulence must be addressed.

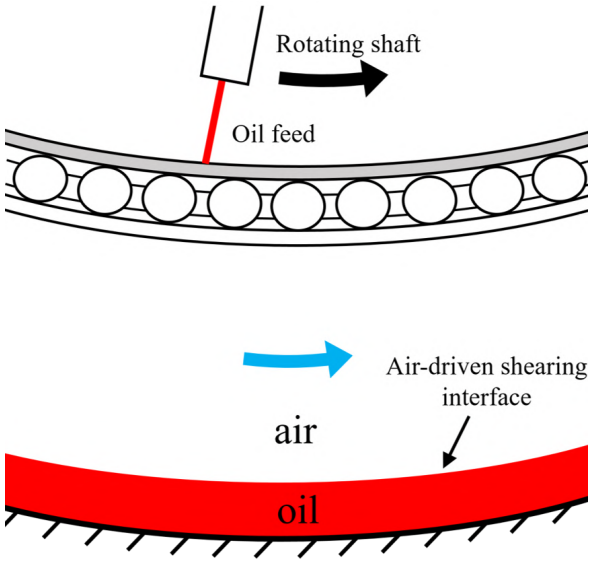


Figure 1.3 Schematic of a zoomed view of a bearing chamber cross-section

1.3 Contribution to knowledge and objectives

In order to tackle the RANS models' inability to correctly predict interfacial turbulence and produce accurate results for stratified flows with a sharp interface, the following questions were answered:

- What is the current state of the art on modelling two-phase flows using RANS methods, what are the up to date developed strategies to overcome the poor interfacial turbulence modelling, and what are their known limitations?

- In the context of stratified gas-liquid shearing flows, what are the most effective high-fidelity simulations and numerical methods to employ, what is the behaviour of the fluids, and how is turbulence distributed in those flows?

- Can a standard RANS model be driven efficiently by high-fidelity simulations results in order to appropriately predict two-phase shearing flows?

- How can RANS models benefit from the use machine learning tools for the types of two-phase shearing flow simulations to be carried out in bearing chambers specifically and what are the limitations of such machine learning tools?

The present research aimed to investigate the physics of stratified gas-liquid flows, and provide new strategies to overcome standard RANS models' failure in accurately evaluating turbulent high shear interfacial regions and thus to model correctly two-phase shearing flows with large scale and sharp interfaces, and especially thin-film flows encountered in aero-engine bearing chambers. The dissertation reports on existing strategies that have been already developed to correct RANS models used for stratified flow simulations in order to understand the limitations of those methods in chapter 3. For example, it was found that damping the turbulence levels in the interface region led to an improvement of the solution. This strategy is based on the hypothesis that the denser phase of a stratified flow behaves like a wall for the lighter phase. In the context of the bearing chamber shear-driven flows, the oil film would be seen as a wall by the gaseous air phase. This method, as well as other existing methods come with their share of limitations, which were brought to light.

The open source CFD code Open∇FOAM® was employed to carry out all the simulations presented in this work. The implementations of new methods for averaged models were also tested with OpenFOAM. Two major multiphase flow modelling methods are available in OpenFOAM to describe the motion of the phases. Thus, the two-phase flow simulations carried out in this work employed the Volume of Fluid (VOF) method in which one set of momentum equations for the whole mixture is used, and the Euler-Euler method in which one set of momentum equations for each phase is employed. Both methods were compared in a preliminary study in chapter 4 in order to determine which solver was the most appropriate to perform the two-phase flow simulations of air-driven liquid films with a sharp, large scale interface. The OpenFOAM solver `interFoam` based on the VOF method was tested against the solver `multiphaseEulerFoam` based on the Euler-Euler method. All the numerical methods employed to carry out the research were presented in chapter 2. Details on the types of simulations employed, the turbulence models, the multiphase solvers, and the solution algorithms were provided, as well as the machine learning methods used for the implementation and training of neural networks.

The research aspires to inform interfacial turbulence levels in a standard RANS model – namely the Wilcox's RANS $k-\omega$ turbulence model – in order to improve its ability to model the types of stratified flow that are found in aero-engines' bearing chambers. More precisely, quasi-DNS high-fidelity simulations were carried out to understand the physics and behaviours of such flows and to study detailed interfacial turbulence in chapter 4, after the preliminary comparative study between the two multiphase approaches. The external portion of the bearing chamber containing the stratified two-phase flow with a sharp interface was numerically simplified into a horizontal peri-

odic channel containing the air phase on top of the water phase. Two main configurations were studied and compared to existing experiments. A deep water case called the "thick-film" configuration, for which the physics of the fluids and turbulence levels were investigated in chapters 4 and 5. The numerical results obtained in the thick-film configuration were compared to the experiments of Fabre et al. [35] who investigated stratified air-water flows in a horizontal and rectangular channel. A shallow-water "thin-film" configuration employed for various flow regimes for which the air phase flow quantities was studied in details in chapter 6. The numerical results obtained in the thin-film configuration were compared against the experimental work of Hann and Kim [54][76]. The assessment of the simulations performance and comparisons to the experiments were based on the study of averaged flow stream velocity, turbulent kinetic energy, Reynolds stress profiles, and in some cases on the turbulence dissipation and specific turbulence dissipation profiles. Vorticity fields along with the Q-criterion method were employed for the identification of turbulent structures in the flow. The two-phase flows investigated in this thesis are always turbulent in the gaseous phase, turbulent in the liquid phase for the thick-film configuration, and laminar in most of the thin-film cases.

The strategy developed to inform the standard RANS $k-\omega$ model's interfacial turbulence using high-fidelity qDNS results was presented in chapter 5. It was found that re-balancing the budget of the specific turbulence dissipation rate in the RANS $k-\omega$ equations with an appropriate source term led to significant improvements, by driving the ω equation towards the high-fidelity solution. To serve this purpose, machine learning models were trained with the high-fidelity quasi-DNS data in order to predict such appropriate budget corrections for the standard RANS $k-\omega$ model. The machine learning models

were developed with feed-forward neural networks (FFNN) implemented by using the open source PyTorch library in Python. Two methods were proposed and developed in order to inform the RANS $k-\omega$ model for the simulation of thin-film flows using the machine learning models. The first method called the "frozen correction field" method only adds one pre-processing step to the simulation in which the machine learning model performs the prediction of a correction budget to be applied in the ω transport equation during the whole RANS simulation, without extending computational time, although, it comes with limitations that are detailed in chapter 6. The second method called the "adaptive correction field" method, is more complex to set up and increases simulation times to a certain extent, but provides more accurate and physical solutions and is potentially much more versatile. The process of both methods is detailed, and tests are provided in chapter 6. The methodology employed for the implementation of the machine learning models, their structure, training, coupling with the RANS model, and the results produced in test cases are also presented.

1.4 Thesis layout

In chapter 2, general knowledge on two-phase flows and numerical methods for multiphase flows are given. More specifically, the VOF and Euler-Euler methods are presented, as well as the open source CFD code OpenFOAM employed for the simulations, turbulence solving and modelling methods, discretisation schemes, and solution algorithms. An introduction to machine learning basics and the implementation of neural networks is also provided in this chapter.

A brief literature review investigating existing numerical and experimental work on two-phase shearing flows and the use of machine learning methods in CFD is given in chapter 3. More specifically, the reference experiments of Fabre et al. [35] in the thick-film configuration and of Hann and Kim [54][76] in the thin-film configuration are introduced.

Chapter 4 first presents the preliminary numerical work on the multiphase solver comparison between the VOF and Euler-Euler methods in LES. It is followed by a deeper analysis of flow turbulence in the thick-film configuration using quasi-DNS with the VOF method.

A proof of concept on how the $k-\omega$ turbulence model can be driven by high-fidelity data to perform RANS simulations of two-phase shear flows is proposed in chapter 5. The "frozen correction field" method is introduced, and the limitations and improvements of this method are discussed. A simple machine learning model is also implemented and tested to close the chapter.

Chapter 6 consists of three parts in which thin-film qDNS simulations are performed and compared to the reference experiments, and two machine learning models are implemented and tested with the two developed methods, namely the frozen correction field method and the adaptive correction field method, in order to inform the $k-\omega$ turbulence model for RANS simulations of thin-film flows. A range of test cases is investigated and each method and machine learning model is discussed.

In the last chapter 7, the general conclusions on the work carried out during this research project and the results obtained are given. Finally, recommenda-

tions for further improvements are proposed, more particularly on the methods developed in the context of CFD models coupled with machine learning models for the purpose of improving the industrial simulations of shear-driven film flows.

Chapter 2

Background knowledge on two-phase flows and numerical methods

In this chapter, a brief introduction to two-phase flows physics is presented in section 2.1 as well as numerical methods for two-phase flow modelling in section 2.2. Additional numerical methods for CFD simulations are described in section 2.3. Finally, an introduction to machine learning methods is given in section 2.4.

2.1 General knowledge on two-phase flows

Two-phase flow fluid mechanics refers to flows in which two fluids are present together. This could involve the same fluid in two different phases such as liquid water and vapour water, or two different fluids in the same phase such as liquid water and oil, or even two different fluids in different phases like liquid water and air for example. The behaviour of a two-phase flow can

strongly differ from behaviours found in single-phase flows and this is why the formulation and prediction of two-phase flows is still a great challenge in fluid mechanics and computational fluid dynamics.

The behaviour of a two-phase flow varies significantly regarding numerous parameters such as the types of fluid and phase involved, the geometry and configuration of the studied system, the temperature of the system, the fluid velocities, etc. Amongst the various possible two-phase flow configurations, stratified flows and dispersed flows are very common when studying gas/liquid mixtures:

- Stratified flows are often encountered in setups using low to moderate flow-rates at which the interface between the two phases remains continuous and sharp. In this configuration, the phases stay separated because of gravity and surface tension forces. The denser fluid remains at the bottom and the shape of the interface is controlled by the competition between inertia forces, which tend to deform it, and gravitational and superficial tension forces, which tend to maintain its sharpness. The domination of the gravity and superficial tension forces results in the interface waviness damping. Examples of stratified flows are shown in figure 2.1.

- Dispersed flows are usually observed when the relative flow-rate between the two fluids is important and inertia forces are so important that the interface is strongly deformed and cannot remain sharp, triggering the mixing of the two phases and causing the formation of bubbles or droplets. Examples of dispersed flows are shown in figure 2.1.

This dissertation focuses on the study of gas/liquid stratified flow configurations that can be observed in aero-engine bearing chambers. At high flow

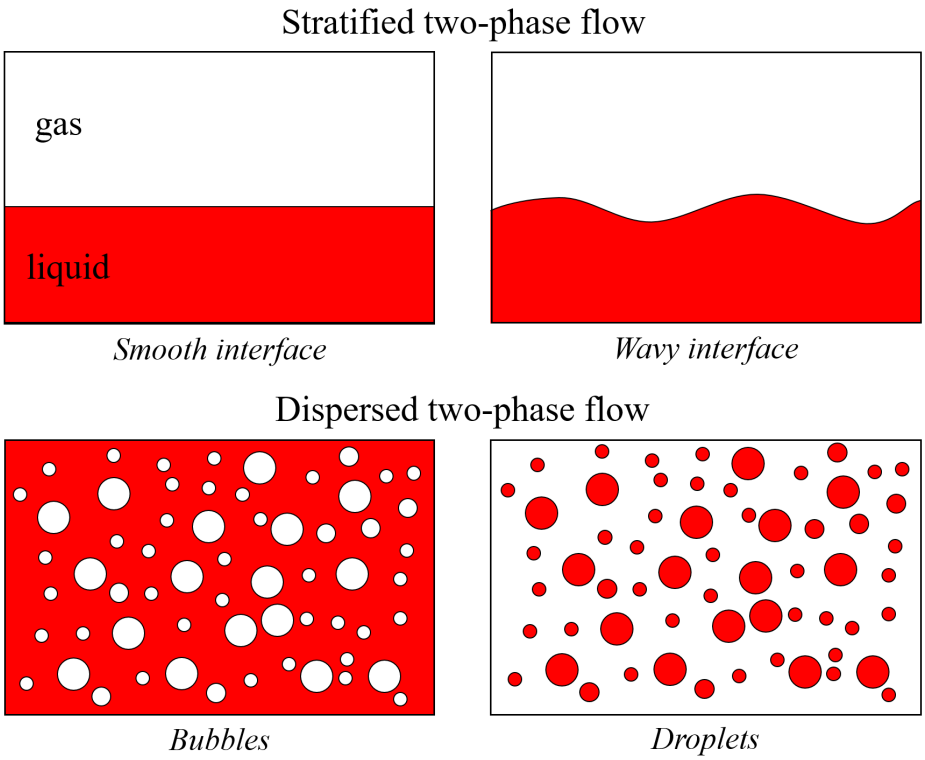


Figure 2.1 Schematics of different gas/liquid two-phase flow configurations

rates, those configurations show high gradients of velocity between the two phases and observable high shear forces at the interface, which is therefore deformed and made wavy. Even though dispersed flows are also present in bearing chambers, the present research remains in the framework of stratified flows and does not study dispersed flow configurations that would happen at even higher flow rates. Hence the two-phase flows' interface remains continuous and sharp in all studied cases of this dissertation. This type of flow is mainly observed on the external wall of the bearing chambers (c.f. figures 1.2 and 1.3).

2.2 Numerical modelling of two-phase flows

The complexity of the physics involved in two-phase flows, involving the direct impact of the turbulence on the flow behaviour and more specifically on

the interface deformation, is investigated experimentally for the development of analytical or empirical models, as well as numerically in order to apply experimental findings and reproduce physical phenomena. The present research focuses on the development of numerical tools for turbulence models based on approximations made on the the Navier-Stokes equations, which are used to describe the behaviour of fluids and can be written as follows [30]:

$$\frac{\partial \rho}{\partial t} + \rho \nabla \cdot \mathbf{u} = 0 \tag{2.1a}$$

$$\left(\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \times \mathbf{u}) \right) = -\boldsymbol{\nabla} p + \rho \mathbf{g} + \nabla \cdot \overline{\overline{\boldsymbol{\tau}}} \tag{2.1b}$$

with the strain rate symmetrical tensor $\overline{\overline{\boldsymbol{\tau}}}$:

$$\overline{\overline{\boldsymbol{\tau}}} = \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^{\intercal}) \tag{2.2}$$

where $\mathbf{u}$ is the velocity, p is the pressure, ρ is the fluid density, μ is the fluid dynamic viscosity, and $\mathbf{g}$ is the gravitational acceleration. Equation 2.1a refers to the continuity equation or the mass conservation equation, while equation 2.1b refers to the momentum equation.

For the sake of this research, one considers all fluids as Newtonian viscous and incompressible. Thus, one can rewrite equations 2.1a and 2.1b as:

$$\nabla \cdot \mathbf{u} = 0 \tag{2.3a}$$

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \times \mathbf{u}) \right) = -\nabla p + \rho \mathbf{g} + \nabla \cdot \overline{\overline{\boldsymbol{\tau}}} \tag{2.3b}$$

Solving numerically the Navier-Stokes equations (2.1a and 2.1b) is a complex task, especially for two-phase flows. For the last decades, predicting the behaviour of a flow phenomena containing two-phases or more within the same computational domain has been widely investigated by researchers. Numerous approaches were developed [91], using the single fluid formulation with interface identification methods when the interface between the two fluids is well defined, or the two-fluid formulation when dealing with dispersed phases for example. In order to numerically treat the interface, one can distinguish two main methods: Lagrangian and Eulerian methods [105]. The difference between the two methods stands in the way the interface is represented. In Lagrangian methods, the interface is tracked explicitly using adapting meshes that fit the deforming interface with time. Meanwhile, Eulerian methods use fixed meshes and the interface is not tracked explicitly but reconstructed using the phase volume fraction [8].

The Lagrangian approach keeps the interface as a discontinuity and explicitly tracks its evolution without using any model for its definition or impact on the rest of the flow. In this approach, boundary conditions can be applied directly at the interface [33]. The Lagrangian approach is used in the particle-in-cell (PIC) method [55] in which particles are described by their finite mass, implying that their movement causes a mass, momentum and energy change which are computed. In the smooth particle hydrodynamics (SPH) method [46], each particle of the flow have finite volumes that travels continuously from cell to cell in the computational domain, and in which more than one particle at the time can be present in one cell.

Eulerian methods require additional modelling or equations to obtain the location of the phases and interface [132][127]. In these methods, the interface is not tracked explicitly and a scalar function is introduced to indicate the phases and set the initial and boundary conditions of the phases. This function is defined for the entire computational domain and allows for the reconstruction of the interface at each time step of the simulation. In Eulerian methods, the interface is diffused and requires a well refined mesh where it is located to reduce the inaccuracies.

Some hybrid Eulerian-Lagrangian methods such as the front-tracking method couple the advantages of both methods to predict multiphase flows. This method uses a fixed Eulerian computational grid, and the interface is tracked and moves through the grid [136].

Because of the the available computational resources and tools in an industrial context, it was decided to solve the Navier-Stokes equations using Eulerian approaches. Eulerian methods are indeed widely used in the research field for CFD problems [105] by means of the discretisation of the transport equations in finite elements [62]. As previously mentioned, one can predict the behaviour of two phases occupying the same computational domain using different methods that can be classified into two groups [89]: the interface identification methods in which the two phases are separated by a defined sharp interface and which quality depends on the local mesh refinement; and the interface modelling methods that model the momentum transfer with respect to each phase's relative motion and the size of the dispersed parts of the fluids. Both approaches are available in OpenFOAM and will later be compared using Volume of Fluid (VOF) as a method of interface identification [63][28], and

the two-fluid model or Euler-Euler model as a method of interface modelling [70][110]. In order to decide on the best approach for our two-phase shearing flow problem, a preliminary study was carried out in chapter 4. It compares the VOF and Euler-Euler methods against experiments of a stratified flow in a horizontal channel containing a liquid phase (water) and a gaseous phase (air).

2.2.1 The Volume of Fluid method

The VOF method has become one of the most used methods amongst interface identification methods and was first introduced in 1976 [96] and published in a journal for the first time in 1981 [63]. The reasons this method is widely used in two-phase flow CFD include its simplicity of implementation in CFD codes and its ability to deal with relatively complex flow configurations involving strong deformations of the interface. In the VOF method, the two phases share the same velocity field and pressure field. A scalar function is defined to indicate the phase: $I(\mathbf{x}, t)$, which is advected by the velocity field $\mathbf{u}(\mathbf{x}, t)$. Assuming the studied problem is a two-phase flow, one can define $I(\mathbf{x}, t)$ such as:

$$I(\mathbf{x}, t) = \begin{cases} 1, \text{ if } \mathbf{x} \text{ is in primary at time } t \\ 0, \text{ if } \mathbf{x} \text{ is in secondary phase at time } t \end{cases} \tag{2.4}$$

And so, the phase volume fraction $\alpha(\mathbf{x}, t)$ can be defined as the integrated value of this function on each cell volume V_c (control volume) of the domain mesh, i.e.:

$$\alpha(\mathbf{x}, t) = \frac{1}{V_c} \int_{V_c} I(\mathbf{x}, t) dV_c \tag{2.5}$$

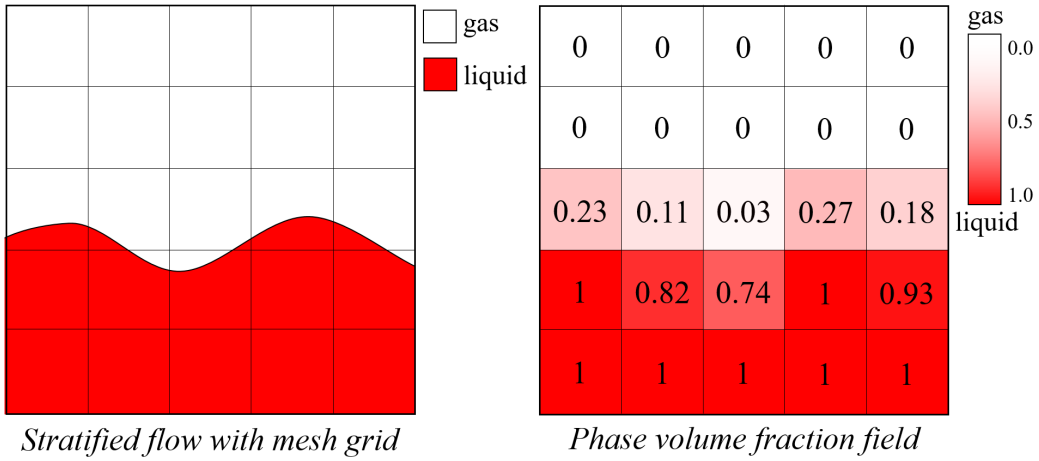


Figure 2.2 Illustration of the volume of fluid method applied to a stratified flow with a sharp interface

The phase volume fraction designates the ratio of volume of cell that is filled with the selected primary phase. According to equation 2.5, the phase volume fraction takes the value of 1 in cells entirely filled by the primary phase and the value of 0 in cells where there the presence of the primary phase is not detected i.e. entirely filled with the secondary phase. Elsewhere, the phase volume fraction respects the strict condition $0 < \alpha < 1$. In order to capture the interface, two categories of VOF methods namely the geometric and algebraic methods are employed. An illustration of this method is shown in figure 2.2 applied to a stratified flow in which the primary phase is the liquid. In geometric VOF methods, an approximation of the interface height is obtained geometrically (e.g. plane, line), while it is depicted by an analytic function in algebraic VOF methods (e.g. polynomial function, trigonometric function). OpenFOAM uses an algebraic VOF method to capture the interface.

Assuming the indices 1 and 2 respectively refer to phase 1 (primary phase) and 2 (secondary phase) at all time t, one can for example write $\alpha_1(\mathbf{x}, t) = \alpha$

and so, $\alpha_2(\mathbf{x}, t) = 1 - \alpha_1(\mathbf{x}, t) = 1 - \alpha$. Hence, one can describe the physical properties of the flow in the entire domain using the phase volume fraction, such as the flow density ρ writes $\rho = \alpha\rho_1 + (1 - \alpha)\rho_2$ and analogously the dynamic viscosity μ writes $\mu = \alpha\mu_1 + (1 - \alpha)\mu_2$.

Therefore, the two-phase flow can be treated as a unique fluid. α becomes a variable of the problem and is advected by the velocity field. Hence, one can then reformulate the continuity equation 2.1a as follows:

$$\frac{\partial \alpha}{\partial t} + \nabla \cdot (\alpha \mathbf{u}) = 0 \tag{2.6}$$

Introducing the resulting interaction forces between the phases $\mathbf{F}$, caused by the surface tension between the two-phases σ and defined as:

$$\mathbf{F} = \sigma \nabla \cdot \mathbf{n} \nabla \alpha \tag{2.7}$$

with $\mathbf{n}$ the unit vector giving the normal to the interface that can be written as $\mathbf{n} = \nabla\alpha/|\nabla\alpha|$, one can rewrite the momentum equation such as:

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \times \mathbf{u}) \right) = -\nabla p + \nabla \cdot \overline{\overline{\boldsymbol{\tau}}} + \mathbf{F} + \rho \mathbf{g} \tag{2.8}$$

Solving the advection of the quantity α in two-phase flow problems remains a problem of a considerable complexity as the choices of solution methods and discretisation schemes order have great impacts on the solution. The discontinuous character of the α field makes those choices delicate: a first order discretisation scheme produces very stable solutions but also very diffusive, increasing the inaccuracies in the interface prediction, while a high order scheme provides accurate solutions with a sharp interface but is unstable [82][20].

It thus becomes important to find a compromise between stability and accuracy.

In order to address this issue, numerical schemes with flux limiters such as the total variation diminishing (TVD) scheme [57] were introduced and combine the bounded character of the upwind schemes (first order) and the sharpening property of the higher order schemes. The TVD schemes are working in a way that they adapt their capacity regarding the numerical needs of the simulation.

The VOF method remains one of the most used multiphase flow modelling methods due to its numerous applications in diverse domains. Free surface problems even with strong instabilities at the interface are usually well modelled by VOF under the condition that the mesh is fine enough in the interface area including the regions where the interface travels locally [36]. The simulations presented in this dissertation remain in the context of smooth to fairly wavy free surfaces, which essentially keep the interface in the same level area depending on the simulated flow regime. Thus a fixed mesh with refinements in the interfacial area is needed, with more refinement adjustments for the waviest flows.

In order to perform the VOF simulations, the `interFoam` solver of OpenFOAM was used. This solver was implemented following equations 2.6 and 2.8 with an additional compression term in the volume fraction equation that amplifies the sharpening of the interface between the two phases [139]. The interface compression method was developed by Wardle and Weller [148] in order to capture large scale sharp interfaces in two-phase flow simulations. This feature proves to be particularly useful for shear-driven flows [88] where the high gradients of velocity at the interface can cause diffusion inaccuracies. The

artificial interface compression term is activated by an interface compression coefficient C_α in the interfacial region. The phase volume fraction then reads:

$$\frac{\partial \alpha}{\partial t} + \mathbf{u} \cdot \nabla \alpha + \underbrace{\nabla \cdot [\mathbf{u}_{\text{comp}} \cdot \alpha(1-\alpha)]}_{\text{artificial compression term}} = 0 \tag{2.9}$$

where $\mathbf{u}_{\text{comp}}$ is the compression velocity applied normally to the interface and that controls the compression level with a compression coefficient C_α such as:

$$\mathbf{u}_{\text{comp}} = \min\{C_\alpha |\mathbf{u}|; \max\{|\mathbf{u}|\}\} \frac{\nabla \alpha}{|\nabla \alpha|} \tag{2.10}$$

where the unit vector $\nabla\alpha/|\nabla\alpha|$ provides the compression velocity direction that will be applied normally to the surface. A value of 0 of the compression coefficient C_α will simply cancel the interface compression term; a value of 1 corresponds to conservative compression and is generally used; and larger values correspond to enhanced compression of the interface [40][148]. In order to avoid any unphysical, extreme values of the phase volume fraction i.e. outside of the interval $[0, 1]$, the multi-dimensional limiter for explicit solution (MULES) algorithm [156] was implemented in OpenFOAM to bound α. A semi-implicit version of the MULES algorithm was added to OpenFOAM [51] allowing for less restraining simulation conditions i.e. for the use of higher Courant (CFL) numbers [24].

2.2.2 The Euler-Euler method

The Euler-Euler method also called two-fluid model [70][71] distinguishes itself from single-fluid approaches by modelling the interfaces instead of reconstructing it. Similarly to the VOF method, the Euler-Euler method defines each phase by a phase indicator $I_i(\mathbf{x}, t)$ where the subscript i designates phase

1 or 2 in two-phase flows. In the Euler-Euler approach, the velocity field is described separately for each phase but the pressure field is shared by the two phases. Therefore, each phase has its own set of mass and momentum equations and own phase volume fraction field denoted $\alpha_i(\mathbf{x}, t)$. Moreover, each phase possesses its own flow characteristics such as the densities can be noted ρ_i and the dynamic viscosities μ_i. Thus, mass and momentum equations for incompressible and viscous Newtonian flows can be written as follows:

$$\frac{\partial \alpha_i}{\partial t} + \nabla \cdot (\alpha_i \mathbf{u}_i) = 0 \tag{2.11a}$$

$$\frac{\partial \alpha_i \rho_i \mathbf{u}_i}{\partial t} + \nabla \cdot (\alpha_i \rho_i \mathbf{u}_i \times \mathbf{u}_i) = -\alpha_i \nabla p + \alpha_i \rho_i \mathbf{g} + \nabla \cdot (\alpha_i \mu_i \nabla \mathbf{u}_i) + F_{D,i} + F_{S,i} \tag{2.11b}$$

In two-fluid models, the interfacial forces are reduced to the drag force F_D and surface tension F_S. In order to model the resulting transfer of momentum between the two phases, the drag force is applied such as one phase is considered as continuous and the other dispersed. Using the subscripts d and c respectively for the dispersed and continuous phase, the drag force for interpenetrating phases writes:

$$F_{D,i} = \frac{3}{4} \alpha_c \alpha_d \rho_c C_D \frac{|\mathbf{u}_d - \mathbf{u}_c|(\mathbf{u}_d - \mathbf{u}_c)}{d_d} \tag{2.12}$$

where d_d is the particle diameter of the dispersed phase and C_D is the drag coefficient and can be calculated using the empirical Schiller-Naumann model [117], which writes:

$$C_D = \begin{cases} 24\left(1 + 0.15 Re_p^{0.687}\right) & \text{if } Re_p \leq 1000 \\ 0.44 & \text{else} \end{cases} \tag{2.13}$$

with Re_p the particle Reynolds number, defined as:

$$Re_p = \frac{2r_d|\mathbf{u}_d - \mathbf{u}_c|}{\nu_c} \tag{2.14}$$

where ν_c is the kinematic viscosity of the continuous phase.

Because of its popularity in two-phase flows and its availability in OpenFOAM, the Euler-Euler method was investigated in this thesis. However, equations 2.2.2 and 2.2.2 are not designed for stratified flows and one should thus not expect to obtain the best results with the Euler-Euler method. This assumption will be confirmed in chapter 4.

The interfacial compression of the interface was originally developed for two-fluid models [148] before being implemented in the VOF solvers of OpenFOAM. The Euler-Euler simulations presented in this dissertation were carried out using the solver `multiPhaseEulerFoam` [110] and includes the artificial interface compression term that is also activated by the interface compression coefficient C_α in the interfacial region. The transport equation of the phase volume fraction α_i is identical to the one described in the VOF method for α (equation 2.9). Similarly to the VOF method, the compression coefficient C_α is usually taken as 1 for Euler-Euler simulations and a value of 0 would leave the phase dispersion unaltered.

2.3 Additional numerical methods for CFD simulations

2.3.1 OpenFOAM

OpenFOAM was chosen to perform all the numerical simulations presented in this dissertation, as it features many advantages. Open∇FOAM® (Open Field Operation and Manipulation) [40] is a multi-physics toolbox mainly focusing on the solving of fluid mechanics equations. It has been distributed under a free and open-source GNU GPL licence since 2004 by the British society OpenCFD Ltd. Initially, OpenFOAM was developed by the Imperial College London in C++, which wanted a code based on the finite volume method (FVM) and benefiting from latest programming language advances. The software is yearly upgraded and actively maintained with regular corrections and updates. It benefits from a large community contributing to its development (around 10000 users [52]). It is mainly composed by a free C++ software library and different tools in the form of libraries and application solvers. Numerous solvers are available in OpenFOAM for a wide range of physical applications, such as compressible and incompressible flows, multiphase flows, combustion, chemical reactions, heat transfer, etc. Multiple turbulence models are also available such as several RANS and LES models. OpenFOAM seen as a C++ library is of great interest when it comes to developing and testing new models. Unlike the majority of most scientific codes written sequentially –usually with Fortran– it benefits from the power of object-oriented languages. This structure under the form of classes permits to be closer to the mathematical writing in terms of divergence, gradient, Laplacian, rotational operators, temporal derivative, etc. For example, take the momentum equation for the velocity field $\mathbf{u}$:

$$\partial_t(\rho \mathbf{u}) + \nabla \cdot (\phi U) = -\nabla p + \mu \nabla^2 \mathbf{u} \tag{2.15}$$

Then, equation (2.15) would simply be implemented in OpenFOAM as follows:

```
solve
(
    fvm::ddt(rho,U)
  + fvm::div(phi,U)
==
  - fvm::grad(p)
  + fvm::laplacian(mu,U)
);
```

While the discretisation of the different mathematical operators occupies a prominent place in the creation of sequential codes, OpenFOAM users do not have to worry about it when writing the scripts and thus can entirely focus on the representation of the physical models. The different discretisation methods are actually already coded in the classes of each operator. Users can also create and add the discretisation methods they desire in the corresponding operator class if not present in OpenFOAM already.

2.3.2 Finite volume spatial discretisation method

OpenFOAM uses a three dimensional Cartesian coordinate system with a finite volume method but allows for 2D and 3D problem solving. The finite volume spatial discretisation method is a spatial discretisation scheme that allows for the application of a set of equations in a computational domain. Most of the time OpenFOAM is used for the solving of non-linear transport

equations of fluid dynamics problems. The FVM consists in the division of the computational domain into a chosen number of control volumes (CV) using the integral forms of the equations as a starting point. These equations are integrated using the mid-point rule (second order accurate) and Gauss' theorem to convert volume integrals into surface integrals resulting in matrices. Fluid dynamic quantities can either be all defined at a single node of CV (co-located grid) [107] or separately defined. Thus, the scalar quantities (pressure, density, temperature, etc.) are located at the cell centre of the CV and flux quantities (velocity and momentum) are located on the cell faces (staggered grid) [56] as shown on figure 2.3. At other locations, quantity values are interpolated according to the chosen interpolation method.

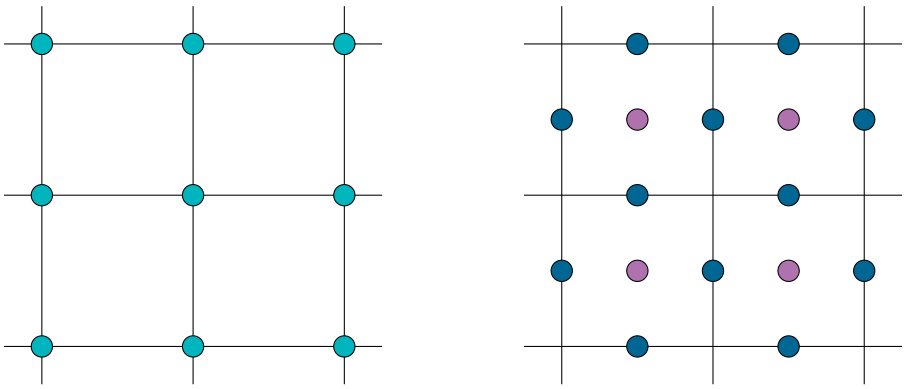


Figure 2.3 Examples of a 2D co-located grid (left) and of a 2D staggered grid (right)

2.3.3 Discretisation schemes

OpenFOAM can run CFD simulations with up to second order accuracy schemes. Those available schemes rely on the FVM method (see section 2.3.2) solving the general transport equation (second order) of ϕ, a continuous and differentiable physical function of the flow:

$$\underbrace{\frac{\partial \rho\phi}{\partial t}}_{\text{time derivative}} + \underbrace{\nabla \cdot (\rho\phi\mathbf{u})}_{\text{convection term}} = \underbrace{\nabla \cdot (\Gamma \nabla\phi)}_{\text{diffusion term}} + \underbrace{S_\phi}_{\text{source term}} \tag{2.16}$$

As required by FVM, one integrates the previous equation on a 3D cell control volume V_C:

$$\iiint_{V_C} \frac{\partial \rho\phi}{\partial t} dV_C + \iiint_{V_C} \nabla \cdot (\rho\phi\mathbf{u}) dV_C = \iiint_{V_C} \nabla \cdot (\Gamma \nabla\phi) dV_C + \iiint_{V_C} S_\phi dV_C \tag{2.17}$$

Using Green-Gauss theorem to write the volume integrals into surface integrals over the control surface S_C and obtain fluxes: $\iiint_{V_C} \nabla \cdot \phi dV = \iint_{S_C} dS \cdot \phi$

One finally obtains the following:

$$\frac{\partial}{\partial t} \iiint_{V_C} \rho\phi dV_C + \iint_{S_C} \underbrace{(\rho\phi\mathbf{u}) \cdot dS_C}_{\text{convection flux}} - \iint_{S_C} \underbrace{(\Gamma \nabla\phi) \cdot dS_C}_{\text{diffusion flux}} = \iiint_{V_C} S_\phi dV_C \tag{2.18}$$

Spatial discretisation

The flux terms are then interpolated using interpolation schemes. Solving a second order equation requires a second order or higher for discretisation in order to obtain higher accuracy. Linear interpolation or central differencing schemes are second order accurate, although, they are unstable as unbounded. A first order scheme such as the upwind differencing scheme is bounded but diffusive. As we are interested in solving two-phase flows with sharp interfaces, the second order total variation diminishing (TVD) scheme is found to be an appropriate compromise between stability and accuracy [57], as already mentioned in section 2.2.1. The addition of a limiter function ψ allows for a bounded and accurate scheme in TVD methods. Assuming a convective flux

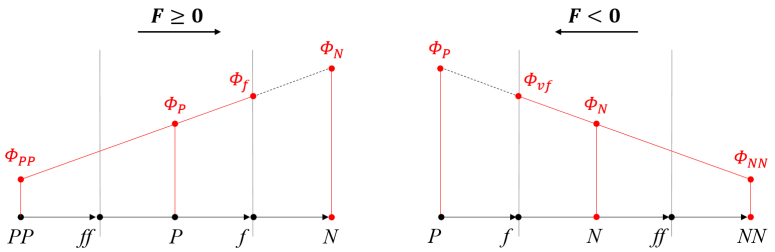


Figure 2.4 Convective flux F across a control volume of volume centroid P and face centroid f

F across a control volume of volume centroid P and face centroid f and the neighbour control volumes of centroid PP, N, and NN (see figure 2.4), a TVD scheme must respect the following conditions [128]:

$$\phi_f = \begin{cases} \phi_P + \frac{1}{2}\psi_P^-(\phi_P - \phi_{PP}) & \text{for } F \geq 0 \\ \phi_N - \frac{1}{2}\psi_P^+(\phi_{NN} - \phi_N) & \text{for } F < 0 \end{cases} \tag{2.19}$$

The slope limiter function ψ prevents oscillations by making the scheme bounded. In order to solve the diffusion fluxes one can use a second order upwind differencing such as $\phi_f = \phi_P + (2\boldsymbol{\nabla}\phi_P - \boldsymbol{\nabla}\phi_f) \cdot \mathbf{d}_{Pf}$ where $\mathbf{d}_{Pf}$ is the vector pointing to f from the centroid P.

Temporal discretisation

In order to solve the time derivative one may use any second order accurate scheme. The second order backward difference scheme for example can be described in two steps. The first order backward difference approximation of ϕ writes:

$$\frac{\partial\phi(t)}{\partial t} = \frac{\phi(t) - \phi(t - \Delta t)}{\Delta t} + \mathcal{O}(\Delta t^2) \tag{2.20}$$

The second order approximation is obtained by using more terms in the Taylor series development. Hence the second order backward difference approximation of ϕ writes:

$$\frac{\partial\phi(t)}{\partial t} = \frac{3\phi(t) - 4\phi(t-\Delta t) + \phi(t-2\Delta t)}{2\Delta t} + \mathcal{O}(\Delta t^2) \tag{2.21}$$

The second order backward time discretisation scheme is implicit and conditionally stable. As an alternative, the Crank-Nicolson method can be used to discretise the temporal derivative. It is implicit and unconditionally stable. In the Crank-Nicolson scheme [25], the spatial derivatives are half evaluated at the time n and half at the time $n+1$. By taking the Taylor series development around $(j, n+\frac{1}{2})$:

$$\frac{\phi_j^{n+1} - \phi_j^n}{\Delta t} = \frac{\alpha}{2\Delta x^n}\left[(\phi_{j+1}^n - 2\phi_j^n + \phi_{j-1}^n) + (\phi_{j+1}^{n+1} - 2\phi_j^{n+1} + \phi_{j-1}^{n+1})\right] + \mathcal{O}(\Delta t^2, \Delta x^2) \tag{2.22}$$

2.3.4 Turbulence modelling

Shear-driven flows are mainly investigated in this dissertation, and the studied regimes show turbulence generation in shearing areas like walls and interfaces. Thus, brief descriptions of the main approaches for solving and/or modelling the turbulence are presented in this section.

In CFD, the choice of a methodology to predict the turbulence depends on different factors. Turbulence can for example be solved directly by solving the Navier-Stokes equations without any models. This approach is known as direct numerical simulations (DNS) and they require to solve all the scales of

the turbulence of the flow, the Kolmogorov scale being the smallest scale of the turbulence [97]. Hence, employing DNS demands unrealistic computational resources for industrial and engineering applications, especially when the studied geometry are often very complex. DNS is thus mainly used in research with simplified geometry in order to improve our vision of the turbulence and understand better complex turbulent phenomena [92].

Alternatively to DNS, quasi-DNS (qDNS) and large eddy simulations (LES) can be performed, requiring less resources that DNS. LES solves the largest scales of the turbulence and applies turbulence models to predict the smallest scales, called subgrid models. Therefore, the use of models allows for coarser computational meshes than DNS. qDNS uses numerical dissipation to act as a subgrid scale model, so that the DNS mesh size can be considerably reduced [133]. One might also call it numerical LES (nLES). Although those methods remains computationally too expensive for typical engineering purposes, this dissertation employed qDNS simulations to generate large high-quality databases for the training of machine learning models. This choice was based on a study comparing both LES and qDNS simulations in chapter 4. It allowed for the production of high-fidelity results at reasonable computational costs.

For industrial applications, the Reynolds averaged Navier-Stokes (RANS) equations method is often chosen. One talks about RANS modelling and as its name suggests, it is obtained by applying averaging operations to the Navier-Stokes equations. Unsteady RANS (URANS) is also performed for industrial problems showing fluctuations. RANS models allows for the use of larger computational domains than in the previously described methods [95] as coarser mesh resolutions required in RANS simulations, which makes them

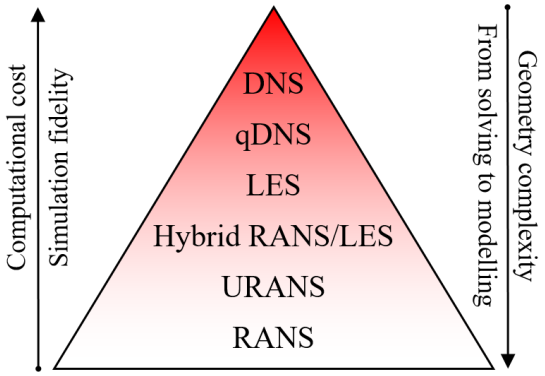


Figure 2.5 Classification of the main turbulence simulation methods. Adapted from [114]

potentially very interesting for complex engineering problems.

One can place all those turbulence simulation methods in a diagram (see figure 2.5) to better illustrate how each of them compares to the others in terms of computational resources needed, and accuracy of the solution.

The RANS model

The so called Reynolds decomposition of the velocity field leads to the RANS equations. It consists in substituting the instantaneous velocity field described in the momentum Navier-Stokes equation by the sum of the time averaged velocity and fluctuation velocity, such as:

$$u_i(x_i, t) = \overline{u_i(x_i, t)} + u_i'(x_i, t) \tag{2.23}$$

with $u_i = \mathbf{u}$ and $x_i = \mathbf{x}$. Note that this decomposition also applies to the pressure field such as $p(x_i, t) = \overline{p(x_i, t)} + p'(x_i, t)$. Applying the Reynolds decomposition to the momentum equation and considering that the time averaged fluctuation velocity is 0, gives the following:

$$\rho\left(\frac{\partial \bar{u}_i}{\partial t} + \nabla \cdot (\bar{u}_i \bar{u}_i)\right) = \nabla \bar{p} + \nabla \cdot (\mu \nabla \bar{u}_i) + \rho g - \rho \nabla \cdot \overline{u_i' u_i'} \tag{2.24}$$

Applying the Reynolds decomposition to the momentum equation results in the creation of an additional term in the right hand side (equation 2.24). The nonlinear term $\overline{u_i' u_i'}$ is also known as the Reynolds stress R_{ii} and is responsible for the turbulence flow physics, its production, destruction, and diffusion.

As for the turbulent kinetic energy (TKE) k, it is obtained by calculating the trace of the Reynolds stress tensor, such as:

$$k = \frac{1}{2}\mathrm{tr}(R_{ij}) \tag{2.25}$$

Many RANS models can be selected to predict the turbulence in a flow simulation. In this dissertation, the standard RANS $k-\omega$ model is investigated. Therefore, only a detailed presentation of this model is provided. One could cite some of the other most widely used RANS models [143] in the industry such as the Spalart-Allmaras (SA) model, the $k-\varepsilon$ model, or the $k-\omega$ SST model. Those models all present pros and cons depending on the flows that need to be predicted:

- The standard SA model [123] is a very stable one-equation model solving for the kinetic turbulent viscosity that is commonly used for aerodynamics problems but presents some limitations for shear flows and decaying turbulence problems [7];
- The Launder $k-\varepsilon$ model [80] is a two-equation model solving for the TKE k and turbulence dissipation ε that is often used for free-shear flows,

external flow interaction problems, however presents limitations to predict near-wall TKE [10];

- The Wilcox $k-\omega$ model [151] is also a two-equation model solving for the TKE and the specific turbulence dissipation rate ω. It is usually convenient for channel and walled flows and deals better than the $k-\varepsilon$ model with near-wall interaction, although it shows some excessive sensibility to ω in free stream flows and inlet boundary conditions which is normally not the case for the $k-\varepsilon$ model [38]. The latter version of $k-\omega$ aims to address this issue. More details about this model are presented thereafter;

- The Menter $k-\omega$ SST model [90] is a two-equation model based on the combination of the $k-\varepsilon$ in the free stream regions model and $k-\omega$ model in the near-wall regions. It takes advantages of the standard $k-\varepsilon$ and $k-\omega$ models.

The governing equations of the standard RANS original Wilcox $k-\omega$ model [151] for the turbulent kinetic energy k and for the specific dissipation rate ω write:

$$\frac{\partial \rho k}{\partial t} + \frac{\partial \rho u_i k}{\partial x_i} = \rho P_k - \beta^* \rho k \omega + \frac{\partial}{\partial x_i}\left[(\mu + \sigma_k \mu_t)\frac{\partial k}{\partial x_i}\right] \tag{2.26a}$$

$$\frac{\partial \rho \omega}{\partial t} + \frac{\partial \rho u_i \omega}{\partial x_i} = \alpha \frac{\omega}{k} P_\omega - \beta \rho \omega^2 + \sigma_d \frac{\rho}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i} + \frac{\partial}{\partial x_i}\left[(\mu + \sigma_\omega \mu_t)\frac{\partial \omega}{\partial x_i}\right] \tag{2.26b}$$

Here, ρ is the fluid density, $\mu = \rho\nu$ is the fluid dynamic viscosity with ν the fluid kinematic viscosity, t is time, u_i are the components of the velocity and x_i the Cartesian coordinates with $i = [\![1, 3]\!]$. $\alpha = \frac{13}{25}$, $\beta = 0.0708$, $\beta^* = \frac{9}{100}$,

$\sigma_k = \frac{3}{5}$, $\sigma_\omega = \frac{1}{2}$ and $\sigma_d = \frac{1}{8}$ are the closure coefficients of Wilcox's problem. The production of kinetic energy P_k for incompressible flow writes:

$$P_k = \tau_{ij} \frac{\partial u_i}{\partial x_j} \tag{2.27}$$

where the Reynolds stress tensor τ_{ij} according to the Boussinesq approximation [14] writes:

$$\tau_{ij} = -\overline{u_i' u_j'} = 2\nu_t \bar{S}_{ij} - \frac{2}{3} k \delta_{ij} \tag{2.28}$$

where $\nu_t = k/\omega$ is the kinematic eddy turbulent viscosity with μ_t the turbulence dynamic eddy viscosity, which writes:

$$\mu_t = \frac{\rho k}{\omega} \tag{2.29}$$

and $\bar{S}_{ij}$ is the mean strain rate tensor, such as:

$$\bar{S}_{ij} = \frac{1}{2} \overline{\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)} \tag{2.30}$$

The Wilcox's standard $k-\omega$ model as well as the other above presented two-equation turbulence models use wall functions in order to treat the turbulence in the near-wall region, in the boundary layer. Therefore, the way those models are calibrated depends directly on the so called 'law of the wall' introduced by Von Kármán in 1930 [146]. This law stipulates the existence of three regions in the near-wall region of the flow:

- The viscous sublayer or laminar sublayer which is the closest to the wall, dominated by the viscous effect of the flow and in which the turbulent scales are more dissipated that produced;
- The inertial sublayer or log-layer which is the furthest from the wall in the near-wall region and in which the flow is dominated by inertia forces (inertia scales);
- The buffer layer which is the intermediary layer between the viscous and inertial sublayers and in which inertia forces and dissipation compensate.

One can define the dimensionless friction velocity u_τ as a function of the wall shear stress τ_w in the viscous sublayer such as:

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}} \tag{2.31}$$

and assuming the normal to the wall is the y direction:

$$\tau_w = \mu \left(\frac{\partial u}{\partial y} \right)_{y=0} \tag{2.32}$$

One defines two dimensionless quantities from the friction velocity and wall shear stress: the dimensionless velocity u^+ and wall distance y^+:

$$u^+ = \frac{\bar{u}}{u_\tau} \tag{2.33a}$$

$$y^+ = \frac{y u_\tau}{\nu} \tag{2.33b}$$

Using the turbulent viscosity, one can rewrite the momentum equation under the following simplified form:

$$(\mu + \mu_t)\frac{\partial u}{\partial y} = \tau_w \tag{2.34}$$

Using the formulations 2.33a and 2.33b along with the definition of the wall shear stress in equation 2.32, and noting $\nu_t = \mu_t/\rho$ one can rewrite equation 2.34:

$$\left(1 + \frac{\nu_t}{\nu}\right)\frac{\partial u^+}{\partial y^+} = 1 \tag{2.35}$$

This formulation describes both viscous and turbulent (inertial) contributions. When studying the flow very close to the wall in the viscous sublayer, the turbulent effects are dominated by the viscous ones and can be neglected in equation 2.35 such as:

$$\boxed{u^+ = y^+} \tag{2.36}$$

This equation describes a linear velocity profile which is proportional to the wall distance. When moving away from the wall, the influence of the viscous terms diminishes until the turbulent effects become dominant and so one obtains the following:

$$\left(\frac{\nu_t}{\nu}\right)\frac{\partial u^+}{\partial y^+} = 1 \tag{2.37}$$

One can then use the Prandtl hypothesis that defines the turbulent viscosity as a function of the mixing length l_{mix} [102] and write the following for near-wall application: $l_{\text{mix}} = \kappa y$ where $\kappa = 0.42$ is the Von Kármán constant. Moreover, a dimensional analysis leads to $|\partial\bar{u}/\partial y| = u_\tau\ (\kappa y)$. One can then deduct the following:

$$\frac{\nu_t}{\nu} = \kappa y^+ \tag{2.38}$$

After integrating the equation 2.37 with respect to y^+ including the formulation 2.38, we obtain the following:

$$u^+ = \frac{1}{\kappa}\ln(y^+) + C \ , \qquad C \in \mathbb{R} \tag{2.39}$$

Or, using the empirical integration constant $E = e^{\kappa C} = 9.79$ representing the wall rugosity:

$$\boxed{u^+ = \frac{1}{\kappa}\ln(Ey^+)} \tag{2.40}$$

The three near-wall regions are represented in figure 2.6. The choice of the upper limit of the log-law region is subjective but most of the time is fixed at $y^+ \approx 300$ and normally depends on the Reynolds number Re. The higher Re, the greater upper log-law region limit.

The LES simulation

An alternative method for turbulent flow simulation is the large eddy simulation (LES). Similarly to the RANS method, LES is based on the decomposition of the instantaneous velocity field, which is partly numerically solved directly with the Navier-Stokes equations, and partly modelled then integrated in the final solution. The largest scales of the turbulence in the inertial range are solved while the effects of the smallest ones are modelled. The field is once again decomposed similarly to the Reynolds decomposition, although this time, using filter operations instead of average. The decomposition is theoretically obtained by use of a low-pass filter, as the convolution of a function with a filtering kernel G [113] is characterised by a cutoff length scale in the physical

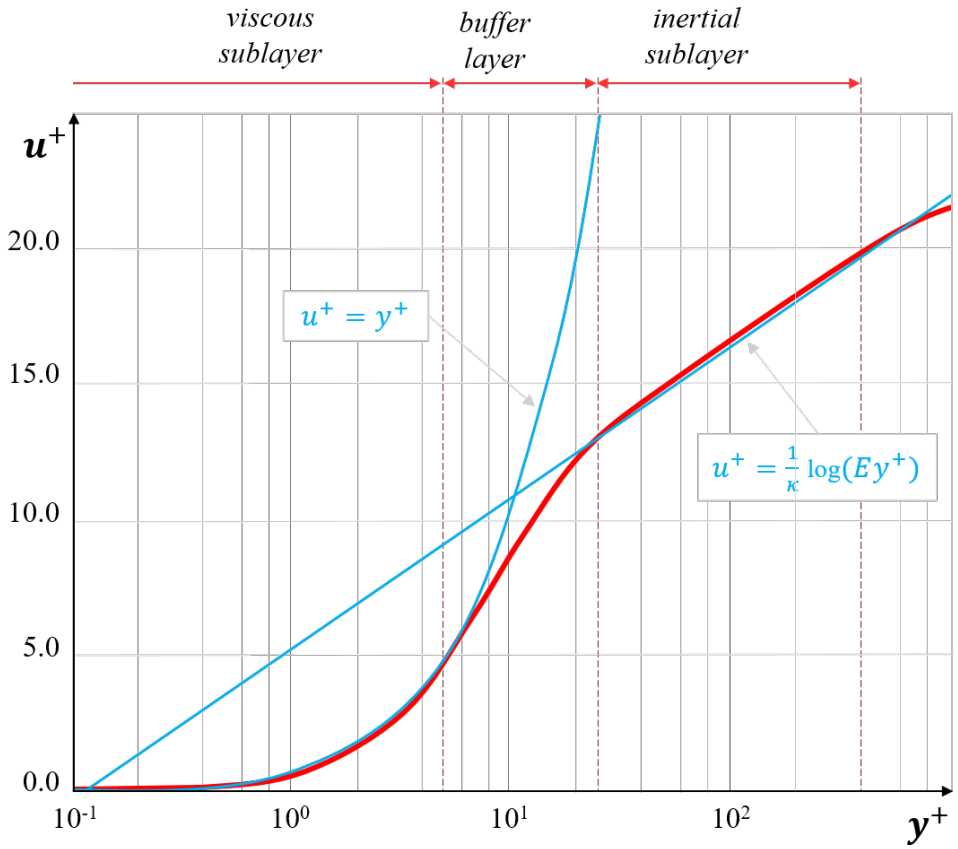


Figure 2.6 Standard law of the wall with sublayers ranges

space that depends on the choice of the filter model (e.g. the Smagorinsky sub-grid scale model [120]). Thus, the velocity $u_i(\mathbf{x}, t)$ field is decomposed [17] as the sum of the filtered velocity $\tilde{u}_i(\mathbf{x}, t)$ and its residual field $\hat{u}_i(\mathbf{x}, t)$ such as:

$$u_i(\mathbf{x}, t) = \tilde{u}_i(\mathbf{x}, t) + \hat{u}_i(\mathbf{x}, t) \tag{2.41}$$

Then, the velocity field $u(\mathbf{x})$ can be filtered such as:

$$\tilde{u}_i(\mathbf{x}) = \int_{-\infty}^{\infty} G(\mathbf{x} - \boldsymbol{\xi}) u(\boldsymbol{\xi}) d\boldsymbol{\xi} \tag{2.42}$$

Applying the decomposition and filtering to the Navier-Stokes equations:

$$\frac{\partial \tilde{u}_i}{\partial t} + \tilde{u_j}\frac{\partial \tilde{u}_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \tilde{p}_i}{\partial x_i} + \frac{\partial}{\partial x_j}\left(\nu \frac{\partial \tilde{u}_i}{\partial x_j}\right) + \frac{1}{\rho}\frac{\partial \tau_{ij}}{\partial x_j} \tag{2.43}$$

where $\tau_{ij} = \tilde{u}_i\bar{u}_j - \widetilde{u_i u_j}$ results from the non-linear advection terms and was already introduced as the Reynolds stress in equations 2.24 and 2.27.

When using the filtered strain rate tensor $\tilde{S}_{ij}$ already introduced in equation 2.30 in the mean form, for the resolved scale and the Boussinesq hypothesis [14] (equation 2.28), one can write:

$$\tau_{ij} - \frac{1}{3}\tau_{kk}\delta_{ij} = -2\mu_t \tilde{S}_{ij} \tag{2.44}$$

Similarly to equation 2.34, we get:

$$\boxed{\frac{\partial \tilde{u}_i}{\partial t} + \tilde{u_j}\frac{\partial \tilde{u}_i}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \tilde{p}_i}{\partial x_i} + \frac{\partial}{\partial x_j}\left[(\nu + \nu_t)\frac{\partial \tilde{u}_i}{\partial x_j}\right]} \tag{2.45}$$

where the pressure term includes the trace term $\frac{1}{3}\tau_{kk}\delta_{ij}$.

While LES remains a cheaper alternative to DNS when it comes to computational resources, it is advised to apply necessary grid refinements following classical guidelines to capture all the largest turbulent structures, especially the near wall structures that have a great impact on the flow and of smaller size than those found in the free-stream. A $y^+ \approx 1$ for the wall distance, $\Delta x^+ \approx 50$ in the flow direction, and $\Delta z^+ \approx 15$ in the remaining cross-stream direction, are recommended for LES [112]. Spalart et al. [124] recommended a grid resolution of at most ten times less refined than in DNS.

The quasi-DNS simulation

Quasi-DNS simulations (qDNS) are employed as realistic alternatives to DNS to carry out high fidelity simulations. As opposed to LES, qDNS do not use any sub-grid model to predict the behaviour of the smallest turbulent scales, it uses numerical dissipation instead, and as a consequence qDNS is often called numerical LES. The mesh must be refined without aiming for the Kolmogorov microscale η as one would do in DNS.

For two-phase flow simulations, the Kolmogorov number η varies depending on the phase type and regime. One formulation of this number is [100]:

$$\eta = (\nu^3/\varepsilon)^{1/4} \tag{2.46}$$

Where ε is the rate of energy dissipation of the phase and can be estimated as follows: $\varepsilon \sim U_b^3/L$ where U_b and L are the bulk velocity of each phase and the characteristic length scale of the flow (e.g. phase thickness, channel height, etc.) respectively. Hence we obtain the following estimation of the Kolmogorov spatial scale:

$$\eta = \left(\frac{\nu^3 L}{U_b^3}\right)^{1/4} \tag{2.47}$$

$$\tau = \frac{\eta^2}{\nu} \tag{2.48}$$

While a DNS simulation would require a refinement in the flow direction to the size $\Delta x = 2\eta$ [147], according to Tiselj et al. [133], qDNS needs around two to five times less resolution than DNS in the flow direction in terms of

number of cells in order to capture the necessary turbulent scales.

2.3.5 Solution methods

In this part, the methods employed to solve the pressure and velocity fields in the simulations are presented.

Pressure-velocity algorithms

In order to couple the momentum and the mass conservation, CFD solvers use solving algorithms such as the pressure-implicit split-operator (PISO) algorithm [72] most of the time for transient problems, or the semi-implicit method for pressure-linked equations (SIMPLE) algorithm [18][98] for steady-state problems.

The PIMPLE algorithm available in OpenFOAM is used for transient problems and is a combination of the PISO and SIMPLE algorithms. All those algorithms solve a pressure equation to satisfy the mass conservation and provide a corrected velocity to satisfy the momentum equation. The PIMPLE algorithm starts by solving the momentum equation – this step is called the momentum predictor – and then enters the PISO loop, in which it iterates to solve for the pressure until a converged pressure solution is found. Once the PISO loop is complete, it assesses the convergence of the coupled velocity-pressure $p - \mathbf{u}$ solution and reiterates the PIMPLE loop if not converged (new momentum predictor). A flowchart of the PIMPLE process is shown in figure 2.7 in which Δt is the time step and t_{end} is the simulation end time.

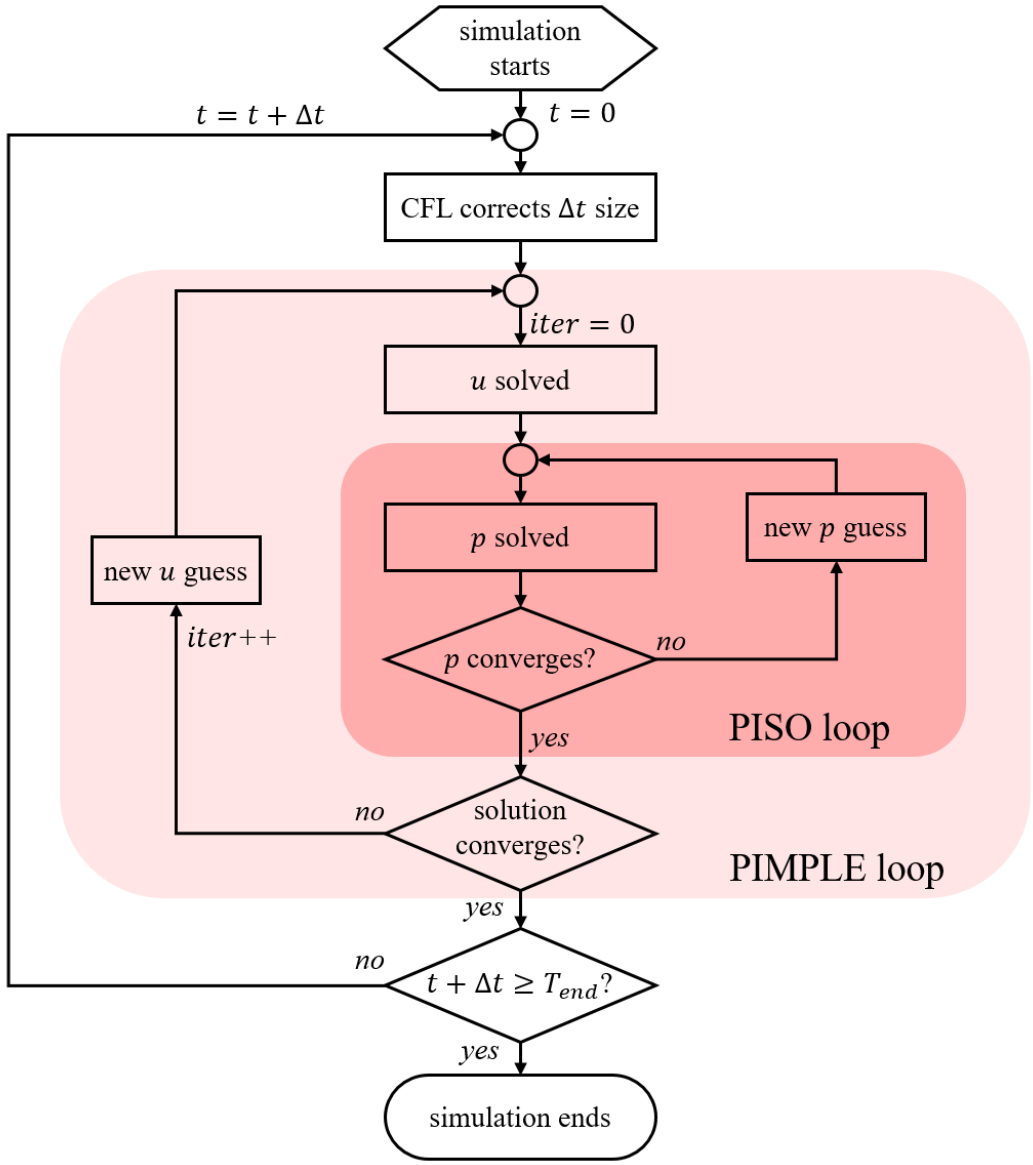


Figure 2.7 Flowchart of the PIMPLE algorithm process

When discretising the simple momentum equation for incompressible and inviscid flows:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \times \mathbf{u}) = -\boldsymbol{\nabla} p \tag{2.49}$$

One obtains the following system of equations using matricial form:

$$M[\mathbf{u}] = -\boldsymbol{\nabla} p \tag{2.50}$$

Where the matrix $M[\mathbf{u}]$ can by decomposed as the difference between a diagonal matrix A and an off-diagonal matrix $\mathbf{H}$ of the contributions:

$$A\mathbf{u} - \mathbf{H} = \boldsymbol{\nabla} p \tag{2.51}$$

Hence, one can solve for the velocity $\mathbf{u}$ using the following matrix system:

$$\boxed{\mathbf{u} = \frac{\mathbf{H}}{A} - \frac{1}{A}\boldsymbol{\nabla} p} \tag{2.52}$$

In order to solve for the pressure p, one can use the volumetric flux corrector equation obtained by interpolating the velocity to the faces of the discretisation control volume and applying the surface vector $\mathbf{S}_f$ such as:

$$\phi = \mathbf{u}_f \mathbf{S}_f = \left(\frac{\mathbf{H}}{A}\right)_f \cdot \mathbf{S}_f - \left(\frac{1}{A}\right)_f \mathbf{S}_f \cdot \boldsymbol{\nabla} p \tag{2.53}$$

Using the continuity equation $\nabla \cdot \phi = 0$ one finally obtains the following matrix system to solve for the pressure p:

$$\boxed{\nabla \cdot \left[\left(\frac{1}{A}\right)_f \boldsymbol{\nabla} p\right] = \nabla \cdot \left(\frac{\mathbf{H}}{A}\right)_f} \tag{2.54}$$

Several methods can be used to solve matrix systems.

The preconditioned conjugate gradient method

The preconditioned conjugate gradient (PCG) method [60] [150] is used as an iterative algorithm. Let us take the linear system of equations of unknowns $\mathbf{x}$:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{2.55}$$

where $\mathbf{A}$ is a known real, symmetric, positive-definite matrix of size $n \times n$ and $\mathbf{b}$ known too. Let us note $\mathbf{x}_*$ the solution of (2.55). The iterative method implies that $\mathbf{x}_*$ must also be the unique minimiser of the function f expressed as:

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x} - \mathbf{x}^\mathsf{T}\mathbf{b}, \qquad \mathbf{x} \in \mathbb{R}^n \tag{2.56}$$

Note $\mathbf{r}_k$ the residual at the k^{th} iteration step, such as $\mathbf{r}_k = \mathbf{A}\mathbf{x}_k$.

Note $\mathbf{p}$ the conjugate vectors such as:

$$\mathbf{p}_k = \mathbf{r}_k - \sum_{i<k} \frac{\mathbf{p}_i^\mathsf{T}\mathbf{A}\mathbf{r}_k}{\mathbf{p}_i^\mathsf{T}\mathbf{A}\mathbf{p}_i}\mathbf{p}_i \tag{2.57}$$

The solution is obtained in converging the solution given by the following linear combination:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \tag{2.58}$$

(note that the iteration process starts with a "guessed solution" $\mathbf{x}_0$) such as $\mathbf{p}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$

$$\boxed{\mathbf{x}_* = \sum_{i=1}^{n} \alpha_i \mathbf{p}_i} \tag{2.59}$$

where:

$$\alpha_k = \frac{\mathbf{p}_k^\mathsf{T}\mathbf{r}_k}{\mathbf{p}_k^\mathsf{T}\mathbf{A}\mathbf{p}_k} \tag{2.60}$$

The Gauss-Seidel method

The Gauss-Seidel method [45] is an iterative method for the resolution of linear systems. Let us take:

$$\mathbf{Ax} = \mathbf{b} \tag{2.61}$$

The matrix linear system to solve, where $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $\mathbf{b} \in \mathbb{R}^n$. We note a_{ij} the elements of $\mathbf{A}$ and b_i the elements of $\mathbf{b}$ such as:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \quad \text{et} \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \tag{2.62}$$

The algorithm assumes that diagonal of $\mathbf{A}$ is composed by non-zero numbers.

Note $\mathbf{x}^k = \left(x_1^k, ..., x_n^k\right) \in \mathbb{R}^n$ the current iterated solution.

The next iterated $\mathbf{x}^{k+1} = \left(x_1^{k+1}, ..., x_n^{k+1}\right) \in \mathbb{R}^n$ is calculated in n steps as follows:

1st step: Assuming that $a_{11} \neq 0$ and knowing $\left(x_2^{k+1}, ..., x_n^{k+1}\right)$, one can calculate x_1^{k+1} with the first equation of the linear system $\mathbf{Ax} = \mathbf{b}$. More specifically, x_1^{k+1} is taken as the solution of:

$$a_{11}x_1^{k+1} + a_{12}x_2^k + ... + a_{2n}x_n^k = b_1 \tag{2.63}$$

2^{nd} step: Assuming that $a_{22} \neq 0$ and knowing $\left(x_1^{k+1}, x_3^k, ..., x_n^k\right)$, one can calculate x_2^{k+1} with the second equation of the linear system $\mathbf{Ax} = \mathbf{b}$. More specifically, x_2^{k+1} is taken as the solution of:

$$a_{21}x_1^{k+1} + a_{22}x_2^{k+1} + a_{23}x_3^k + ... + a_{2n}x_n^k = b_2 \tag{2.64}$$

i^{th} step, $i \in [\![1, n]\!]$: Assuming that $a_{ii} \neq 0$ and knowing $\left(x_1^{k+1}, ..., x_{i-1}^{k+1}, x_{i+1}^k, ..., x_n^k\right)$, one can calculate x_i^{k+1} with the i^{th} equation of the linear system $\mathbf{Ax} = \mathbf{b}$. More specifically, x_i^{k+1} is taken as the solution of:

$$a_{i1}x_1^{k+1} + ... + a_{i,i-1}x_{i-1}^{k+1} + a_{ii}x_i^{k+1} + a_{i,i+1}x_{i+1}^k + ... + a_{in}x_n^k = b_i \tag{2.65}$$

Hence, assuming the diagonal components of $\mathbf{A}$ are not equal to 0, the general solution to calculate the components x_i^{k+1} of $\mathbf{x}^{k+1}$ for $i = 1, ..., n$ can be written:

$$\boxed{x_i^{k+1} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{k+1} - \sum_{j=i+1}^{n} a_{ij}x_j^k\right)} \tag{2.66}$$

The algorithm can be written under matrix form assuming $\mathbf{A}$ decomposes as follows:

$$\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U} \tag{2.67}$$

where $\mathbf{D}$ is the diagonal part of $\mathbf{A}$, $\mathbf{L}$ its strict lower triangular part and $\mathbf{U}$ its strict upper triangular part [49]. Hence, one iteration of the Gauss-Seidel method from x^k to x^{k+1} consists in solving the inferior triangular system:

$$\boxed{(\mathbf{L}+\mathbf{D})\mathbf{x}^{k+1} = b - Ux^k} \tag{2.68}$$

from top to bottom i.e. by successively determining $x_1^{k+1}, x_2^{k+1}, ..., x_n^{k+1}$.

2.4 Introduction to machine learning

Machine learning (ML) is a way to model phenomena in order to optimise modelling processes. It aims to synthesise the different variables that come at stake in the problem and enable the visualisation of the behaviours and correlations involved within the data. In data science, ML is the step where the computer tries to model the data we are feeding it with. In this case, "modelling" means representing the behaviour of the studied phenomenon. Supervised machine learning algorithms are generally used when the handled data is concrete and corresponds to physical quantities. In this dissertation, supervised machine learning methods were used as the relationships between flow quantities and how they affect transport equations are investigated.

2.4.1 Machine learning process and neural network

The ML process can roughly be decomposed into the following steps:

- First comes the problem definition phase in which we define the nature of the problem and the behaviour we want to model and which tools are available for the treatment of data and other technical requirements.
- Then comes the exploration phase in which we look at the data available, how it is structured, where it can be collected and which data will be the

most relevant to use. In this phase we also want to draw the form of the solution, decide on the architecture of the Neural Network (NN) and the language that will be used for the algorithm setup.

- The next step is the collection of the relevant data. In this step we want to collect as much data as needed to obtain the best ML model. The data must avoid redundancy so that our ML model can adapt to different cases efficiently. In ML, more quality data will enable the improvement of the model.

- The data is then pre-processed for the ML model. In this phase we need to ensure that the data that will be provided to the ML model is clean i.e. no irrelevant or aberrant data is present in the set, and normalised so that all the data the ML model takes at the same order of magnitude.

- Then comes the implementation of the ML model. In this phase the ML model is fed with input data and the desired corresponding solution for each given input. The solution is called the output. The ML model will be modelling the relationship between the inputs and the outputs. ML can use a parametric method in which the number of parameters can be fixed to find a fitting for the relationship. In this case we already assume the behaviour adopted by the modelled phenomenon, and can tell the ML the structure of the function it has to use. When we have no knowledge about how the outputs and inputs are linked, ML can use a non-parametric method in which a mapping function will be used to fit the data. For example, the k-nearest neighbours algorithm is a non-parametric method in which the algorithm will make predictions based on the k most similar trained patterns for each new input. This

method estimates the behaviour of the closest neighbours based on how one behaves near them.

- The training of the ML model comes next in which a comparison between each model predicted value and the output value is made. The ML improves by backpropagation within the neural network (NN) in which the ML model minimises the gap between the predicted and true values thanks to loss functions.

- Then the ML model is tested with a new dataset which solutions are known and its performance is evaluated.

- Finally the ML model is used to predict unknown solutions in new cases.

In this dissertation, machine learning models were implemented with supervised learning using feed-forward neural networks (FFNN). FFNN are flexible non-parametric regression models [108]. NN are systems of neurons that can be structured as layers such as the first layer is the input layer and the last layer is the output layer. Between the input and output are a number of hidden layers that will apply a function – an activation function – to the neurons of the previous layer and communicate the result of the application to the next layer until the output, which is the prediction of the NN. The input layer is constituted of neurons containing each an input feature vector $\mathbf{x} = (x_{l,1}, ..., x_{l,i}, ...x_{l,n_l})$ for a network of l layers with n_l neurons in the l^{th} layer and with $i \in [\![1, n_l]\!]$. In that same network, the values of the nodes of the l^{th} layer x_{l,n_l} are obtained by doing the weighted sum of the values of the $(l-1)^{\text{th}}$ layer i.e. $x_{l-1,n_{l-1}}$, such as:

$$x_{l,n_l} = a_{l,n_l}\left(w^0_{l,k_l} + \sum_{k=1}^{n_{l-1}} w^j_{l,k_l} \cdot x_{l-1,j}\right) \tag{2.69}$$

Where w^j is the weight applied to the j^{th} neuron of the layer l, $j \in [\![1, n_{l-1}]\!]$) and the additional term w^0 is the bias term. a_{l,n_l} is the activation function associated with the layer $l-1$. Some activation functions are commonly used in ML algorithms: the sigmoid $x \to 1/(1+e^{-x})$, the tanh function $x \to \tanh x$, and the ReLU function $x \to \max\{0, x\}$ because they are monotonic, non-linear and bounded. A representation of an example of FFNN is given in figure 2.8.

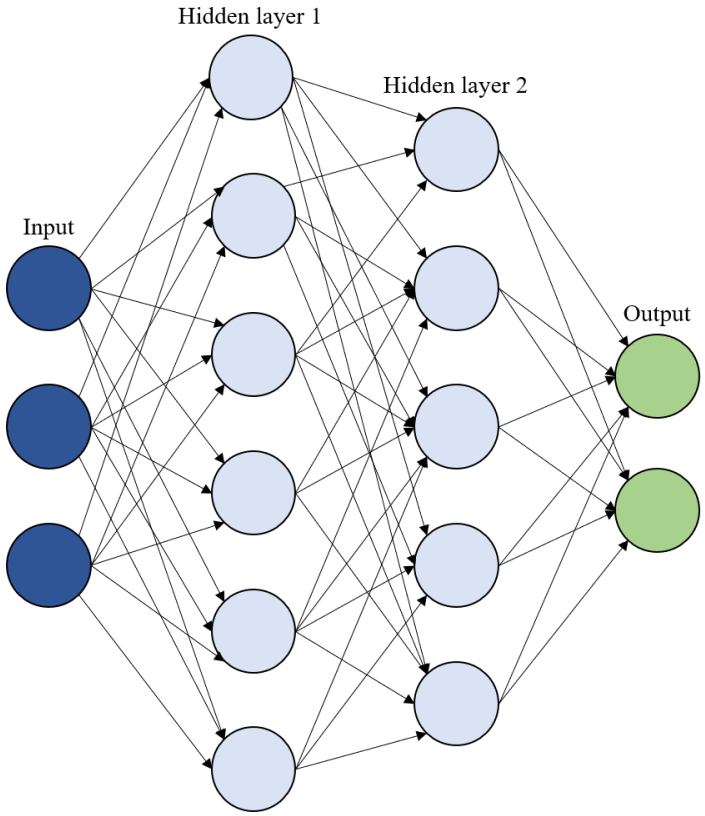


Figure 2.8 Representation of a feed-forward neural network containing three inputs, two hidden layers of six and five neurons and two outputs.

2.4.2 Data normalisation

The choice of the input features η_i is of great importance in the process of implementation of the ML model. They are features available for the training

of the model and later for its use in new cases. Ideally, those inputs must depict the characteristic of the data. For example, for stratified two-phase flows, it seems important to inform the ML model with features describing the position of each phase and where the high gradients of velocities are located to locate the interface and the walls. Geometry characteristics can also be used to add more precision on the environment of the flow. However, too many input features might reduce the ability of the ML model to adapt to new cases that are different from the training dataset cases. Too much data on a few particular cases could as well result in over-fitting, which negatively impacts the ML model performance in making accurate predictions in new cases. Choosing non-dimensional inputs also helps the model to perform well in new scenarios.

All input and output features of the machine learning model are normalised in order to maximise the chances of obtaining an effective model and to improve the training speed [69]. When manipulating different inputs with different orders of magnitude, normalisation helps put all the data in the same range of the activation functions, usually between 0 and 1. It allows for non-zero gradients during the training and thus, a faster learning. Especially since that in fluid dynamics, it is common to handle data with very disparate orders of magnitude, such as the kinematic viscosity of order of magnitude $\mathcal{O}(10^{-6})$ for water, or $\mathcal{O}(10^{8})$ for the specific turbulence dissipation rate in the air in shearing regions of a flow. Each input and output feature is normalised prior to the training phase using standardisation such as:

$$\widehat{X_i} = \frac{X_i - \mu_X}{\sigma_X} \tag{2.70}$$

where X_i is the quantity to scale in the vector $\mathbf{X}$, $\widehat{X_i}$ is the standardised value of X_i, μ_X is the mean of all the components of $\mathbf{X}$ and σ_X is its standard deviation. This operation is implemented along with the ML model and saved as "scaler" for the training data but later also for the testing data and the future predictions of the ML model.

2.4.3 Loss function and training quality

During the training phase, the ML model minimises the gap between its predicted solution and the true solution. In the context of this study, the gap to minimise is between the predicted solution and the high fidelity solution, that is denoted by the subscript *true*. For each cell of the domain, a set of inputs based on the flow features $\boldsymbol{\varphi}$ and domain characteristics and one output i.e. a correction term for the $k-\omega$ model equations, denoted as $\beta(\boldsymbol{\varphi})$, are given to the model. To minimise the gap between the predicted solution $\beta_{i,\text{pred}}$ and the true solution $\beta_{i,\text{true}}$, the ML model sets the parameters of the algorithm so that the sum $\mathcal{L}$ of the loss functions $L(\beta_{i,\text{true}}, \beta_{i,\text{pred}})$ calculated for each cell is minimised. In a domain of n cells using a squared loss function for example, the sum writes:

$$\mathcal{L} = \sum_{i=1}^{n} (\beta_{i,\text{true}} - \beta_{i,\text{pred}})^2 \tag{2.71}$$

As previously mentioned, one would aim for a model accuracy that allows for good predictions in agreement with the true solutions. Too much accuracy in the model training could lead to the over-fitting of the training data, making the model irrelevant for predictions of unseen data. The model must remain capable of adapting to cases that differ from the training cases. There is no

rule to know in advance which training accuracy is optimal and tests with different levels of accuracy are generally carried out from one case to another. The accuracy of the model during the training phase is calculated using the coefficient of determination or "R^2 score". Considering a set of n true values: $\boldsymbol{\beta}_{\text{true}} = \{\beta_{1,true}, \beta_{2,true}, ..., \beta_{n,true}\}$, and its set of predicted solutions by the ML model: $\boldsymbol{\beta}_{\text{pred}} = \{\beta_{1,pred}, \beta_{2,pred}, ..., \beta_{n,pred}\}$, one can write the R^2 score as follows:

$$R^2 = 1 - \frac{\mathcal{L}}{\sum_{i=1}^{n} (\beta_{i,true} - \bar{\boldsymbol{\beta}}_{true})^2} \tag{2.72}$$

One can see R^2 as the model error or loss function divided by the basic model error that always predicts the mean of the predicted quantity. The R^2 becomes higher when increasing the model accuracy and its maximum is 1 or 100% when all the predictions are exact. There is no minimum score but a model that always predict the mean value will reach a score of 0, meaning that a model presenting a negative R^2 score is worse than a model that would always predict the mean value.

The gradient descent algorithm

The gradient descent is a method in which the minimum value of a function is searched in an iterative algorithm. It is widely used in machine learning in order to minimise the loss function and the optimal parameters for the model. Every iteration, the algorithm calculates the next point of the regression using the gradient at the current position of the iteration. Then the gradient is scaled by a parameter, λ, the learning rate, and substracted from the current position. During the training of the ML model, one seeks to minimise the loss function

$\mathcal{L}$ and update the weight w_t at the time t. Thus, the gradient descent process applied to the linear regression of a ML model can translate to:

$$w^{j+1} = w_t - \lambda \frac{\partial \mathcal{L}}{\partial w} \tag{2.73}$$

The learning rate must be chosen carefully as it influences the performance of the ML training process. If λ is set too small, the algorithm would take a longer time to converge, however, if λ is set too large, the gradient descent might not converge at all to the optimal minimum and may diverge.

In order to optimise the gradient descent algorithm, one can use optimisation algorithms that will make the algorithm converge faster. The Adam optimisation algorithm [77] has proven to be one of the most efficient extensions of the gradient descent algorithm to be used in machine learning. It combines the gradient descent with the momentum algorithm and the root mean square propagation (RMSprop) algorithm. The momentum gradient [104] boosts the algorithm by applying an exponentially weighted average of the calculated gradients and writes:

$$w_{t+1} = w_t - \eta m_t \tag{2.74}$$

where m_t is the aggregate of gradients and writes:

$$m_t = \gamma m_{t-1} + (1 - \gamma) \frac{\partial \mathcal{L}}{\partial w_t} \tag{2.75}$$

in which γ is the moving average parameter or momentum term taken at 0.9. The RMSprop algorithm takes an exponential moving average and can be described as:

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t + \epsilon}} \frac{\partial \mathcal{L}}{\partial w_t} \tag{2.76}$$

where ϵ is a strictly positive and small constant (10^{-8}) to avoid division by zero, and where v_t is the sum of the squares of the previous gradients and writes:

$$v_t = \gamma v_{t-1} + (1 - \gamma) \left(\frac{\partial \mathcal{L}}{\partial w_t} \right)^2 \tag{2.77}$$

Therefore, from equations 2.75 and 2.77, one can write:

$$\begin{cases} m_t = \gamma_1 m_{t-1} + (1 - \gamma_1) \dfrac{\partial \mathcal{L}}{\partial w_t} \\ v_t = \gamma_2 v_{t-1} + (1 - \gamma_2) \left(\dfrac{\partial \mathcal{L}}{\partial w_t} \right)^2 \end{cases} \tag{2.78}$$

where $\gamma_1 = 0.9$ and $\gamma_2 = 0.999$ as suggested by the developers of the Adam algorithm [77] and are called decay rates. It was observed that m_t and v_t are biased towards zero as they are initialised as zero vectors, more specifically during the first time steps of the process. This is accounted for by computing bias-corrected values for m_t and v_t respectively noted $\widehat{m_t}$ and $\widehat{v_t}$ and write:

$$\begin{cases} \widehat{m_t} = \dfrac{m_t}{1 - \gamma_1} \\ \widehat{v_t} = \dfrac{v_t}{1 - \gamma_2} \end{cases} \tag{2.79}$$

Finally, the Adam optimiser can be described as follows:

$$\boxed{w_{t+1} = w_t - \eta \frac{\widehat{m_t}}{\widehat{v_t}}} \tag{2.80}$$

2.5 Remarks

In this second chapter, the numerical methods employed for the numerical work carried out in this thesis were introduced. More particularly, the averaged CFD RANS $k-\omega$ model was presented. Note that the $k-\omega$ SST model was not employed in this work because the complexity to implement appropriate corrections in comparison to the standard Wilcox's model. It will be seen that very satisfying results were obtained with this model. Moreover, the two multiphase methods available in OpenFOAM namely the VOF and Euler-Euler methods were introduced, with an emphasis on the fact that it is expected from the VOF method to perform much better than the Euler-Euler method as the latter one was not designed for two-phase stratified flows. Finally, the mesh requirements for the DNS, qDNS and LES simulations were given. The choice of qDNS over LES will be justified in chapter 4.

Chapter 3

Literature review: two-phase shear flows and ML applications in CFD

In this chapter, a state of the art on two-phase shear flow is presented in section 3.1 and in particular on stratified two-phase shear flows that one can expect observe in aero-engine bearing chambers. Additionally, some of the existing work on the use of machine learning methods in computational fluid dynamics is examined in section 3.2.

3.1 Existing work on two-phase shear-flows

As stated in the Introduction chapter, one of the objectives of this dissertation was to investigate the behaviour of stratified shear flows in order to provide an appropriate correction for the interfacial turbulence intensity in standard averaged RANS models. In this section, some existing experiments on two-phase shear flows that are later used at a reference and base for the simulations

carried out in this dissertation are presented. Existing work on two-phase flow simulations is also presented in a second part in order to make note of the current existing methods and understand their limits.

3.1.1 Experimental work

Two-phase flows can be encountered everywhere in the nature (rivers, clouds, waterfalls, ocean, etc.), in the engineering industry (nuclear engineering, aerospace, chemical engineering) and show a variety of patterns. When studying gas-liquid flows, one can observe many phase configurations depending on the flow rate of each phase. One can describe those configurations based on visual examination, most of the time being a subjective and qualitative assessment [5, 4]. The description of those configurations takes into account the geometrical structure of the interfaces (bubble, plug, slug, stratified, annular, etc.) and the characteristic dimensions of the flow (bubble size, pocket size, interface height or film thickness, etc.). Co-current horizontal gas-liquid flows have been classified into a few main configurations: bubbly, smooth stratified, wavy stratified, slug and annular [27]. Some of the main two-phase flow configurations can be seen in figure 2.1, chapter 2 and in figure 3.1 for co-current and horizontal gas-liquid flows. Similar patterns were observed for immiscible liquid-liquid two-phase flows: experiments were conducted for oil-water flows for instance [1].

Many experiments have been conducted to better understand two-phase flow physics and patterns, and improve the existing numerical models available for two-phase flows. Krepper et al. [79] studied the bubble rising in a vertical water column that are commonly seen in nuclear and chemical reactors. Air-water flow around an obstacle in a vertical column was experimentally investigated by Prasser et al. [103] in order to develop models for bubble coalescence and

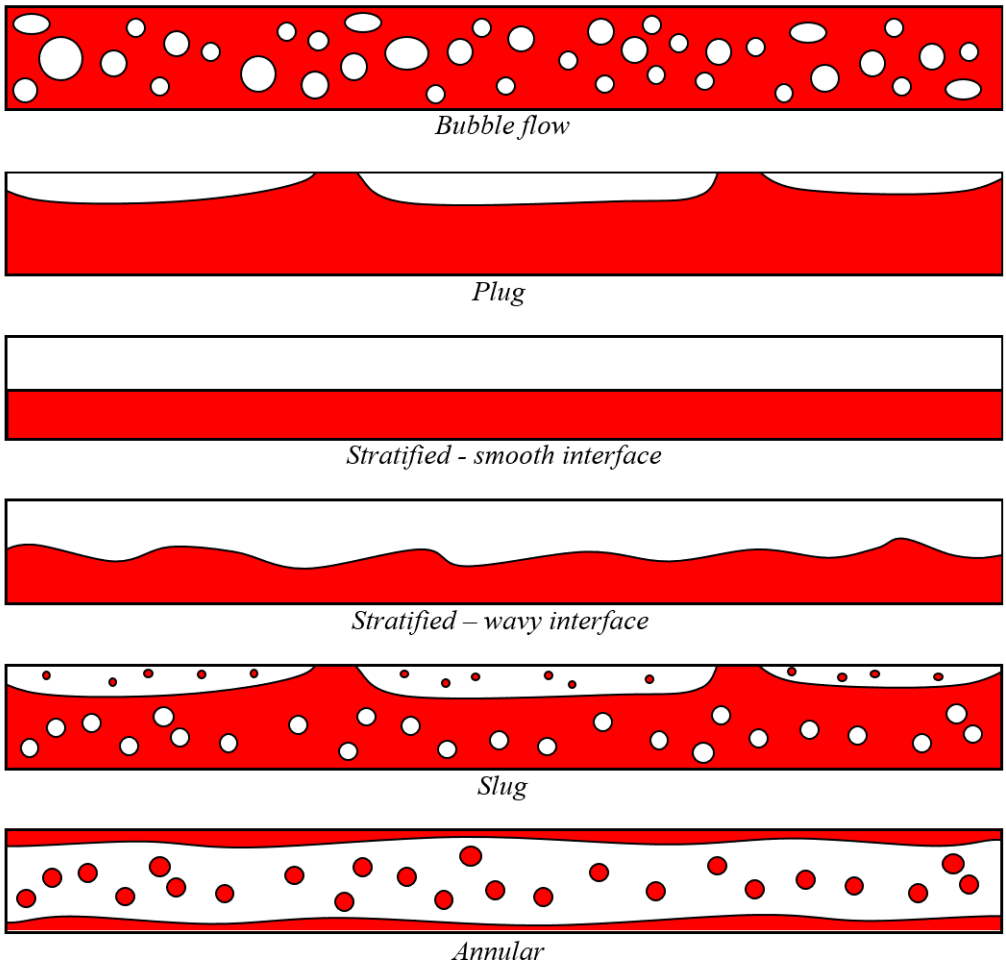


Figure 3.1 Main gas-liquid co-current flow configurations in channels with liquid in red and gas in white

fragmentation in gas-liquid two-phase flows. Stäbler et al. [125] carried out experiments for counter-current two-phase gas-liquid flows in a horizontal channel and their experiments were used as a reference for many numerical works [152][9][130][101]. More recently, Gordino et al. [48] conducted experiments on air injection in a vertical water column in order to assess blending models' performances of two-fluid models, under unsteady flow conditions while former experimental research focused on steady-state results.

Experiments also serve the purpose of validation for the improvement of CFD models. Once more in this dissertation, most of the presented numerical results are based on existing experimental results obtained in different gas-liquid stratified flow configurations in horizontal channels. A particular care is given to the investigation of the interfacial shear forces and turbulence. Stratified flows are characterised by the shape of the interface between the liquid and gaseous phases. The interface can be defined as smooth or wavy depending on the flow regime. A smooth interface is usually observed at low gas speeds, while two and three-dimensional waves appear at higher speeds [84]. Smooth interfaces are observed when the gas velocity is not too large and the gravity and surface tension maintains the interface sharp and non-wavy. Experiments on co-current flows of air and liquid have shown that the gaseous phase transmits its energy to the waves via shear stress forces and pressure forces [22]. The formation of two-dimension waves with small amplitudes and large wavelengths was experimentally observed for low-viscosity liquid phases induced by air [74]. Those waves can be called the J-waves. It was also shown that increasing the velocity led to the formation of two-dimension waves with larger amplitudes and shorter wavelengths [3]. One talks about the KH-waves as a reference to the Kevin-Helmholtz instabilities occurring at similar flow conditions. At even higher gas velocities, steeper waves are observed and are known for breaking at large enough amplitude. They are called roll waves and their wavelength and amplitude is not constant [145][126]. Hashmi [59] performed experiments on stratified flow in co- and counter-current configurations of air-water flows and for very small levels of liquid (thin film flow). They calculated the transferred momentum to the film and investigated the correlation between momentum transfer and flow rate by carrying out many flow conditions from smooth interface states to very wavy and three-dimensional

interface states in order to develop an enhanced version of the VOF method for wavy films. More detail on their numerical work is provided later in section 3.1.2. They observed droplet shedding for reasonably high superficial gas velocities in the co-current flow configuration due to shear forces entraining the liquid film. In the counter-current configuration, only a small gas velocity triggered droplet shedding. The droplet shedding state is out of the scope of this thesis and only smooth and wavy sharp interface states are investigated. Wintterle et al. [152] conducted experiments on counter-current flow of air and water in order to establish statistical correlation models between the phase volume fraction and the turbulent kinetic energy calculated from the velocity fluctuations. With this correlation, they highlighted the important impact of the turbulent and potential energy of the liquid phase on the wavy interface region. They implemented the derived differential equation obtained from this correlation into a CFD code in order to inform interaction numerical models for stratified flows. They obtained satisfying results, in good conformity with the experiments. The approach developed by Wintterle et al. [152] shows that informing the interfacial turbulence of CFD models with experimental results is a promising and operational method. More details on their numerical work is provided in section 3.1.2

Reference experiments on stratified two-phase flows in horizontal channels

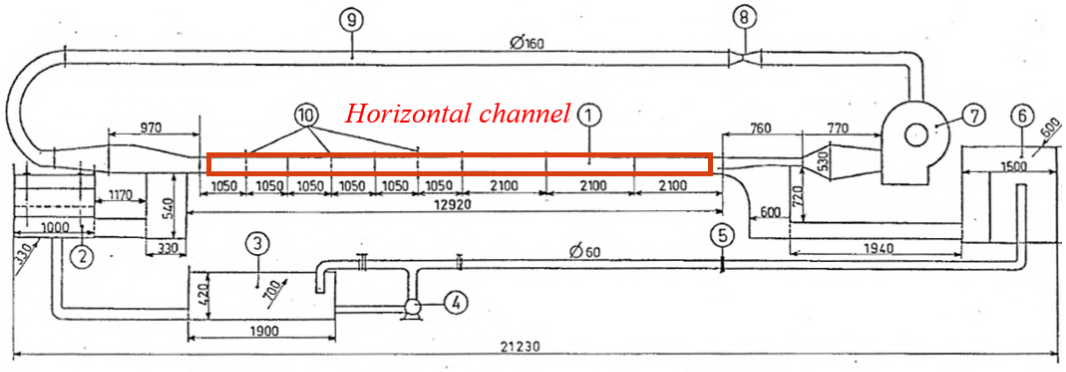
The two-phase flow simulations presented in this dissertation can be split in two configuration groups that can be described as a "deep water" or "thick-film" configuration and a "shallow water" or "thin film" configuration. The latter being being the configuration that comes closest to the oil film found in aero-

engine bearing chambers.

Thick-film experiments:

The simulations of the first deep water, thick-film configuration were based on the experiments of Fabre et al. [35]. Their work was used to perform the preliminary studies with OpenFOAM for the choice of multiphase solver and discretisation schemes, and later for the proof of concept of the dissertation. Fabre et al. [35] carried out laser Doppler anemometry (LDA) measurements on stratified flows in a 12 m long, 0.1 m high, and 0.2 m wide closed horizontal and rectangular channel. The experimental setup is shown in figure 3.2. The gaseous phase used in their experiments was air and the liquid phase was water. In this 'deep water' configuration, the mean liquid thickness represents between 22% and 38.0% of the channel height. The researchers studied three flow regimes namely 'RUN 250', '400' and '600' in which the gaseous phase velocity was set at least 9 times larger than the liquid phase velocity and up to 11 times larger. The difference in phase velocity resulted in the generation of high shear forces across the interface, inducing its waviness. The researchers aimed to investigate the interfacial transfers between the two phases that are increased by the generation of interfacial turbulence caused by the velocity gradients between the two phases. They measured the mean velocity, turbulent kinetic energy and Reynolds stress vertically in the centre of the channel in both phases.

Figure 3.3 shows the mean velocity profiles they obtained with $\overline{U}_G^+$ and $\overline{U}_L^+$ being the mean axial velocity normalised by the friction velocity in the gaseous phase and in the liquid phase respectively. η_G and η_L are the vertical coordinates normalised by the water depth. Those experiments have been largely used



1 – Rectangular channel 240 × 140 mm²
2 – Constant level tank
3 – Water tank
4 – Centrifugal pump
5 – Orifice meter
6 – Tranquilisation tank
7 – Centrifugal tank
8 – Venturi meter
9 – Air loop
10 – Measuring test section

Figure 3.2 General scheme of the experiments of Fabre et al. [35]

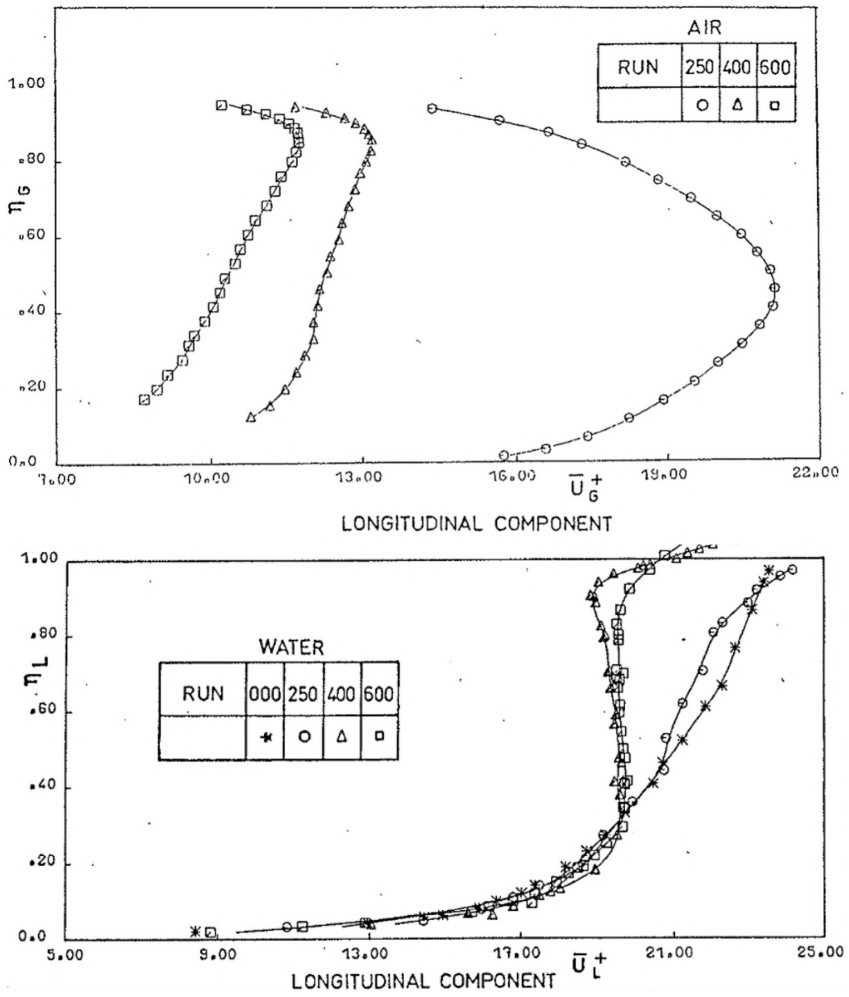


Figure 3.3 Mean axial velocity profiles measured in [35]

by many researchers as reference for their numerical results [41][29][37]. Fabre et al. [35] observed the generation of secondary flows in both phases for the two highest flow regimes and underlined the difficulty to realise measurements in the interfacial region especially for the waviest cases. The case 'RUN 250' is presented more in depth in chapter 4 and has served as the reference for the previously mentioned preliminary study (chapter 4) with OpenFOAM and for the proof of concept (chapter 5).

Thin-film experiments:

The high-fidelity simulations performed in order to train the machine learning models and the test cases using the coupled RANS-ML model presented in this dissertation were based on the work of Hann et al [54] and Kim et al. [76], who developed and carried out particle image velocimetry (PIV) measurement techniques on stratified flows in a shallow water, thin-film configuration in a closed horizontal and rectangular channel, which was 2 m long, 0.026 m high and 0.166 m wide. Their experimental setup is presented in figure 3.4. Similarly

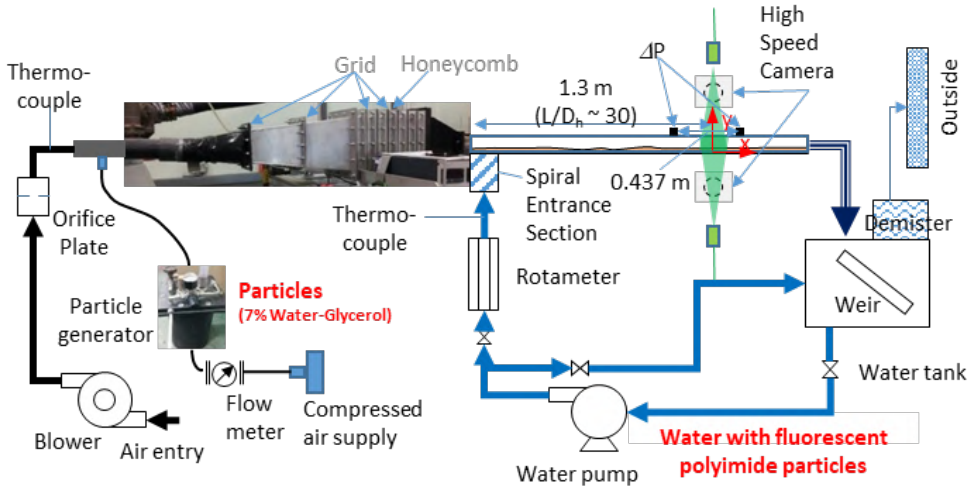


Figure 3.4 General scheme of the experiments of Kim et al. [76]

to Fabre et al. experiments [35], the researchers have used water for the liquid phase and air for the gas. In this thin-film configuration, the film thickness was comprised between 9% and 24% of the channel height. A range of 23 cases was investigated by Kim et al. [76] with a gas velocity of at least 58 times larger than the liquid velocity and up to 722 times larger. Most of the cases investigated by the researchers were wavy in two or three dimensional manner. As opposed to Fabre et al. experiments [35], the researchers have examined regimes for which the liquid phase is not turbulent. The 2 or 3 dimensionality of the interface was mainly due to an entrainment of the liquid phase by the gaseous phase. Most of the measurements made by Kim et al. [76] focused on the in-

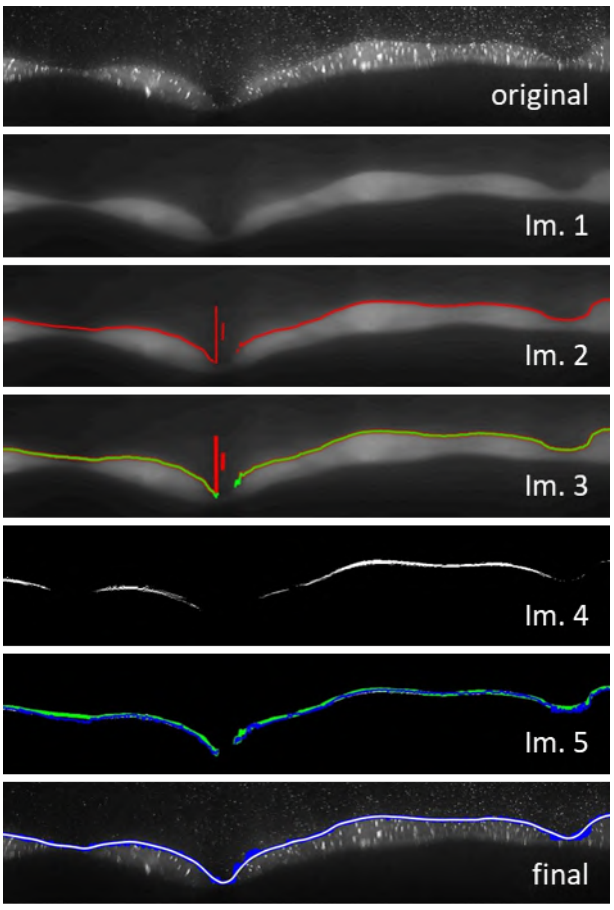


Figure 3.5 Image processing procedure for the detection of the interfaces developed by Hann et al. [54] and performed in [76]

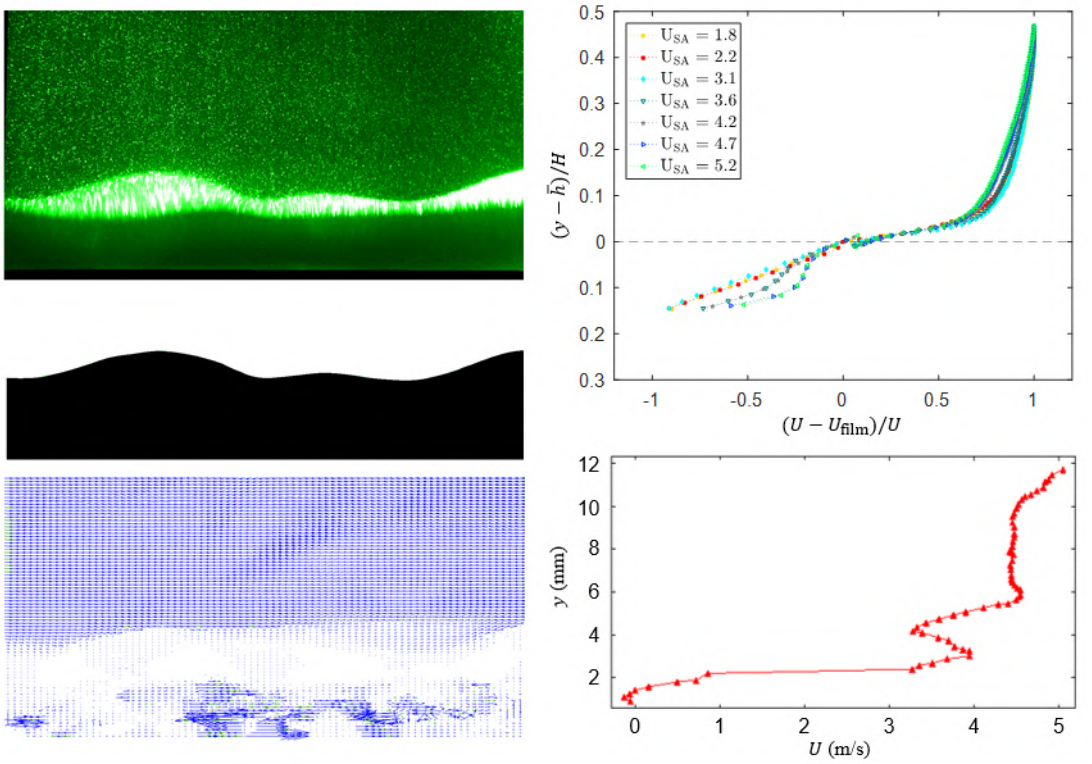


Figure 3.6 Kim et al. [76] raw PIV image of the air flow field (top left), velocity vector field obtained using the adaptive PIV algorithm (bottom left), corresponding mask (centre left), averaged velocity profiles for different flow regimes probed vertically in the channel (top right), and instantaneous velocity profile (bottom right). Superficial velocity: 3.6 m/s (air), 0.019 m/s (water).

ferior part of the gaseous phase with a special care given to the interfacial region.

Hann et al. [54] and Kim et al. [76] developed PIV measurement techniques for the detection of the interfaces (c.f. figure 3.6), and simultaneous time-resolved PIV measurements in both of the phases of the thin-film flow (c.f. figure 3.6). The researchers managed to perform the velocity measurements in the two phases simultaneously by splitting the laser beam and illuminating the channel from the two sides. They highlighted the presence of clear vortex structures in the gaseous turbulent flow. Kim et al. [76] also demonstrated a strong influence of the waviness of the interface on the root mean square of

fluctuation velocity profiles. Nearly constant profiles were obtained for smooth interface flow conditions, while the same profiles showed important variations across the gaseous phase for wavy interface flow conditions.

3.1.2 Numerical work

While experiments have continuously provided validation for CFD models in many applications, CFD always offers new challenges when it comes to improving the modelling and especially in the field of two-phase flow simulation. Some of the initial work on predicting two-phase stratified flows investigated the correlations in the shear stresses in two-phase flows from the known correlations existing in single-phase flows based on the mean velocity [111]. Empirical correlation for wavy interfaces' shear-stress in stratified flows were proposed by researchers in the late 90's [2, 4]. When it comes to numerically modelling stratified flows, CFD struggles to model the sharp interface between the gas and the liquid. The sharp interface is characteristic of strong velocity discontinuities between the two phases and RANS models over-predict the turbulent kinetic energy in the interfacial area resulting in poor predictions of the momentum transfer between the phases [84, 41]. An important factor is that RANS turbulence models commonly used for two-phase flows are identical to the ones used for single phase flows and this can lead to the poor prediction of the interfacial turbulence. The first model to treat stratified flows was proposed by Taitel and Dukler [129] for low-viscosity liquids. Their two-fluid model assumed a smooth interface in which the interfacial stress was underestimated, which is inconsistent with the interfacial friction increasing with the gaseous Reynolds number. More recent models proposed to use a single-phase flow approach in which the interface in characterised by a roughness from the gas perspective [34]. However predicting this interfacial

roughness is another difficult problem to address, as it depends on many flow characteristics such as the phase velocities and the fluctuations of the liquid height [23]. Mouza et al. [94] also highlighted the fact that this roughness methodology cannot be used to model the dynamic interactions between the phases that are responsible of the generation, expansion and propagation of the waves, especially for cases presenting large secondary flows as the ones observed in the experiments of Fabre et al. [35] in a rectangular channel at high velocities.

The difficulty to correctly predict the turbulent kinetic energy near the interface led to the development of new numerical models. In the early 2000's, approaches were based on the smooth interface theory in which one assumes that the interface may be represented as a solid wall moving with the shear velocity [13]. The small disturbances occurring in these types of flow are accounted for using the previously mentioned interface roughness. The idea that the gaseous phase of a stratified flow sees the much heavier liquid phase as a solid wall at the interface was widely used in the most recent numerical models [41]. Egorov et al. [32] developed an approach using this assumption and in which they suggested a wall-like treatment of the interface in the gas by adding damping of the turbulence in the interfacial area. The Egorov approach has been widely used to perform RANS simulations and showed significant improvements in the prediction of the velocity field near the interface [131] and even agreeing with experimental results [141, 64]. Some issues emerge from the way the Egorov approach is implemented in standard RANS models. One of them is the fact that the damping of the interfacial turbulence rests on a parameter B called the turbulence damping parameter. No guidelines exist for the choice of its value even though Egorov et al. [32] suggested that B should be set higher than 10. Numerical works have shown that much larger values

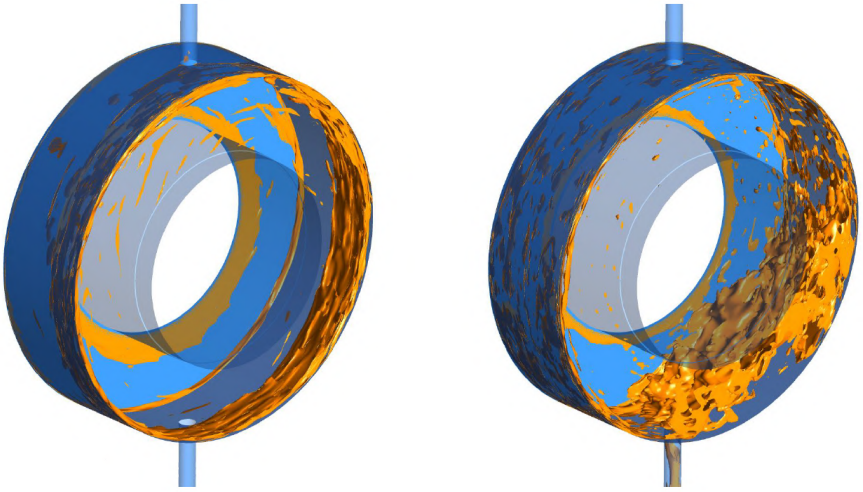


Figure 3.7 Oil film thickness distribution in an aero-engine's bearing chamber by Bristot et al. [15] for $B = 100$ (left) and $B = 10$ (right)

of B gave consistent results [86]. Furthermore, the choice of the coefficient greatly impacts the predictions of the flow as shown by Bristot et al. [15]. They emphasised on the fact that the choice of B is particularly difficult when the expected flow conditions are not known. Figure 3.7 shows the impact of the choice of the turbulence damping parameter in an air-oil flow in a simplified version of an aero-engine's bearing chamber rotating with the RANS $k - \omega$ shear stress transport (SST) turbulence model and a VOF solver and for the values $B = 10$ and $B = 100$.

It was also observed that the models using the Egorov approach are mesh dependent [125, 44, 37]. Moreover, those models treat the two-phases symmetrically [37, 41]. This causes the liquid phase to also see the interface with the gaseous phase as a non-slip wall boundary condition, while it should rather be seen as a slip-wall boundary condition instead.

In order to damp the interfacial turbulence, Egorov et al. [32] suggested to reduce the Wilcox standard $k - \omega$ turbulence model's (see system of equations

2.26) eddy viscosity (see equation 2.29) by taking large values of ω and to apply this treatment in the interface area only. Hence Egorov et al. [32] proposed to add a source term similar to the destruction term $\beta\rho\omega^2$ in the specific dissipation rate equation of the system 2.26. The following source term correction was proposed:

$$S_{\omega,i} = A_i \Delta x_i \beta \rho_i \left(B \frac{6\mu_i}{\beta \rho_i \Delta x_i^2} \right)^2 \tag{3.1}$$

where the subscript i denotes the phase meaning that there is one source term per phase i. A_i is the interfacial area density used as a switching term to ensure that the source term is activated in the interfacial region only. A_i is indeed zero in regions where only one phase is present and $A_i = |\nabla \alpha_i|$. Δx_i is the cell size across the interface and in the direction normal to the interface. As the interface is assumed to behave like a wall in smooth interface theory, the wall distance y^+ must be taken into account in order to choose cell sizes in the interfacial area. Thus, usually the cell size across the interface writes $\Delta x_i = \Delta y$. B is the damping parameter that Egorov et al. recommended to be higher than 10. However, it was shown that A_i is mesh dependent implying that B must be chosen with respect to the refinement of the mesh [86].

Frederix et al. [41] showed the impact of the over-production of the interfacial turbulence in the standard RANS $k - \varepsilon$ and $k - \omega$ models using the Euler-Euler method [70] in a shear-driven flow in a channel containing water and air and based on the 'RUN 250' case (smooth interface flow regime) from the experiments of Fabre et al. [35] described in the previous section. Frederix et al. [41] used the Egorov correction to develop modified versions of the RANS

$k-\omega$ and $k-\varepsilon$ turbulence models for cases in which the smooth interface assumption could be applied. Figure 3.8 shows the predictions of the mean axial velocity profiles u_x using the modified RANS models, highlighting the difference with the standard models. The interface between the two phases is illustrated by a strong discontinuity in the axial velocity profile as seen on the same figure. The numerical curves of the mean axial velocity predicted by the standard models are shifted upwards indicating an over-estimation of the turbulence in the gaseous phase above the interface. Frederix et al. [41] obtained very good agreement with the experiments using the Egorov approach in the models they implemented. However, the improved Egorov method developed by Frederix et al. holds some of the downsides of the standard Egorov approach. The symmetry of the model based on the fact that both phases receive the same interface treatment has yet to be addressed. The researchers also showed that the addition of the interfacial turbulence damping term led to significant under-prediction of the turbulent kinetic energy in the interfacial region.

Numerical models developed for the simulation of stratified shear flows should therefore take into account a wall-like damping of the interfacial turbulence. Reboux et al. [106] employed LES to model a gas-liquid flow following the DNS study carried out by Fulgosi et al [43]. They developed a corrected Smagorinsky subgrid model integrating a near-interface damping DNS-based function and a variational multiscale (VMS)-based approach that was initially introduced by Hughes et al. [66]. They obtained promising results in modelling turbulent shear flows, especially with the VMS based approach.

Recent developments on improving the Egorov method were proposed and more particularly to adress the previously mentioned issues concerning the

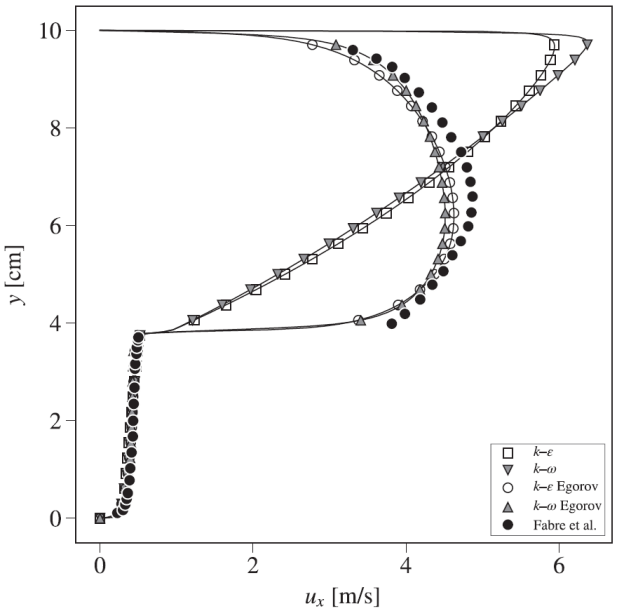


Figure 3.8 Mixture velocity profiles of the corrected and standard RANS models obtained by Frederix et al. [41] compared with Fabre et al. experiments [35]

lack of guidelines on the choice of B, the symmetrical treatment of the two phases and the mesh dependency. Polansky and Schmelter [99] implemented the Egorov method in OpenFOAM in the $k-\omega$ model and compared results with Fabre et al. experiments. They proposed a novel method to choose B by considering the pressure drop in a scheme. Fan et al. [37] proposed a novel approach with more physics and made the Egorov method mesh independent with an asymmetric treatment of the two phases, leading to more physical predictions of the flow turbulent kinetic energy and velocity especially in the interfacial region.

Wintterle et al [152] developed a phase interaction model to simulate stratified flows and tested their method using the Prandtl mixing length scale approach and an extended RANS $k-\omega$ turbulence model containing an additional term to improve the turbulence behaviour in the interfacial region. Their derived their method from a statistical approach that correlated the

turbulent kinetic energy of the liquid phase within the phase volume fraction distribution in experiments on counter-current gas-liquid flow. The researchers showed the considerable impact of the turbulent energy of the liquid on the interfacial region [153]. They obtained numerical predictions in agreement with the experiments.

Hashmi [58] investigated thin-film flow modelling in the context of the thermal management of bearing chambers in which oil fire and coking must be prevented. The researcher aimed to effectively quantify the oil film dynamics for various operating conditions and temperatures and performed experimental work on co-current and counter-current stratified gas-liquid flow in a horizontal channel in order to better understand the mechanisms of thick-film flows. Hashmi et al. [59] developed an 'enhanced VOF model' with a refined interface treatment and obtained promising results when applying their method on stratified flow cases using the RANS $k-\varepsilon$ model. Their methodology was based on an alteration of the turbulent viscosity in the vicinity of the gas-liquid interface. This was achieved by adding locally turbulence production in the dissipation ε or specific dissipation rate ω transport equation. They successfully tested their method against the experiments of Fabre et al. [35] on stratified gas-liquid flow in co-current configurations.

Fulgosi et al. [43] carried out Direct Numerical Simulations (DNS) of a countercurrent air-water flow in order to investigate the behaviour of the interface and more specifically the turbulence in the interfacial area. Results showed that the lighter phase can indeed see the interface like a deformable solid wall in the limit of non-breaking waves in numerous tested flow conditions. When fields were time-averaged, they observed that the turbulent fluctuating

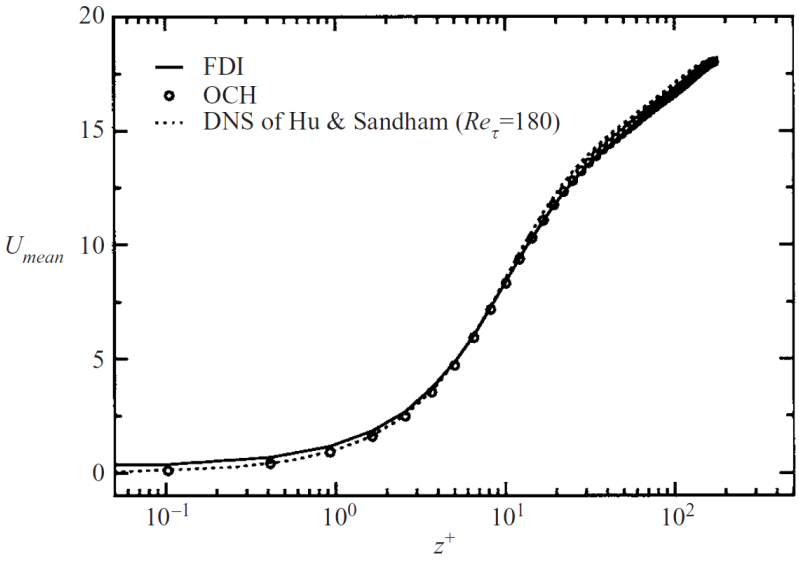


Figure 3.9 Mean streamwise velocity profile above the interface obtained by Fulgosi et al. [43]

field was indeed damped in the interfacial area, which conducts to an increase in the interfacial dissipation. Figure 3.9 shows the mean streamwise velocity profiles obtained by the DNS of Fulgosi et al. [43] (namely FDI) compared with experimental measurements (namely OCH) and another DNS study [65]. Results indicate that there is a clear similarity between wall turbulence and interfacial turbulence as it can be compared to the law of the wall previously described in figure 2.6. Fulgosi et al. [43] affirm that differences exist but are very subtle. This confirms the assumption that the lighter phase sees the heavier phase and must be taken into account in the methods developed for the treatment of interfaces.

Marschall [89] proposed a segregated model for the simulation of co-current two-phase liquid-liquid flows in a horizontal channel. The approach developed by the researcher is a multi-scale two-fluid model allowing for the capture of under-resolved flow structures and remain scale-consistent with DNS simula-

tions performed with two-fluid methods. Very good agreement were obtained with the analytical solution of the fluid velocity profile.

Godino et al. [48] reviewed the performances of the Eulerian two-fluid model for the prediction of dispersed and segregated two-phase flows investigated by researchers who conducted experiments to obtain validation of their numerical results. The previously mentioned researchers Krepper et al. [79] reproduced their experiments using different numerical methods to better model bubble rising cases, including the Euler-Euler method with the RANS $k-\omega$ SST model and concluded that their model was improvable due to errors caused by gas flow instabilities. Prasser et al. [103] also used the two-fluid method along with the $k-\omega$ SST model to reproduce their experiments on an air-water flow around an obstacle and found good agreements except for an overestimation of the phase volume fraction in the region located behind the obstacle. Porombka and Höhne [101] reproduced Stäbler et al. [125] experiments on counter-current two-phase gas-liquid flows with an improved free-surface drag model based on local shear stress with an interfacial damping function in the $k-\omega$ model similar to the Egorov's approach and achieved satisfactory quantitative agreements. Tekavcic et al. [130] also reproduced the experiments of Stäbler et al. using URANS simulations with the $k-\omega$ SST turbulence model and accounted for the interfacial turbulence damping by implementation of the method developed by Federix et al. [41] based on the Egorov's approach. The researchers demonstrated that an asymmetric treatment of the damping approach with damping applied only to the gas phase, significantly improves turbulent kinetic energy predictions.

The research presented in this dissertation aims to find an alternative method to the Egorov correction for RANS models using a similar foundation that is the addition of a source term to re-balance the budget of the transport equation of the specific turbulence dissipation rate. The new approach should also be case, parameter and mesh independent, unlike prior research.

3.2 Existing applications of machine learning in CFD

With the increasing efficiency of flow modelling methods, it has become more and more common to reach excellent accuracy using CFD simulations in the past decade, provided that computational resources are not an issue. However, the most realistic and precise results are usually obtained when the smallest scales of the turbulence are resolved. It has previously seen that solving those scales needs very high grid resolution and thus is computationally too expensive. Therefore, alternative methods are employed to make up for this deficit in resolution and machine learning has proven to be a useful tool to aid CFD when it comes to increase computation speeds, enhance simulation accuracy, or even improve simulation post-processing [78, 144, 50, 140, 121]. Artificial intelligence has known a growth of success for numerous of applications in science and technology. Neural networks (NN) have been the most used machine learning method in the context of CFD such as feed forward neural networks (FFNN), convolution neural networks (CNN) or physics informed neural networks (PINN). Those neural networks have been particularly used in CFD for multidimensional regression tools to predict the temporal evolution of specific aero- or hydrodynamics properties from experimental data, or in order to allow for the reconstruction of specific flow fields from CFD simulation results.

More particularly, encouraging developments in machine learning techniques to assist CFD for turbomachinery applications have been implemented recently over the past few years. They aim to carry out more robust and faster design analyses enabling improved efficiencies and cost reductions, as reviewed by Hammond et al. [53].

In the context of this thesis, we are particularly interested in ML techniques that could be used to improve existing averaged turbulence models, such as RANS models. Literature has shown promising work on data-driven RANS models to produce more accurate results [73, 154, 155]. Experimental and high-fidelity data, such as DNS or LES data, can be employed to drive RANS models. With the aim to train a ML model to inform a RANS model, high-fidelity CFD data seems ideal as it can potentially provide any needed flow information in any location of the domain, where experiments might struggle to gather data. Therefore, using high-fidelity simulations instead of experiments to train a ML model has pros and cons. On the one hand, experiments provide with the best quality results, but are very expensive to carry out especially when it comes to train a neural network that needs large datasets to be reliable. When it comes to modelling two-phase flows with sharp interfaces, measurements are especially hard to carry out in the interfacial area and in the context of the present research, that is the region where we need the most information. As a matter of fact, the quantities needed in the training of the ML models developed in this thesis required the measurements of the fluctuation velocities in all directions as well as their gradient and Laplacian, which would be very challenging to carry out from an experimental point of view. On the other hand, high-fidelity simulations are much cheaper than experiments and can provide data anywhere in the flow and more specifically in the region of the interface in order to feed

the ML model training with. Thus, high-fidelity simulations are a good choice for the training of ML models in two-phase flow simulation as long as the simulation data is of great quality. In order to obtain such data quality, one has always to be cautious on the the number of computational cells in the studied domain in order to capture the smallest scales of the turbulence. Moreover, the high-fidelity simulations are compared against the reference experiments for the validation of the numerical results.

3.2.1 ML applications in single-phase flows

Existing work has shown that machine learning models could even replace CFD models [16] for the prediction of turbulence for instance. In fact, ML models can be trained to reproduce some RANS turbulence models outputs such as the prediction of turbulent source terms in transport equations. It was seen by Tracey [135] that machine learning models could be used to learn the standard Spalart-Allmaras source term for example. The researcher trained his model on two single-phase flow configurations: a turbulent flat plate and air around an airfoil. When looking at the friction coefficient, Tracey [135] found very good agreement between the ML model and the Spalart-Allmaras model as shown in figure 3.10. Here the ML model replaces the source term in the boundary layer. When looking at the residuals, differences between the ML model and the standard Spalart-Allmaras models are the greatest in the near-wall region and tail of the airfoil as seen in figure 3.11, even though it does not have an important impact on the skin friction coefficient. Despite the ML model providing promising results, Tracey [135] affirmed that such differences between prediction and truth could be explained by the fact that the ML algorithm was trained on only converged flow solutions. Using the same

approach with a ML model trained with high-fidelity data this time, Tracey et al. [134] obtained even more accurate predictions.

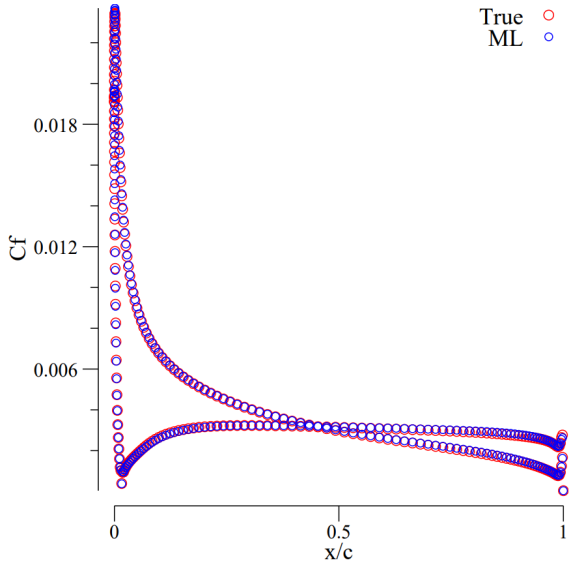


Figure 3.10 Example of a comparison obtained replacing the source term in the SA model within the boundary layer of the NACA 0012 airfoil at 8 degrees using training data from the flat plate solutions [135]

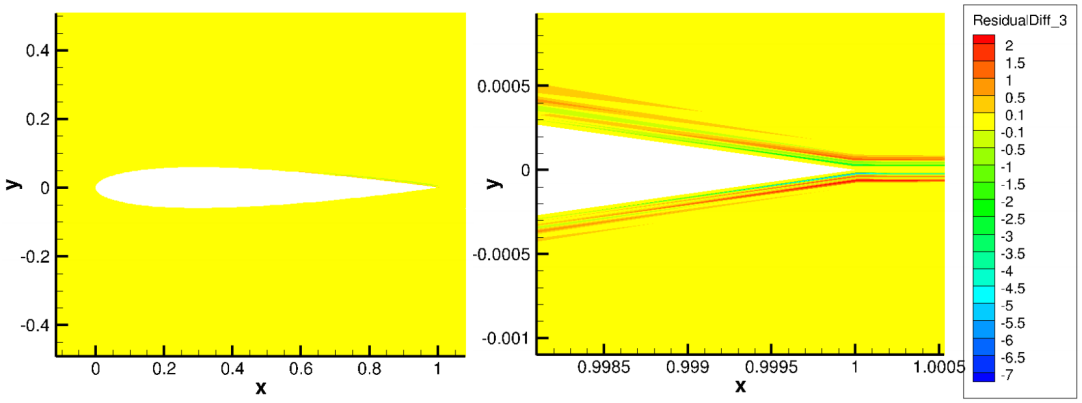


Figure 3.11 Difference in the true Spalart-Allmaras source term and the machine learning prediction in a global view (left) and a zoomed view on the tail (right) [135]

Similarly, in another example of a RANS data-driven machine learning model, Singh et al. [118] used multiple sources of data for the ML training such as DNS, LES and experiments. They carried out an inverse modelling approach to infer the spatial distribution of model discrepancies and trained the ML model to predict the discrepancy in the form of a correction term for the RANS model equations. Here, ML was employed to improve the Wilcox's $k-\omega$ model for single-phase adverse pressure gradient flows such as flows over a bump. The researchers focused on adding a correction function in the $k-\omega$ equations applied to the production term. Their ML model was trained to provide a correction term applied to correct the net balance of the source terms and resulting in an improvement of the standard model. The correction term was introduced in the ω transport equation only as the authors affirmed that it is more ad-hoc than the k equation [118]. Results showed a great improvement of the solution when looking at the friction coefficient for example, presented in figure 3.12. The researchers demonstrated the portability of this approach in other applications of a single-phase turbulent flow over airfoils [119].

Using high-fidelity data to train machine learning models to inform RANS turbulence models has then become more popular as Ling et al. [85] employed this approach to directly operate on the model eddy-viscosity. The novelty of their research also resides in the use of a deep neural network to learn the Reynolds stress anisotropy tensor. A neural network is more likely to be qualified as 'deep' when it is structured with numerous hidden layers and perceptrons, and used with high volumes of training data. Ling et al. developed an improved and advanced multi-layer perceptron (MLP) neural network and obtained significant improvements compared to simple MLP neural networks

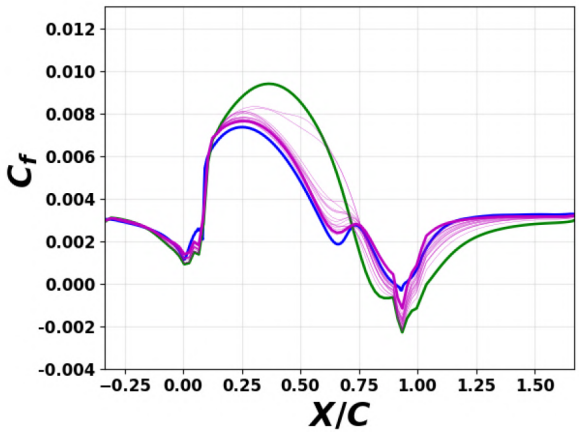


Figure 3.12 Skin friction prediction for a case example of a flow over a bump. The magenta lines represent predictions using an ensemble of machine-learned models trained on different combinations of the inverse solutions. LES is represented in blue and the standard Wilkox's $k-\omega$ in green. [118]

to inform RANS models.

More recent research [26] employed ML to correct the Reynolds force vector field and the divergence of the Reynolds stress tensor separately in the standard RANS $k-\varepsilon$ model using a DNS training dataset of a square duct. An attenuation of the error in the prediction of the mean velocity field was found with a better efficiency when applying the correction to the stress tensor for the representation of secondary flows.

ML has also proven to be useful in the optimisation of the existing standard turbulence models. For example, Luo et al. [87] implemented a ML model capable of modifying the closure coefficients of the RANS $k-\varepsilon$ model for the simulation of a flow over a bump. The ML model they used was trained with DNS data. Results showed some improvement of the RANS simulation with inferred parameters from the ML model. The mean absolute error of the velocity

profile between RANS and DNS decreased by 22% with the implemented model.

3.2.2 ML applications in two-phase flows

While numerous cases have been investigated in single-phase flow applications over the past decade, research on the potential of the use of ML for two-phase flow application remains rarer. Evidently, many ML methods developed in single-phase flow configurations have the potential to find relevancy in two-phase flow configurations. In fact, it will be seen in this dissertation that the method developed to improve the performance of the RANS $k-\omega$ model is based on the ML prediction of correction terms for the turbulence transport equations, similarly to the correction provided in previous research [119**?**].

ML have been massively employed and for a long time for the classification of two-phase flow patterns in pipes [83, 115, 6] in order to identify a given flow regime instead of basing the identification on visual observation [137]. More recently, researchers [116, 75, 93] made use of machine learning to predict the pressure drop in channel two-phase flows by including more physics in the process.

3.2.3 Concluding remarks

From simple MLP neural networks to deep MLP neural networks, the complexity of a machine learning model can vary a lot. And it is legitimate to ask: how complex should the neural network used to train a machine learning model be? Obviously the answer to this question depends on the ML application. Vinuesa and Brunton [144] discussed the relevancy of the use of neural networks

for CFD applications. They highlighted the potential of machine learning to aid CFD in numerous applications while pointing out that classical methods still remain efficient in many cases. Finding the appropriate structure of neural network for a given application and generate the high-fidelity or experimental data for the ML model training is highly time consuming, especially considering that ML model usually need large amounts of data. It should be of great importance to consider the portability of a method when developing a new ML model and generating the training datasets.

Chapter 4

High-fidelity simulation of a stratified flow in a channel

In this chapter, a preliminary study on the comparison between two available multiphase flow solvers using large eddy simulation (LES) in OpenFOAM is presented in section 4.1 for the prediction of a two-phase stratified flow in a horizontal and rectangular channel. Accuracy towards experiments and computational speed are assessed. Then, further results using the chosen multiphase method on channel stratified flow behaviours using quasi-direct numerical simulation (qDNS) are presented in section 4.2. The aim of this section is to better comprehend how the turbulence develops in these types of flow and identify the most turbulent regions of the flow with high-fidelity simulation.

4.1 Preliminary study: comparison of the VOF and Euler-Euler methods

The first objective of this study is to identify which multiphase flow method between the VOF and the Euler-Euler methods is the most appropriate

to predict stratified shear-driven flows encountered in aero-engine's bearing chambers. As seen previously, the flow of interest is located on the circular wall of the bearing chamber as presented in a schematic on figure 1.3 in chapter 1. For the purpose of this research, it was assumed that this oil-air entrained flow could be assimilated to a two-phase co-current air-water flow in a rectangular and horizontal closed channel. As previously mentioned in chapter 3, the experiments of Fabre et al. [35] were used as a first reference for the realisation of the preliminary "thick-film" high-fidelity two-phase flow simulations. This primary investigation is based on the comparison between the mean axial velocity, TKE and Reynolds stress profiles measured at the centre of the channel in both phases obtained numerically in LES with the VOF and Euler-Euler methods and obtained experimentally. The performance of each multiphase solver is also assessed on their respective computational times. In this section, the index l is used to describe the liquid phase and g the gaseous phase.

4.1.1 Simulation setup and methodology

In this part, the geometry and mesh of the computational domain used for the high-fidelity simulations are presented along with the fluid properties of each phase, the simulated flow regime and the numerical time and spatial discretisation schemes employed for the simulations in OpenFOAM.

Description of the computational domain

The experimental work of Fabre et al. [35] was briefly introduced in chapter 3, section 3.1.1. They investigated the air-water stratified flow with co-current phases in a channel of 12 m in length, 0.1 m in height and 0.2 m in width. The channel was entirely closed with walls at the top and bottom as well as

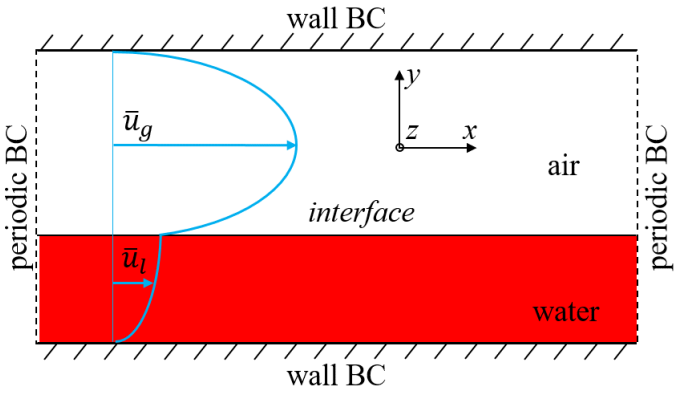


Figure 4.1 2D schematic of the simplified channel with boundary conditions (side view)

on the sides. In this chapter, we are interested in numerically representing the flow regime in which only small amplitude waves are generated. In Fabre et al. experiments, this flow regime is called 'RUN 250' and in this case, the channel is filled with air and water in such a way that the liquid thickness represents 38% of the total channel height. In the RUN 250 flow regime, the gas bulk velocity is $U_{b,g} = 4.2$ m/s while the liquid bulk velocity is $U_{b,l} = 0.45$ m/s corresponding to a gaseous phase flowing 9.3 times faster than the liquid phase. The gradient in phase velocity results in the generation of high shearing forces across the interface, inducing wavy patterns.

The geometry adopted for the comparison of the VOF and Euler-Euler methods is a periodic (recycled or cyclic) version of the experimental configuration used by Fabre et al. High-fidelity simulations require high mesh resolutions, and carrying out simulations on the full length of the original channel would be unrealistic, as computationally too expensive. Therefore, cyclic boundary conditions were set in the axial flow direction and a length of 1 m was initially chosen in order to carry out the preliminary simulations. Both

Table 4.1 Case description based on Fabre et al. [35]

Characteristic	**Symbol**	**Units**	**Value**
Gas bulk velocity	$U_{b,g}$	$\mathrm{m{\cdot}s^{-1}}$	4.20
Liquid bulk velocity	$U_{b,l}$	$\mathrm{m{\cdot}s^{-1}}$	0.45
Interface mean height	h_{int}	m	0.038
Gas hydraulic diameter	$D_{h,g}$	m	0.095
Liquid hydraulic diameter	$D_{h,l}$	m	0.055
Gas Reynolds number	$Re_{D_h,g}$	–	$2.7 \cdot 10^4$
Liquid Reynolds number	$Re_{D_h,l}$	–	$2.5 \cdot 10^4$
Gas density	ρ_g	$\mathrm{kg{\cdot}m^{-3}}$	1.00
Liquid density	ρ_l	$\mathrm{kg{\cdot}m^{-3}}$	$1.00 \cdot 10^3$
Gas kinematic viscosity	μ_g	$\mathrm{kg{\cdot}m^{-1}.s^{-1}}$	$1.48 \cdot 10^{-5}$
Liquid kinematic viscosity	μ_l	$\mathrm{kg{\cdot}m^{-1}.s^{-1}}$	$1.00 \cdot 10^{-6}$
Gas/Liquid surface tension	σ	$\mathrm{N{\cdot}m^{-1}}$	$7.00 \cdot 10^{-2}$

phases are driven towards the streamwise direction using velocity sources added to the velocity transport equations and applied to each phase. The option is specified using the *meanVelocityForce* keyword within the *fvOptions* dictionary in the OpenFOAM simulation directory. This feature enables the application of a force to maintain a user-specified volume-averaged mean velocity, and more specifically one bulk velocity for each phase in the context of this study. The channel height H and width w were kept identical as the values used in the experiments. Figure 4.1 shows a two-dimensional schematic of the cyclic channel and highlights the boundary conditions of the domain and the initial phase distribution field. Scales are not respected in the axial direction ($\vec{x}$), however the domain aspect ratio $x : y$ is normally of 10.

To characterise the flow in the liquid phase, a hydraulic Reynolds number Re_{l,D_h} based on the hydraulic diameter D_h, was calculated as follows:

$$Re_{D_h,l} = \frac{U_{b,l} \cdot D_{h,l}}{\nu_l} \tag{4.1}$$

For partially filled rectangular ducts, the hydraulic diameter writes $D_{h,l} = 4wh/(w + 2h)$ where h is the approximate interface height. In the upper part of the channel, one can consider that the gas is flowing in a fully filled duct of height $H - h$ and the equivalent hydraulic diameter writes $D_{h,g} = 2w(H - h)/(w + H - h)$. The corresponding Reynolds number is:

$$Re_{D_h,g} = \frac{U_{b,g} \cdot D_{h,g}}{\nu_g} \tag{4.2}$$

The characteristics of the case are presented in Table 4.1.

The length of the cyclic channel must be long enough in order to avoid turbulent structures to overlap themselves in the axial direction. The side, top and bottom boundary conditions of the channel are set as walls with a non-slip condition. An analysis on the cyclic lengths of the channel is proposed later in this chapter (section 4.2) in order to further simplify the geometry and save computational resources. It will also be seen that the use of cyclic boundary conditions in the z direction instead of walls allows for computational cost reductions without any loss of accuracy.

The computational mesh was generated in order to meet the guidelines for LES simulation i.e. between ($\Delta x^+ \approx 50$, $y^+ \approx 1$, $\Delta z^+ \approx 15$) [112] and ($\Delta x^+ \approx 100$, $y^+ \approx 1$, $\Delta z^+ \approx 20$) [138], which are classically required in LES to capture near wall turbulent structures [122]. y^+ is calculated as the first off-wall grid node such as $y^+ = y\sqrt{\rho\tau_w/\mu}$. The dimensionless Δx^+ and Δz^+ are the needed non-dimensional grid spacing respectively in the flow direction ($\vec{x}$) and the spanwise direction ($\vec{z}$). Far from the wall, one can take $\Delta y^+ = \Delta z^+$. A mesh refinement similar to the wall mesh refinement was also applied in the

interfacial region using a fixed grid with a chosen dimensionless wall distance from the mean interface level y^+. This was set as follows: $\bar{y}^+_{\max} \approx 1$ and $\bar{y}^+_{\text{mean}} \approx 0.5$, with y^+_{mean} the spatial mean y^+ in the domain at a given time, $\bar{y^+}$ the time averaged of y^+, and $y^+_{\max}$ the spatial maximum of y^+. The resulting maximum grid spacing respectively in the streamwise and cross-stream directions was $\Delta x = 3 \cdot 10^{-3}$ m and $\Delta z = 1 \cdot 10^{-3}$ m in order to respect the above recommendations. The mesh contains respectively in the x, y and z directions 330, 96 and 160 cells for a total of 5 068 800 cells.

4.1.2 Numerical methods

Multiphase solvers

The two-fluid Euler-Euler method was employed with the OpenFOAM solver `multiphaseEulerFoam` and the VOF method was selected with the `interFoam` solver. In both approaches, the interface compression method was employed with a compression coefficient $C_\alpha = 1$ in order to render more sharpness at the interface.

Simulation model

For both the VOF and the Euler-Euler methods, LES were performed with the Smagorinsky subgrid model [120] available in OpenFOAM. The turbulence viscosity is given by:

$$\nu_t = C_k \Delta k^{1/2} \tag{4.3}$$

where Δ is the grid size defining the subgrid length scale, and the turbulent kinetic energy k is obtained by solving the following quadratic equation for k:

$$\frac{C_e}{\Delta}k^2 + \frac{2}{3}\mathrm{tr}(\bar{\mathbf{S}})k + 2C_k\Delta\bar{\mathbf{S}} : \bar{\mathbf{S}} = 0 \tag{4.4}$$

where $C_e = 1.048$ and $C_k = 0.094$ are default model coefficients and $\mathbf{S}$ is the mean rate of strain as defined in 2.30: $\bar{\mathbf{S}} = \frac{1}{2}(\nabla\mathbf{u} + \nabla\mathbf{u}^{\mathsf{T}})$.

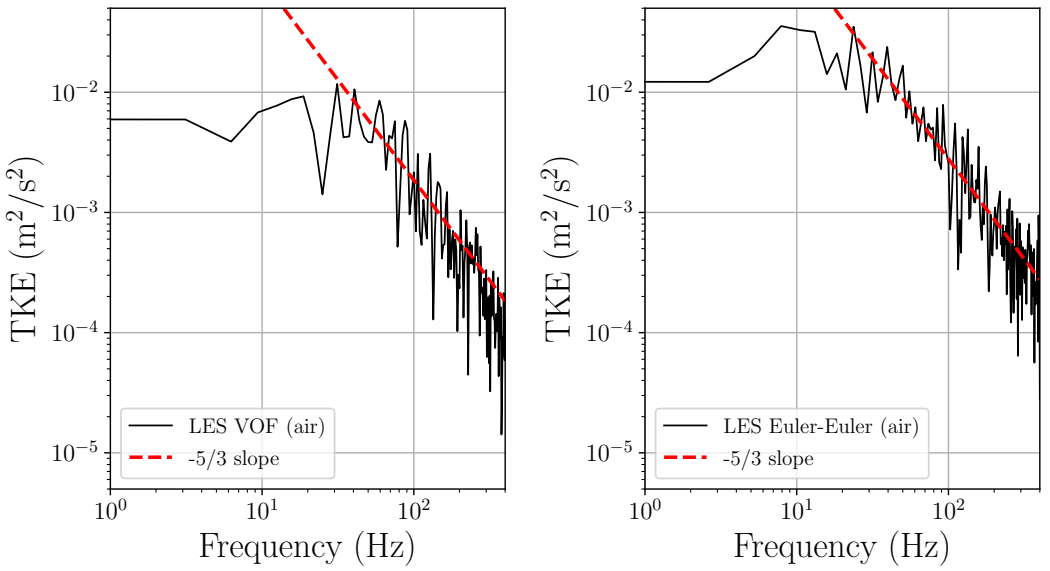
Discretisation schemes and algorithms

For the time discretisation scheme, the second order Crank Nicolson scheme [25] was selected with a coefficient of 0.9. The second order TVD "van Leer" scheme [142] was selected to solve the gradient and divergence terms, the Gauss limited linear scheme for the Laplacian term and components of gradient normal to cell faces. The pressure-velocity coupling is managed by the PIMPLE algorithm, a combination of the SIMPLE [39] and PISO [72] algorithms as described in chapter 2. The PIMPLE algorithm reaches convergence with sub-iterations and is relevant for the simulation of transient incompressible flows. For both methods, pressure was solved using the Preconditioned Conjugate Gradient (PCG) method, while the phase volume fraction, velocity and kinetic energy were solved using the Gauss-Seidel method.

The time step was adaptive and restrained by a chosen Courant number of maximum value 0.5 for all the simulations.

Velocity probing

In order to compare the numerical results to the experiments, the velocity field was probed along vertical lines in the centre of the width of the channel. The velocity field was measured in both phases allowing for the calculation of the instantaneous velocity and fluctuation velocity, mean velocities, turbulent kinetic energy (TKE), and Reynolds stresses and TKE spectra.



(a) Using the VOF method (b) Using the Euler-Euler method

Figure 4.2 Turbulent kinetic energy spectrum measured in the gaseous phase

Turbulence development and field averaging

The TKE spectra are used to monitor how the energy cascade behaves and to check whether large and small turbulent structures are observable in the LES simulations. The TKE spectra for the VOF and Euler-Euler methods were calculated by measuring the velocity fluctuations in the centre of the gaseous phase. The TKE spectra of the two multiphase methods employed are shown in figure 4.2.

The large turbulent structures are visible in the left part of the spectrum in the low frequencies. The Kolmogorov cascade is well represented here as the decrease in the turbulent kinetic energy is fitting the typical Kolmogorov -5/3 slope of equation $y = Cx^{-5/3}$ (with $C \in \mathbb{R}$), characteristic of the inertial subrange turbulent scales. This slope suggests that energy is transmitted from the large structure to the small ones. Moreover, figure 4.2 tells us that the

Euler-Euler method tends to predict larger values of TKE than the VOF method.

Once a developed flow had been obtained in each phase, field averaging operations were applied to the velocity field in order to obtain the mean velocity, the shear-stress tensor and the TKE over a period of 10 s. One liquid flow-through is approximately 2.2 s. It corresponds to the time needed by the liquid phase to go through the full length of the cyclic domain. It takes only 0.24 s for the gaseous phase to make one flow-through. A field averaging over 10 s allowed for a convergence of the mean axial velocity, shear stress and TKE profiles in the two phases, measured in the centre of the domain. The computational time to simulate one through-flow of the gaseous phase once the turbulence established in the entire domain was about 5 hours with the Euler-Euler method and less than 1 hour with the VOF method. The simulations were performed in parallel on 200 cores (Intel Skylake 6138 processors, 2.0 GHz). Thus, 25 344 cells per processor were allocated for optimal computational speeds after a scaling analysis was performed.

The function object `fieldAverage` was used to perform the field averaging operations. Two types of averages are calculated with this feature: the arithmetic mean of the velocity field and the prime-squared mean. Those operations are applied to a discrete sample of values and thus, the average velocity is obtained such as follows:

$$\bar{\mathbf{u}} = \frac{1}{T} \sum_{k=t_0}^{t_0+T} \mathbf{u}_k \tag{4.5}$$

where t_0 corresponds to the time, the averaging starts and T is the period of time over which one wants to average the velocity field u_i.

The prime-squared mean velocity writes:

$$\bar{\mathbf{u}'^2} = \frac{1}{T}\sum_{t_0}^{t_0+T}(\mathbf{u}_k - \bar{\mathbf{u}})^2 \tag{4.6}$$

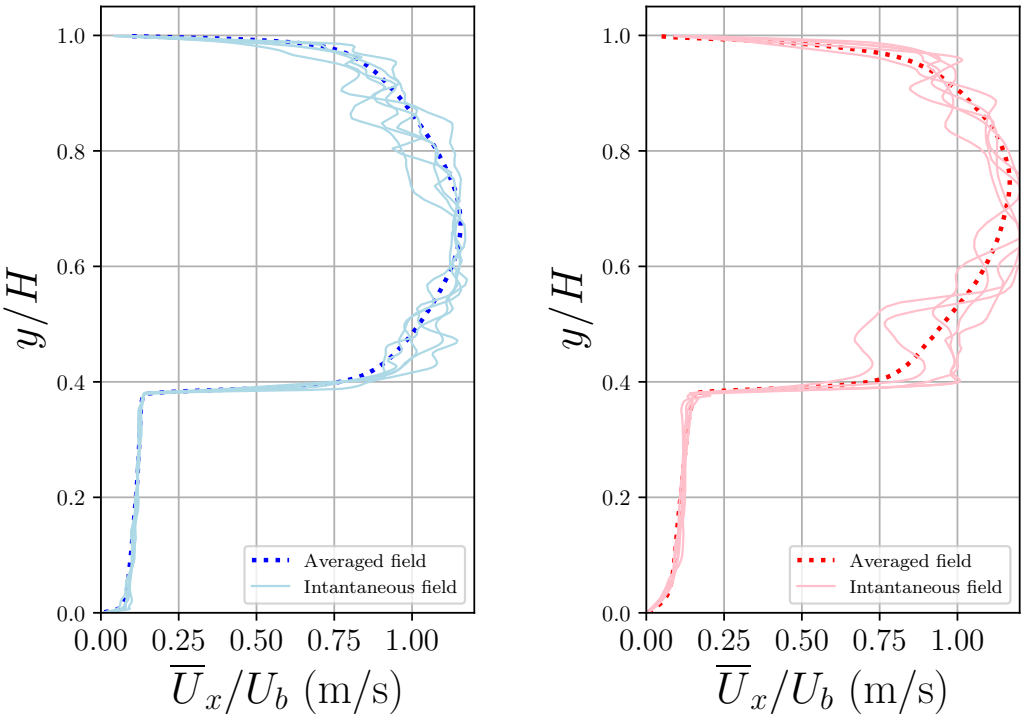
The averages are calculated "on the fly" i.e. as the simulation progresses. The prime-square mean velocity corresponds to the Reynolds stress tensor R_{ij}. Therefore the turbulent kinetic energy can directly be obtained as $k = \frac{1}{2}\mathrm{tr}(R_{ij}) = \frac{1}{2}\mathrm{tr}(\bar{\mathbf{u}'^2}) = \frac{1}{2}\mathrm{tr}(\overline{u_i' u_i'})$.

4.1.3 Results and comparison between the VOF and Euler-Euler methods

Profiles comparison

The vertical profiles of the mean axial velocity obtained with both multi-phase methods are shown in figure 4.3 and compared with a few instantaneous profiles. One can observe that the VOF profiles are more symmetrical in the gaseous phase than the Euler-Euler ones, which slightly tend to be shifted upward. In the liquid phase, the Euler-Euler method seems to predict slightly smaller values of the axial velocity in the bottom wall region.

The mean axial velocity profiles obtained with the VOF and the Euler-Euler methods are compared together with the experiments of Fabre et al. [35] in figure 4.4. Results were plotted on a logarithmic scale to better see the comparison in the liquid phase in which the bulk velocity is nearly ten times



(a) Using the VOF method

(b) Using the Euler-Euler method

Figure 4.3 Mean and instantaneous velocity profiles obtained with the two multiphase methods

smaller than the gaseous bulk velocity. According to the plots, both multiphase methods predicted results in good agreement with the experiments. The VOF method performed significantly better than the Euler-Euler as one observed that VOF is fitting almost perfectly the experiments. In the gaseous phase, the VOF slightly underestimated the velocity in the interfacial region and slightly under-predicted the velocity in the liquid phase near bottom wall. In contrast, the Euler-Euler method considerably underestimated the velocity in the gaseous phase near the interface, and overestimated it in the upper part of the channel, presenting an asymmetrical profile as mentioned above. In the liquid phase, the Euler-Euler method overestimated the velocity in comparison with the VOF and the experiments.

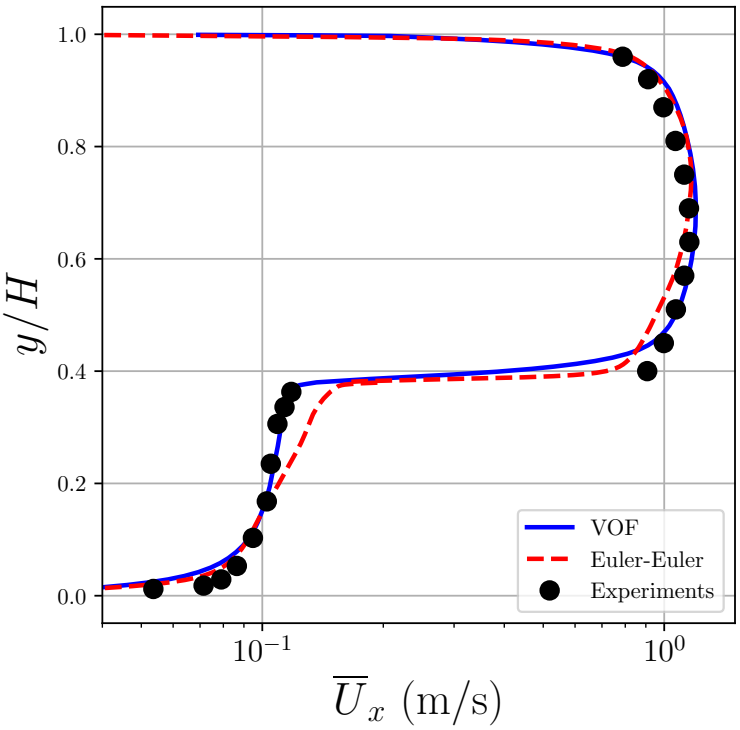
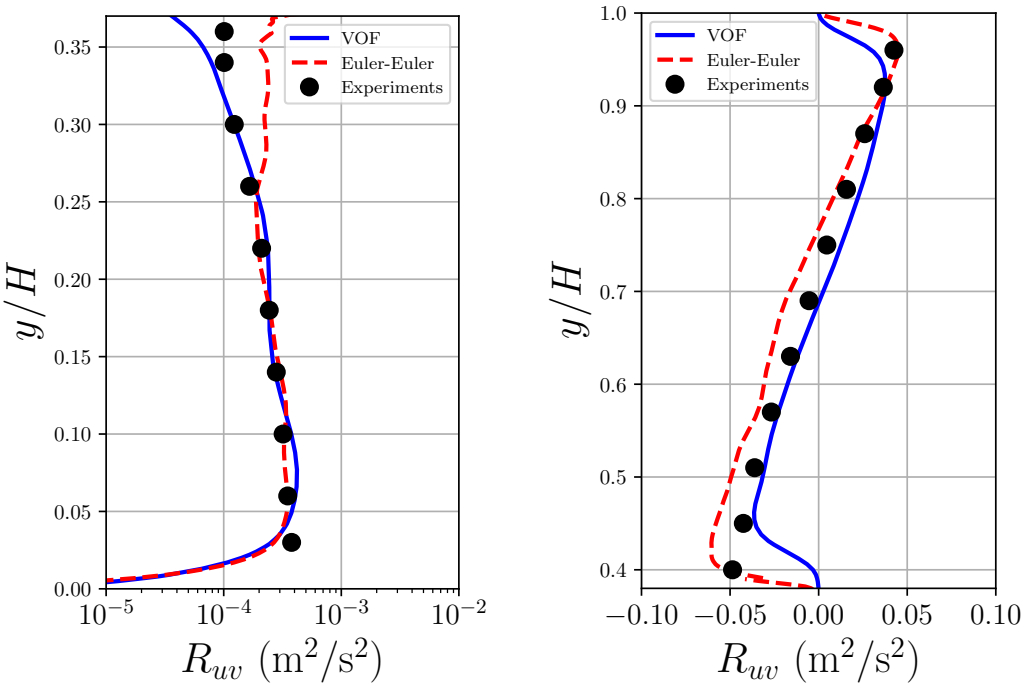


Figure 4.4 Mean axial velocity profiles

When comparing the predicted Reynolds shear stress R_{uv} with the experiments, it is observed that overall, the VOF method is performing better than the Euler-Euler method. It is seen in figure 4.5 that the Euler-Euler method underestimated R_{uv} in the near wall region of the liquid phase and slightly in the entire gaseous phase. Besides, the Euler-Euler prediction overestimated the shear stress in the liquid interfacial region. The poor performance of the Euler-Euler method in the interfacial area was expected from its inability to represented stratified flows with a sharp interface as mentioned in chapter 2. The VOF method overestimated the shear stress in the lower part of gaseous phase but overall fitted the experiments very well.

The TKE profiles were plotted in figure 4.6 using a logarithmic scale in order to better visualise the results in the liquid. It is obvious from the graph



(a) Reynolds stress in the liquid (b) Reynolds stress in the gas

Figure 4.5 Axial velocity profiles

that the VOF method predicted much better TKE values than the Euler-Euler method when compared to the experiments. The VOF method overestimated the TKE in the lower part of the gaseous phase and lower part of the liquid phase, but overall, results were in good agreement with the experiments. On the contrary, the Euler-Euler method performed significantly much less, especially in the gaseous phase where the TKE was largely overestimated, as anticipated by the TKE spectra plots in figure 4.2.

Interfacial vortex identification

In order to observe the behaviour and shapes of the turbulent flow structures and especially in the interface region, the vorticity magnitude was calculated in each phase to highlight the presence of large turbulent structures. Vorticity

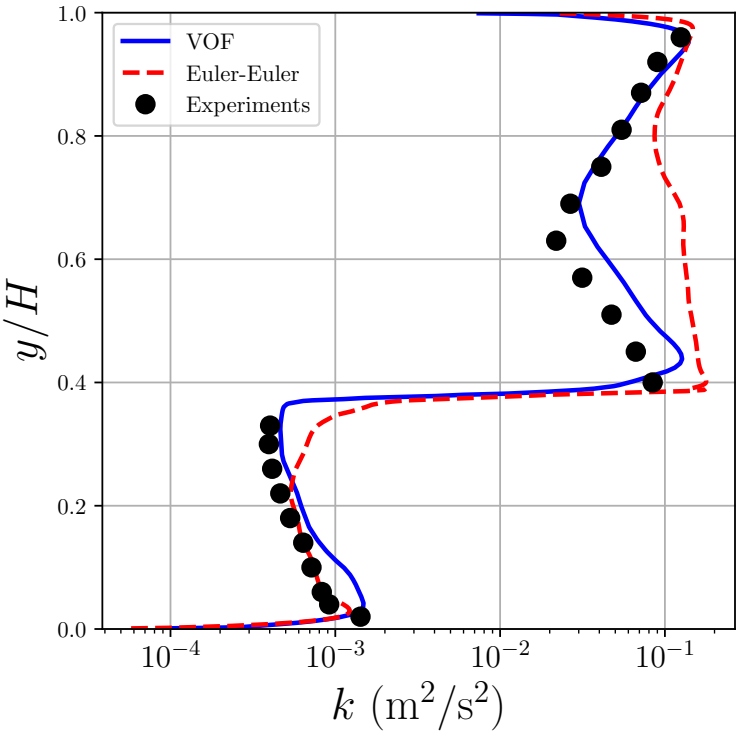


Figure 4.6 Turbulent kinetic energy profiles

magnitude contours obtained with the VOF and Euler-Euler methods are shown in figure 4.7 and 4.8. In figure 4.7, the contour plane (x,z) was placed just above the interface at $y = 0.040$ m in order to observe the interfacial turbulence in the air phase and in figure 4.8 the plane was placed a little further away up from the interface level at $y = 0.045$ m.

Figure 4.7 shows no clear difference between the two multiphase methods on the horizontal plane just above the interface in the gaseous phase. The VOF method appeared to present periodic wave-like turbulent structures in the flow stream direction. No remarkable differences in the vorticity contours between the two methods are visible in figure 4.8 when looking further away from the interface. The coherent structures expanding in the flow direction are assimilated as vortices and can be better identified with a vortex identification

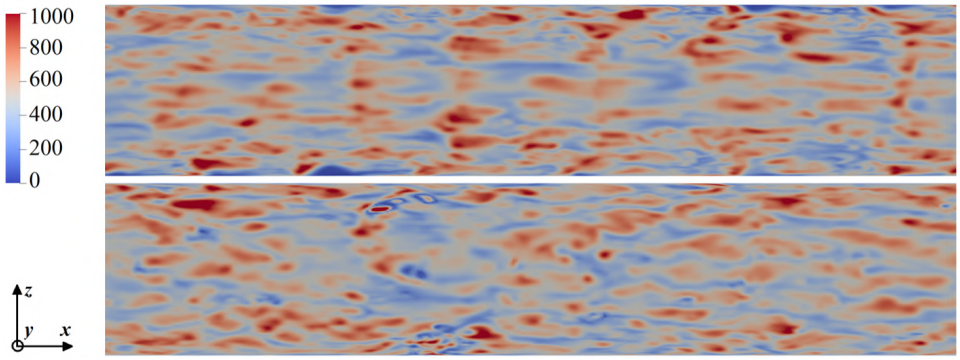


Figure 4.7 Contours (x, z) of vorticity magnitude with the VOF (top) and Euler-Euler method (bottom) on a plane located above the interface at $y = 0.040$ m

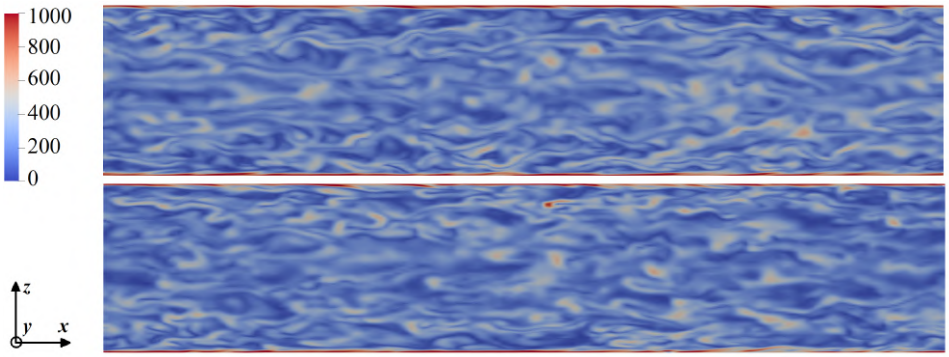


Figure 4.8 Contours (x, z) of vorticity magnitude with the VOF (top) and Euler-Euler method (bottom) on a plane located above the interface at $y = 0.045$ m

method. The Q-criterion method [67] was employed for that purpose. It can be written as follows: $Q = 0.5 \cdot (||\mathbf{\Omega}|| - ||\mathbf{S}||)$. Those regions corresponds to a positive value of Q. Filtering the regions in which $Q >> 0$ is equivalent to selecting the regions where the vorticity is much greater than the rate of strain and where we would expect to see the largest scales of the turbulence. Three-dimensional isosurfaces of Q-criterion are plotted in figure 4.9.

Vortices and large eddies are identified in the gaseous phase near the wall and the interface. As can be seen the two approaches do not show any major differences. We notice that there are turbulent structures detaching from the interface that enter the bulk air flow and decay in both multiphase methods.

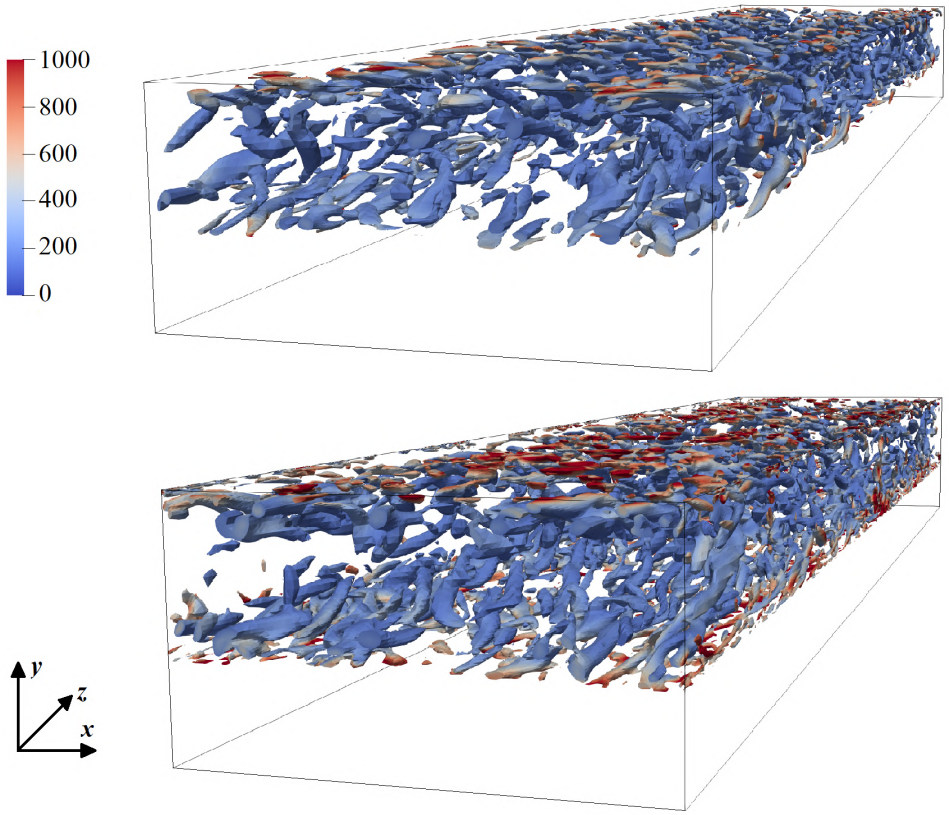


Figure 4.9 3D isosurfaces of Q for $Q > 4\ 500$ with contours of vorticity magnitude with the VOF method (top) and with the Euler-Euler model (bottom)

Conclusions on the results

Large eddy simulations of an air-water stratified flow in a "thick film" configuration in a horizontal, closed, cyclic, and rectangular channel were performed. The two main multiphase methods available in OpenFOAM, namely the Euler-Euler method and the volume of fluid method, were employed for the LES simulations. The numerical results were compared to the experiments of Fabre et al. [35] conducted on stratified flows in a rectangular channel. Overall, numerical results showed good agreements with the experiments when comparing the mean axial velocity and shear stress profiles. The main difference between the two multiphase methods was observed in the TKE profiles in which the Euler-Euler approach performed poorly in the gaseous phase by significantly

overestimating the values in the centre of the phase. This is an interesting find given the Euler-Euler approach solves a set of Navier-Stokes equations for each phase and is hence at least twice as expensive. It was indeed found that the VOF method was actually five times less expensive than the Euler-Euler method for the fully developed flow. There are however more significant variations in root mean square turbulent kinetic energy and cross-Reynolds stress. In high-fidelity eddy-resolving approaches such quantities are well known to be highly sensitive to CFD code numerics and model implementations. In fact, such variations can be significantly greater than those relating to choice of model itself, see for example Eastwood et al. [31]. From a practical engineering perspective the small variation observed in the mean velocity profiles between the two models of quite different complexity is a very useful find. TKE spectra were also plotted in order to examine the presence of turbulent-scales following the Kolmogorov cascade. Isosurfaces of Q-criterion highlighted the presence of large turbulent structures near walls and near the interface. In conclusion, the VOF method was retained to perform all the other simulations presented in this dissertation.

4.2 Further analyses using the VOF method

In this second part, it was aimed to observe the turbulent eddies generated in the flow. More specifically, the vorticity magnitude in the interfacial region of the stratified flow was studied in order to identify vortices and large coherent structures generated by the shearing forces. The Q-criterion method was also employed to help this flow identification and will be further described in this section. In this part, quasi-DNS simulations were performed with the VOF method and the same VOF solver, discretisation schemes, algorithms

and field averaging techniques as described in section 4.1.2 were employed. A methodology was proposed to simplify the previously presented computational domain.

4.2.1 Domain geometry

In order to further analyse the two-phase flow based on the experiments of Fabre et al. [35], a simplified geometry was developed using the same boundary conditions (BC) as previously (c.f. figure 4.1) and the same flow conditions as also described previously (c.f. table 4.1), only this time the length of the channel was reduced and recycled boundary conditions were set in place of the side walls, as well as a reduction of the channel. This simplification was carried out in order to increase the mesh density while saving computational resources. The cyclic length L in the flow direction ($\vec{x}$) was reduced to 0.1 m and the cyclic width w in the span-wise direction ($\vec{z}$) was set to 0.025 m. An autocorrelation analysis of the fluctuation velocity in the two cyclic directions is presented in section 4.2.4 in order to check that such simplifications do not trigger any unphysical results. Moreover, as the channel is now also cyclic in the width direction, the hydraulic diameter is evaluated differently. One can use $D_{h,l} = 4h$ in the liquid phase and $D_{h,g} = 2(H - h)$ in the gaseous phase. A 3D schematic of the new domain geometry is shown in figure 4.10.

4.2.2 Quasi-DNS and small turbulent scales

In the preliminary study, results using LES with a Smagorinsky subgrid model on a stratified flow in a horizontal channel were presented. Better results were obtained using numerical LES, also named quasi-DNS (qDNS) in this research. It means that the simulations were carried out without any subgrid

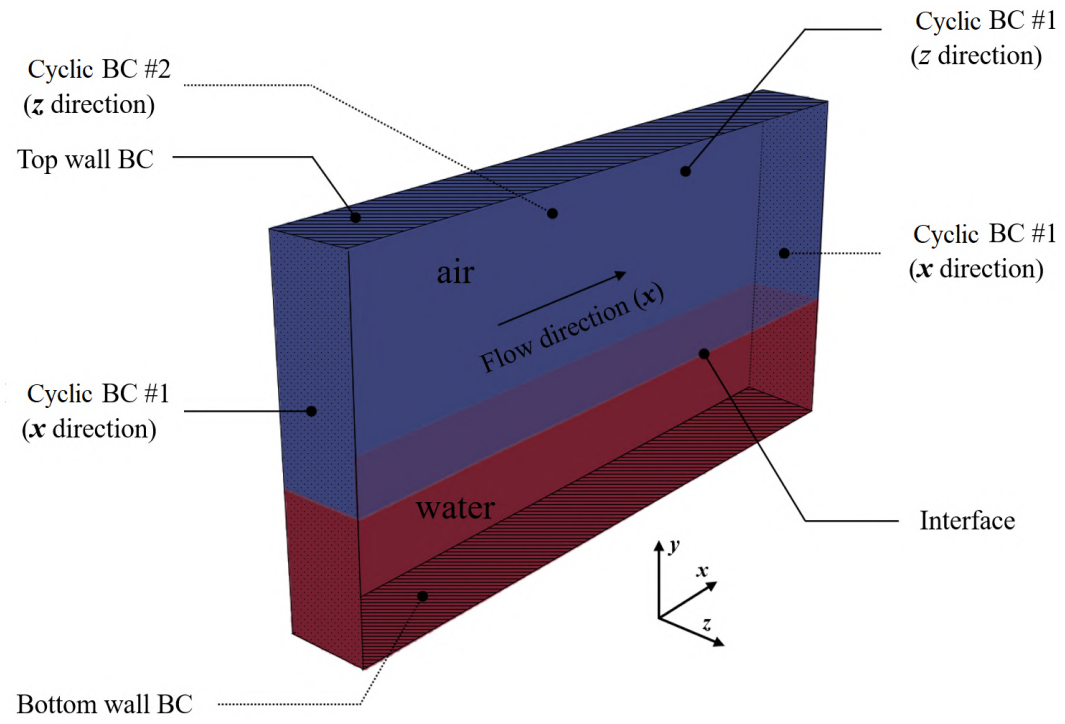


Figure 4.10 Schematic of the simplified domain's geometry and boundary conditions

model and use numerical dissipation in its place. No wall function were used to model the flow in near-wall regions. The grid cell size Δy must be sufficiently small to capture the important small scales of the turbulence, as no subgrid model is used to model them. To assess the mesh resolution, a comparison to the Kolmogorov turbulent scale η can be performed. One Kolmogorov scale for each phase is calculated using $\eta_i = (\nu_i^3/\varepsilon_i)^{1/4}$ as described in chapter 2, section 2.3.4, where ε_i is the rate of energy dissipation in each phase and can be estimated as follows: $\varepsilon_i \sim U_{b,i}^3/L_i$ where $U_{b,i}$ and L_i are the bulk velocity of each phase and the characteristic scale of the flow respectively. Hence, the following estimation of the phase Kolmogorov scale can be made:

$$\eta_i = \left(\frac{\nu_i^3 L_i}{U_{b,i}^3}\right)^{1/4} \tag{4.7}$$

For the studied stratified flow, the characteristic scale of each phase is the phase height in the channel, as channel periodicity in the width direction is equivalent to an infinitely wide channel. The role of the smallest turbulent scales is to convert the TKE into internal energy and when slightly under-resolved mesh is used, the role of those smallest scales can still be completed by the remaining resolved scales. For under-resolved mesh ($\Delta y > 20\eta$), numerical instabilities are more likely to happen and lead to nonphysical results. Typically, qDNS needs at least a resolution two to five times coarser than the Kolmogorov scale [133]. In this work, interfacial regions were refined similarly to the wall refinements and on both sides of the interface.

The calculated Kolmogorov length scale for the gaseous phase was $\eta_g = 1.2 \cdot 10^{-4}$ m and for the liquid phase $\eta_l = 2.4 \cdot 10^{-5}$ m. Hence the qDNS guidelines recommend a refinement corresponding to the interval $[2\eta_g, 5\eta_g]$ in the gas and $[2\eta_l, 5\eta_l]$ in the liquid to capture the physics in the wall regions. The interface region must also be refined to capture wall-like physics. In order to keep consistency with physics, the grid size in the interfacial region should ideally be smaller than $5\eta_i$ on both sides of the interface. To avoid any nonphysical results, the interfacial grid size should remain smaller than $20\eta_i$ [133].

4.2.3 Mesh resolution

Six levels of refinement "s0", "s1", "s2", "s3", "s4", and "s5" were tested to study the mesh convergence on the mean axial velocity, TKE and Reynolds stress. "s0" corresponds to the maximum level of mesh refinement and "s5" the coarsest mesh. The computational meshes were generated in order to meet the qDNS requirements as previously described. The mesh characteristics for

the six meshes are presented in table 4.2. The mesh density of each case is given by the table, calculated on a computational domain volume of 250 cm^3. The table also provides the average dimensionless wall distances $\bar{y}^+$ and a comparison of the first off-wall and -interface grid size vertically (Δy) with the Kolmogorov length scale. The directions $\vec{x}$ and $\vec{z}$ were uniformly discretised as they correspond to the cyclic directions. Thus, the grid size in the both cyclic directions is obtained as follows: $\Delta x = L/N_x$ in the flow-stream direction and $\Delta z = w/N_z$ in the span-wise direction, where N_x and N_z correspond to the number of cells in the x and z direction respectively.

The three cases with the most refined meshes s0, s1, and s2 all meet the guidelines for qDNS simulations when comparing wall and interface grid sizes to the Kolmogorov microscale i.e. they were constructed with refinements allowing for wall and interface grid sizes Δy belonging to $[2\eta_g, 5\eta_g]$ in the gas and to $[2\eta_l, 5\eta_l]$ in the liquid. The three other cases with the coarsest meshes s3, s4, and s5 also meet the guidelines for the interface and wall refinements in gaseous phase and for the liquid wall refinements, however, they are under-refined in the liquid interfacial region i.e. $\Delta y_l > 5\eta_l$ from the refinement level s3.

4.2.4 Results

Convergence study

In terms of computational times, every simulation costed on average 13.8 core-hours by distributing on average 20000 cells per processor. In order to assess the quality of the qDNS results for the six levels of mesh refinement, the root mean square errors the mean axial velocity, TKE, and Reynolds stress

Table 4.2 Characteristics of the six meshes

Feature	**s5**	**s4**	**s3**	**s2**	**s1**	**s0**
Nb. cells mesh	8k	18k	30k	114k	250k	1011k
Nb. cells x	16	22	26	38	50	80
Nb. cells y	64	82	96	158	200	316
Nb. cells z	8	10	12	19	25	40
Nb. cells/cm^3	33	72	120	456	1000	4045
$\bar{y}^+$	1.9	1.4	1.2	0.81	0.67	0.41
Wall Δy_l	$2.2\eta_l$	$1.8\eta_l$	$1.5\eta_l$	$0.8\eta_l$	$0.6\eta_l$	$0.4\eta_l$
Wall Δy_g	$1.8\eta_g$	$1.2\eta_g$	$1.1\eta_g$	$0.8\eta_g$	$0.6\eta_g$	$0.4\eta_g$
Interface Δy_l	$17\eta_l$	$14\eta_l$	$9.5\eta_l$	$5.0\eta_l$	$4.7\eta_l$	$3.1\eta_l$
Interface Δy_g	$3.2\eta_g$	$2.5\eta_g$	$2.3\eta_g$	$1.4\eta_g$	$1.1\eta_g$	$0.7\eta_g$

between the interpolated experimental measurements and the profiles obtained in qDNS were calculated. Results are shown in figure 4.11 and give a rough idea of the trend within the numerical simulations. One can observe that the most refined meshes are closer to the experiments, although the level s1 seemed to produce results closer to the experiments for the shear stress.

One can better visualise the mesh convergence by looking at the mean axial velocity, TKE and shear stress profiles plotted in figure 4.12. The Reynolds stress profiles seem indeed to converge towards a certain profile that was predicted in the s0, s1 and s2 refinement levels. Results were plotted in the gaseous phase only in order to better visualise the convergence of the profiles. Results plotted in the gaseous were clearly in good agreement with the experiments.

In order to further study the thick film configuration, the results obtained with the refinement level s1 were investigated. According to table 4.2, mesh s1 meets the qDNS guidelines. Figure 4.14 shows a side view and cross-section view of the refinement level s1. The other two meshes use the same type of

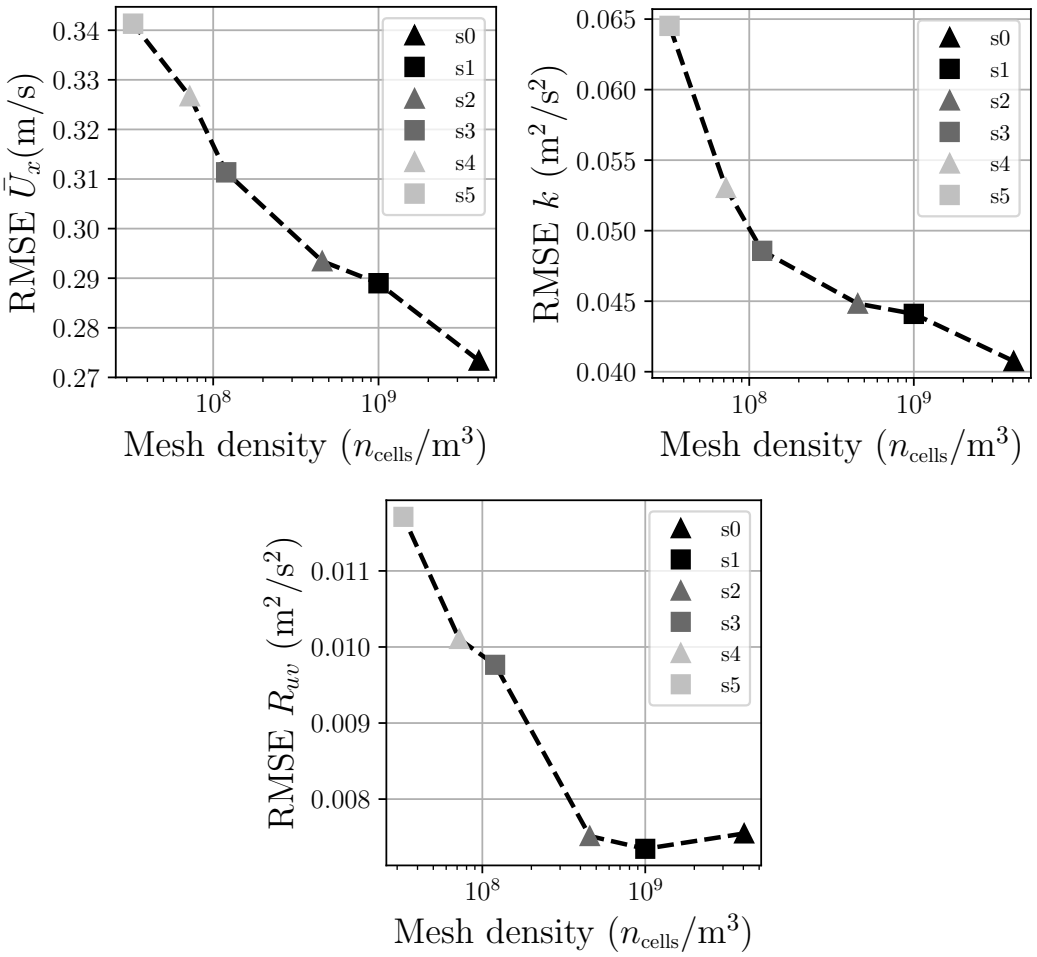


Figure 4.11 Root mean square error of the mean axial velocity (top left), turbulent kinetic energy (top right), and Reynolds stress (bottom) between Fabre et al. experiments [35] and the VOF simulations, against the mesh density

grading for the interface and wall regions. Moreover, its predictions were very close to s0 refinement level's predictions. The computational cost of the case s1 is reasonable, being nearly five times lower than the s0 case at 0.69 core-seconds. For instance, it would take 7.6 hours on 10 cores for the case s1 to simulate ten physical seconds, while s0 would need 39.4 hours to simulate the same amount of physical time.

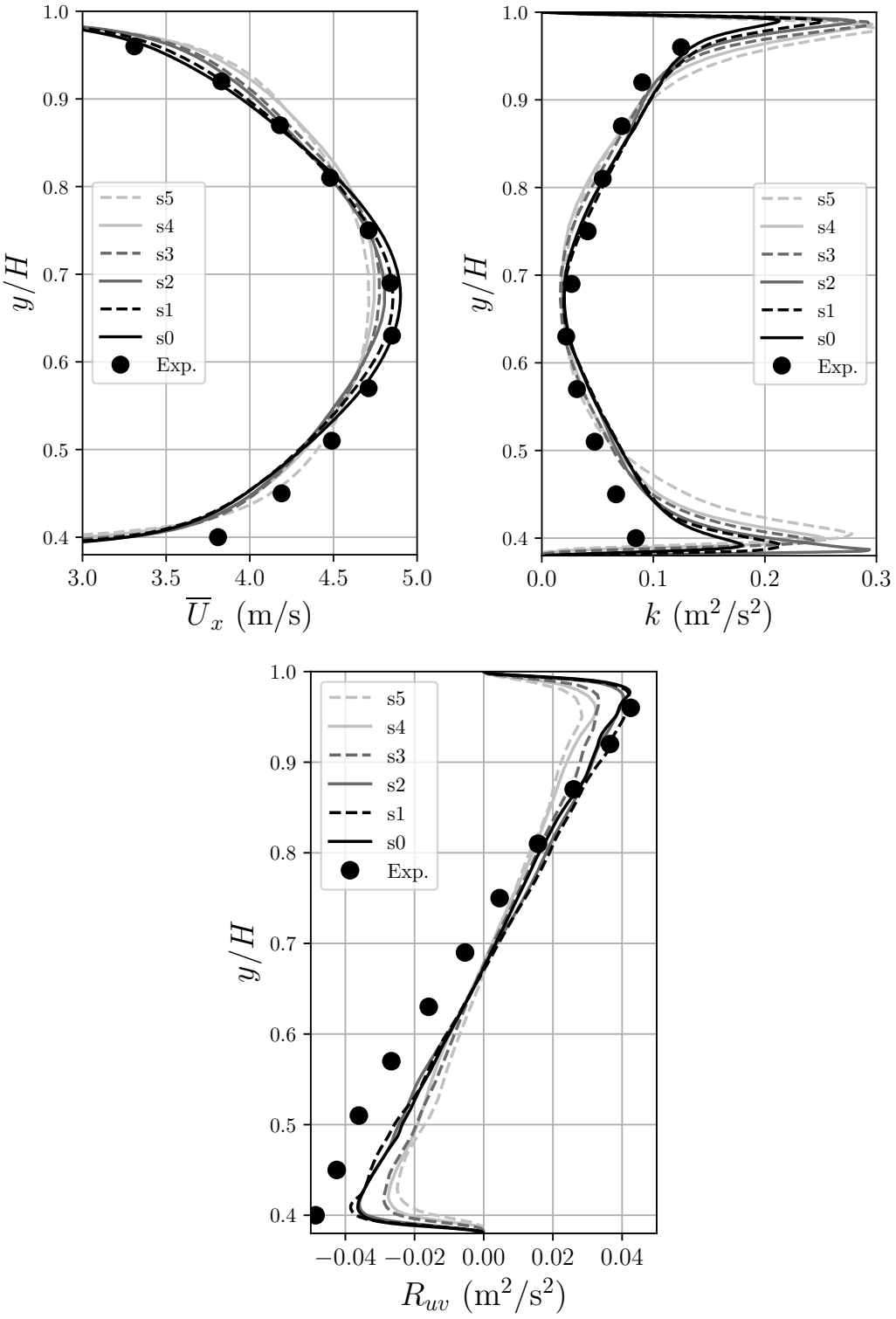


Figure 4.12 Mesh convergence analysis for the mean axial velocity (top left), turbulent kinetic energy (top right), and Reynolds stress (bottom)

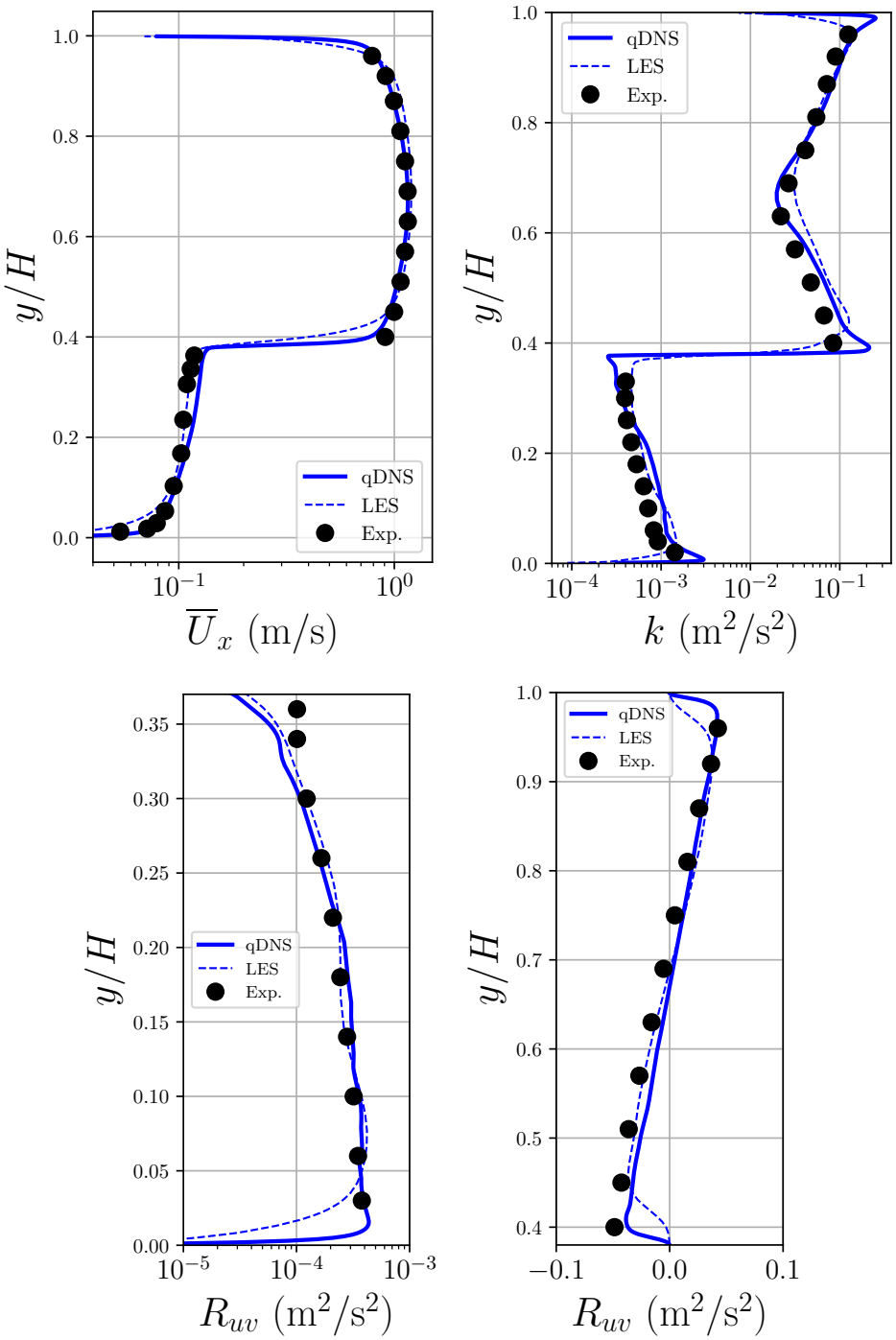
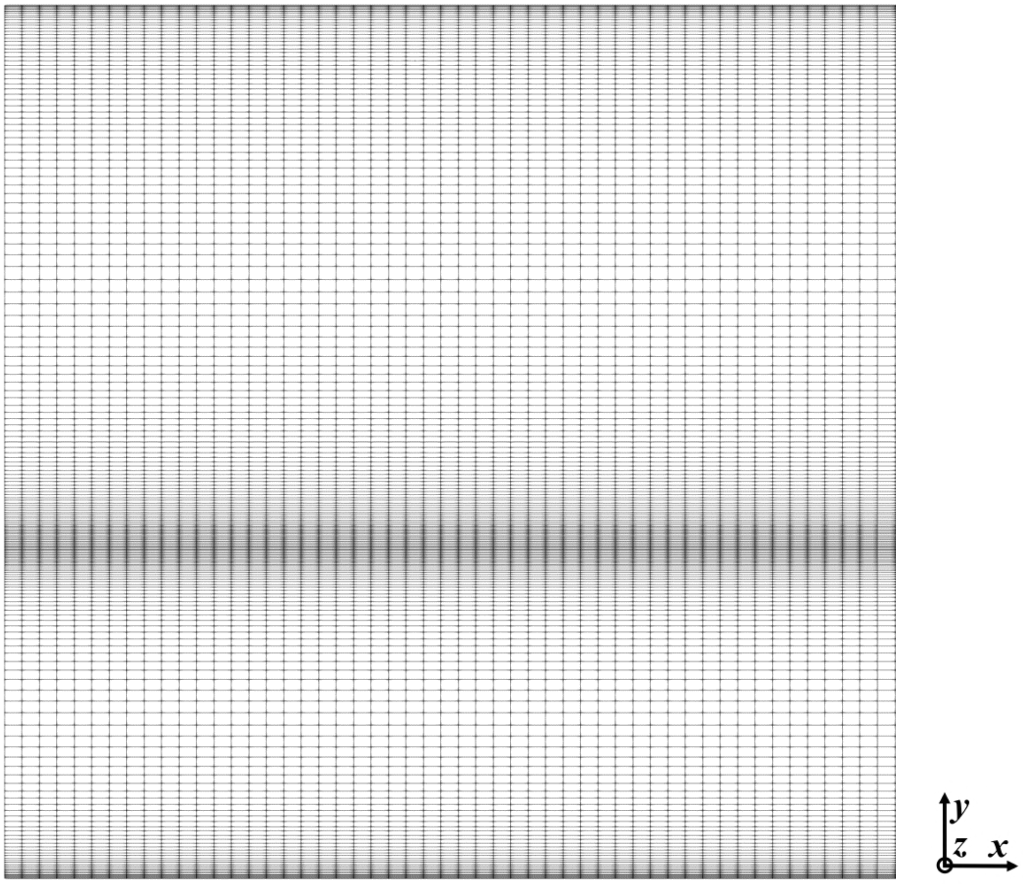
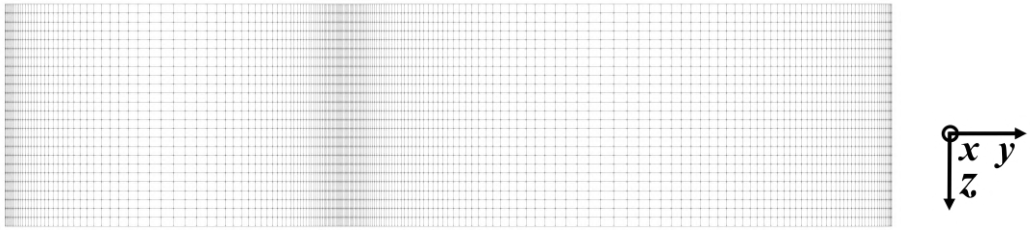


Figure 4.13 Mean axial velocity (top left), TKE (top right), shear stress in the liquid (bottom left) and in the gas (bottom right)

A comparison between the qDNS predictions using the refinement level s1 and the LES predictions with the VOF method presented in the preliminary study is shown in figure 4.13. Results show clearly that the qDNS simulation



(a) Side view



(b) Cross-section view

Figure 4.14 Mesh representation corresponding to the refinement level s1

provided excellent predictions of the mean axial velocity, TKE, and shear stress profiles. Overall, the qDNS simulation fitted the experiments better than the LES simulation, especially in the wall and interfacial regions. One can note, however, that the liquid velocity was slightly overestimated by the qDNS in the upper part of the film. In their experiments Fabre et al. [35] used the well-tried

method to determine the stress profiles i.e. they deducted it from the difference between the measured intensities corresponding to a beam plane angle of +45, –45 degrees. They also smoothed the data to avoid scattering, which contributed to a decrease in accuracy. Despite this methodology, the predicted Reynolds stress profiles were found to agree very well with the experiments.

Autocorrelation and integral length scales

The computational domain was made cyclic for computational cost reduction. To examine the extent of the cyclic dimensions an autocorrelation study of the flow was performed. The autocorrelation function can be used to determine the minimum cyclic length of a computational domain that is needed to avoid any nonphysical results when it comes to represent the largest turbulent structures of the flow. If the cyclic length is too small then large turbulent structures might overlap and lead to unphysical results. It is safe to affirm that the cyclic length is sufficiently long if the autocorrelation function of the fluctuation velocity drops to zero halfway or very small [42]. The autocorrelation of a quantity is a function of the time lag Δt and space lag Δx where $x_i = x$ in the flow direction and $x_i = z$ in the cross-flow direction. The autocorrelation of the fluctuation velocity is calculated as follows:

$$R_{x_i x_i}(\Delta x_i, \Delta t) = \frac{\overline{u'_x(x,t)u'_x(x+\Delta x, t+\Delta t)}}{\sqrt{\overline{u'^2_x(x,t)}}\sqrt{\overline{u'^2_x(x+\Delta x, t+\Delta t)}}} \tag{4.8}$$

$R_{x_i x_i}$ is bounded by -1 and 1. A value of 1 indicates that the flow is perfectly correlated and a value of 0 indicates no correlation of the flow. Figure 4.15 shows the spatial autocorrelation of the axial fluctuation velocity which is the quantity of interest, $R_{xx}(\Delta x, 0)$, averaged over a period of 10 seconds. It

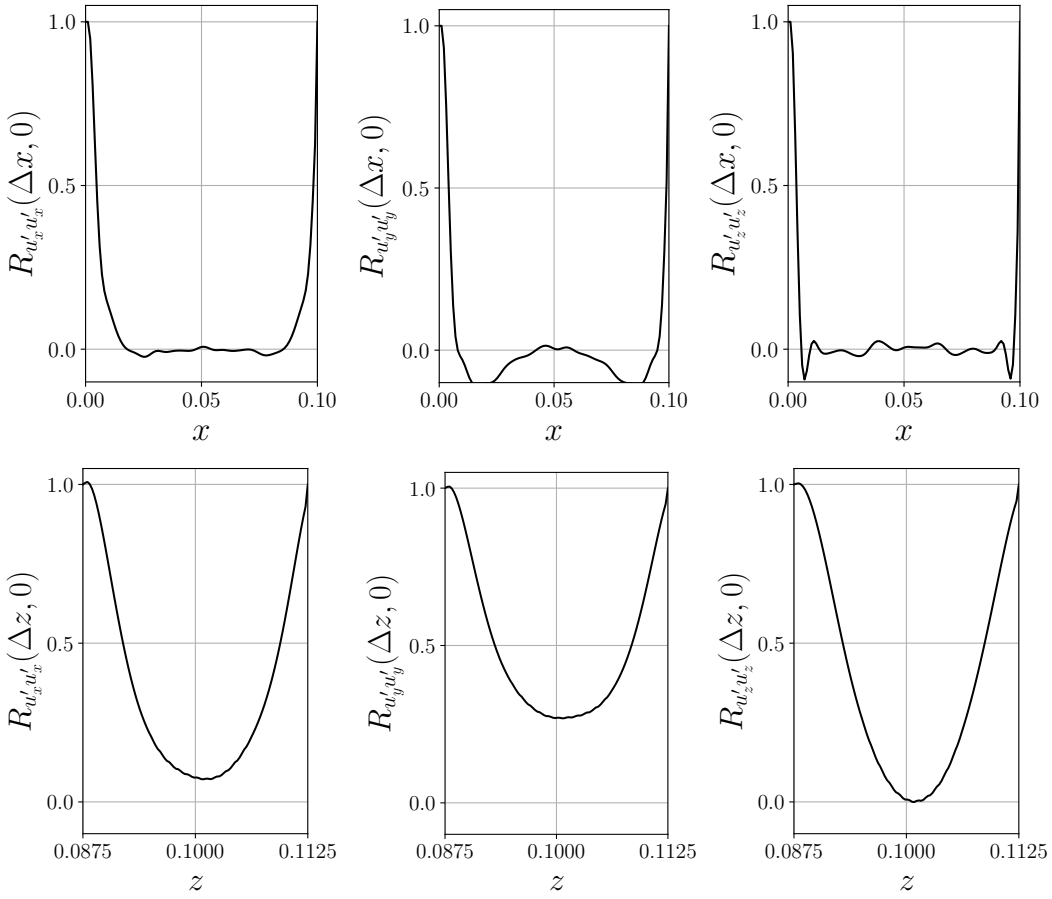


Figure 4.15 Spatial autocorrelations of the x, y and z-components of the fluctuation velocity measured in the centre of the gaseous phase in the flow direction (top) and in cross-flow direction (bottom)

was evaluated at the height $y = 0.069$ m in the centre of the gaseous phase, where the largest turbulent structures are generated. One can deduct from the spatial autocorrelation plots that the cyclic length is sufficient in the flow direction as R_{xx} drops to 0 before the half of the length. The autocorrelation in the spanwise direction also reaches very small values halfway through the width of the channel and indicates a sufficient decorrelation of the flow in the cross-stream direction.

Figure 4.16 shows the contours of autocorrelation of the axial fluctuation velocity over space and time in the gaseous phase. The map was obtained

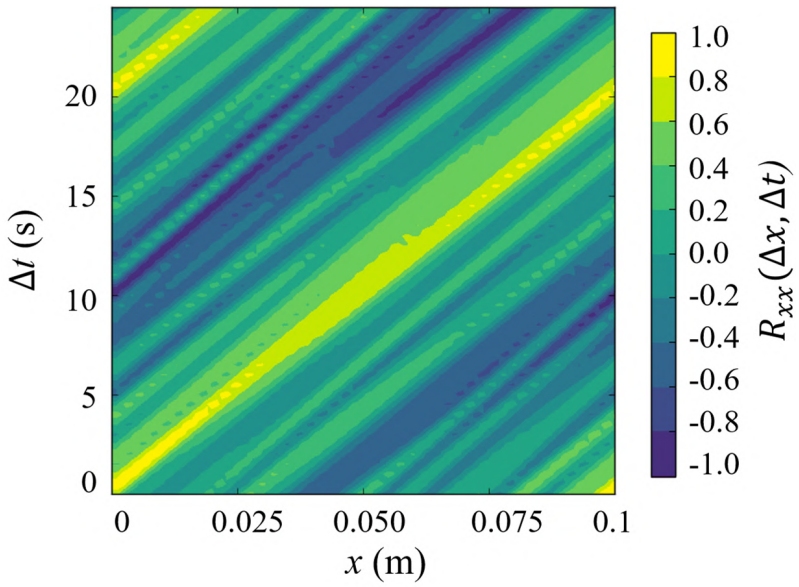


Figure 4.16 Autocorrelation contour map $R_{xx}(\Delta x, \Delta t)$ of the axial fluctuation velocity in the gaseous phase measured in the streamwise direction for $50\Delta t = 0.05$ s.

by probing two lines horizontally in the (x, z) plane the centre of the phase ($y = 0.069$ m) over time. We observe a contour line of strong autocorrelation of the flow moving at the velocity $x/\Delta t$ where $\Delta t = 0.005$ s. The full time interval over which the autocorrelation was plotted corresponds to one flow through over time of the cyclic length. The flow was partially averaged over each Δt letting us see variations of the autocorrelation with time and space.

Turbulence analysis

As already mentioned in the previous section, the refinement level s1 was used for the rest of the analysis, corresponding to a mesh density of 1000 cells/cm^3 (c.f. table 4.2). In comparison, the LES results presented in the preliminary study were performed on a mesh density of 253 cells/cm^3, which is four times less dense than s1. The LES mesh lies between the s2 and s3 refinement levels. Therefore, one expects to resolve much more turbulent scales in qDNS using the mesh s1 than in the former LES study. The TKE spectra

measured in the centre of the gaseous phase in qDNS using the mesh s1 and in LES using the VOF method are compared together in figure 4.17.

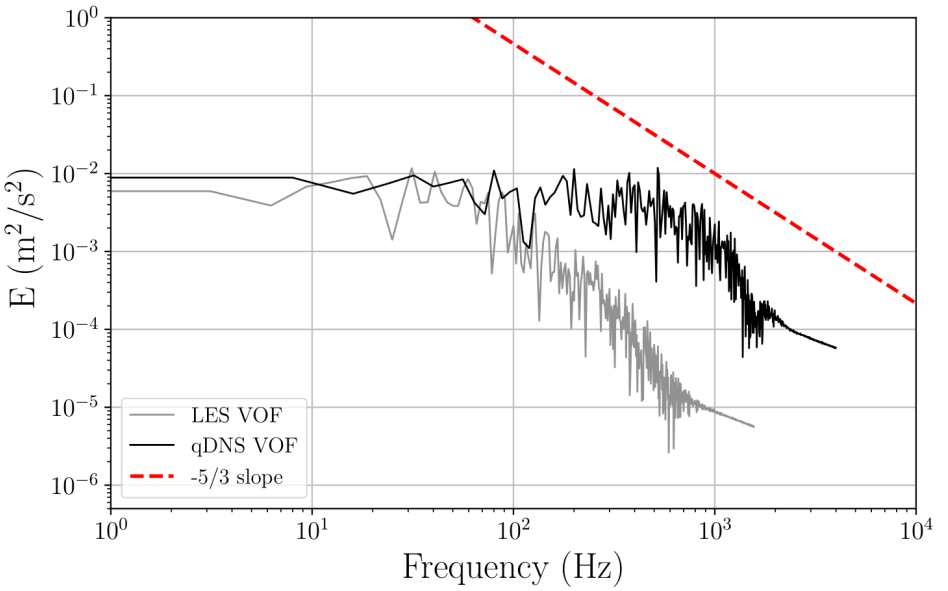


Figure 4.17 Energy spectrum measured in the gaseous phase in qDNS compared with the LES measurements

According to the TKE spectra, both the LES and the qDNS simulations solved similar levels of turbulent energy in the lower frequency domain, corresponding to the largest turbulent structures. When moving to the higher frequency range in the inertial subrange of turbulent scales, the qDNS TKE spectrum also follows the Kolmogorov slope. Furthermore, one can observe that the qDNS solved higher frequency turbulent scales, which was expected as qDNS was performed with a better resolved mesh and without any subgrid model. It is clear that qDNS solved high energy turbulent structures until nearly $2 \cdot 10^3$ Hz, while LES could only solve lower energy structures and at lower maximum frequencies ($\approx 7 \cdot 10^2$ Hz). The high energetic small eddies resolved in qDNS made a difference in the accuracy of the flow prediction, as

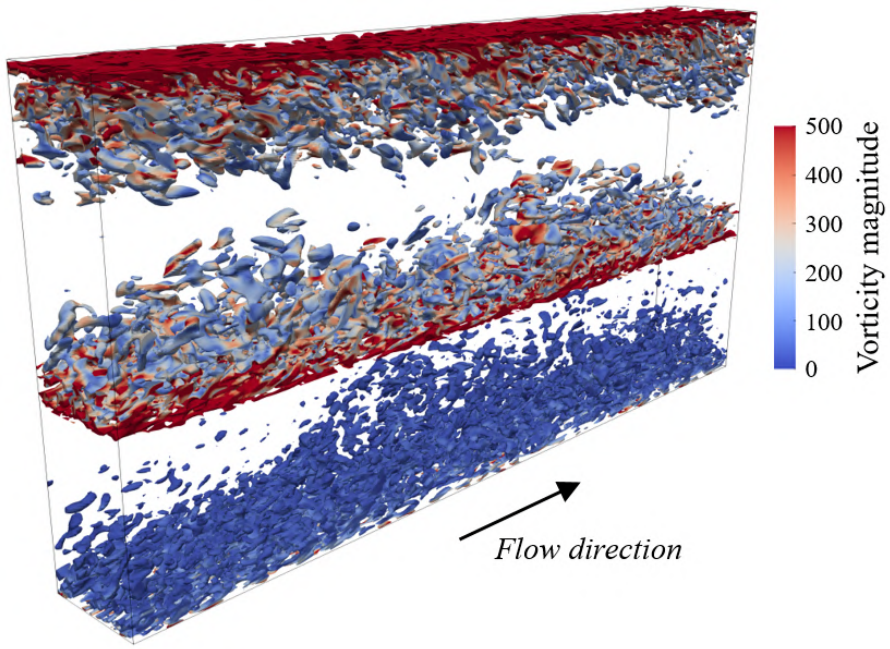


Figure 4.18 Isosurfaces of Q-criterion and contours of vorticity magnitude on a 3D view of the cyclic channel using the refinement level s1. The Q-criterion threshold is 100 times higher in the gaseous phase than the liquid phase

previously shown in the profile plots in figure 4.13. However, those differences could also be explained by differences in the mesh resolution between the two simulation methods, which impacts directly the time step of the simulation through the Courant number.

The strongest discontinuities in the velocity and energy fields were found in the interfacial regions, where the biggest differences with the experiments were also found. This section focuses on identification of the type of turbulent structures that are observed in this region of the channel. Again, in order to highlight the regions where the flow is dominated by the vorticity $\mathbf{\Omega}$ rather than by the rate of strain $\mathbf{S}$, the Q-criterion method was employed. Three-dimensional isosurfaces of Q-criterion are shown in figure 4.18 with contours of instantaneous vorticity magnitude.

Large coherent structures expanding from the interface and walls are observed in the flow direction in both phases. Thus the lowest turbulence intensities are located in regions where no isosurface is represented i.e. in the centre of each phase. The turbulence cascade of Kolmogorov is well described with the creation of large turbulent structures with high energy values i.e. high vorticity magnitudes are detected near the walls and above the interface, generating eddies that detach and flow towards the centre of the phase and dissipate. This process was already visible in the TKE profiles in figures 4.12 and 4.13 where high peaks of turbulent energy were visible near the walls and above the interface and a decreasing in energy by a factor of at least 10 compared to the top wall and interface was observed in the centre of the gaseous phase as well as a decrease in energy in the upper part of the liquid phase far from the wall.

Figure 4.19 shows the distribution of the instantaneous vorticity magnitude along the channel and enhances the view of the energy dissipation in the centre of the gaseous phase and in the upper part of the liquid phase. It also shows the liquid volume fraction at the interface.

In order to observe the flow in the cross-stream direction, maps of instantaneous vorticity with its velocity vector fields are given in figure 4.20. A logarithmic scale was used in order to better see the variation of vorticity magnitude in both phases as the vorticity in the liquid phase is up to a thousand times lower than the vorticity in the gaseous phase. The mesh was duplicated four times in the direction of the cyclic width L_z to enhance the visualisation of the vortices. It is seen that large coherent structures are formed along the width and the sharp discontinuity between the two phases is well exposed.

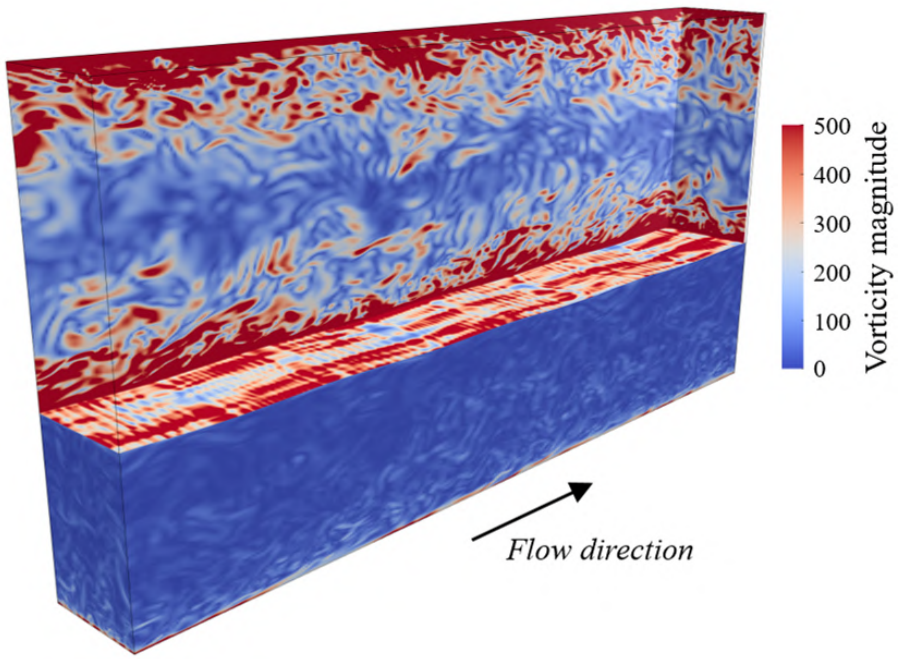


Figure 4.19 Contours of vorticity magnitude and liquid phase fraction on a 3D view of the cyclic channel with the refinement level s1

Conclusions on the qDNS results using the VOF method

Quasi-DNS of turbulence in a shear-driven co-current flow in a rectangular horizontal cyclic channel in a thick film configuration was investigated. The flow presented high discontinuities in all studied quantities in the interfacial region. The qDNS methodology presented in this work was later used to inform the interfacial turbulence in the standard Wilcox's RANS turbulence model in order to improve the modelling of wavy films in aero-engine bearing chambers in the context of industry. Previous work showed some promising results using OpenFOAM with the VOF method for this type of flow using LES with a Smagorinsky subgrid model (Bertolotti et al. [11][12] and section 4.1), thus the same multiphase method was employed with qDNS and mesh refinement with interface sharpening, to represent the averaged velocity, turbulent kinetic energy and shear stress profiles. Results were compared with existing experiments. The averaged velocity profiles obtained numerically showed very good results in

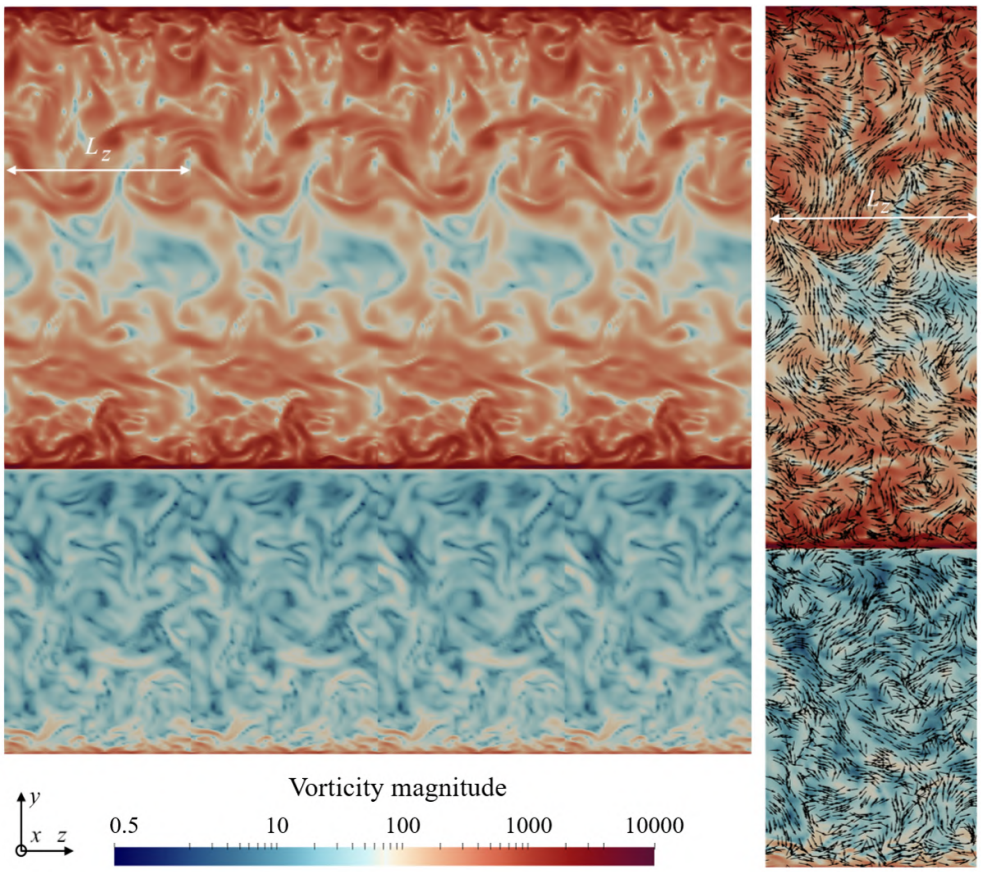


Figure 4.20 Contours of vorticity magnitude on a cross-flow plane (y, z) duplicated 4 times for a total width of 0.1 m (left) and contours of vorticity magnitude with velocity vector field on a single cyclic domain in the cross-stream direction with the refinement level s1

agreement with the experiments. Only very small differences were found in the interfacial region in the gaseous phase where the turbulence was found to be the highest. The turbulent kinetic energy was very well represented too especially in the gaseous phase and the shear stress was also well represented particularly in the interfacial region. A vortex analysis was carried out using Q-criterion and vorticity magnitude and results showed a dissipation of energy from the large turbulent scales localised near the walls and above the interface towards the centre of each phase, supporting the turbulent kinetic energy profiles. Quasi-DNS has given very promising results for this type of shear-driven flow and for reasonable computational times. The feasibility of qDNS simulation for

stratified shear-driven flows was confirmed and the same methodology was later employed to create the training datasets for the machine learning models in the implementation of new improved RANS $k - \omega$ turbulence models. The qDNS seemed to predict less turbulence dissipation than the LES in the wall and interfacial regions, and this was reflected by the qDNS results being closer to the experiments than the LES in the predictions of the mean axial velocity, TKE and cross-stress.

Chapter 5

Proof of concept: implementation of a data-driven turbulence model

In this chapter, the thick-film stratified flow based on the experiments of Fabre et al. [35] investigated in the previous chapter is considered for the implementation of a proof of concept to inform the standard Wilcox's RANS $k-\omega$ turbulence model with high-fidelity simulation data. The aim of this concept is to show that one can produce an appropriate source term to add to the specific turbulence dissipation rate equation ω in order to correct the prediction of the turbulence, especially in the interfacial region where levels of turbulence are overestimated by the model. In the first place, the methodology used to implement the correction source term for the ω equation is presented in section 5.1. Secondly, the concept is applied to the modelling of thick-film stratified flow configurations with the standard RANS $k-\omega$ model in section 5.2, starting with the flow conditions introduced in chapter 4. Finally, a simple machine learning model was implemented in order to test the developed concept in section 5.3.

5.1 Informing the $k-\omega$ model with high-fidelity data

5.1.1 Performance of the standard $k-\omega$ model in two-phase shear flows

As discussed in chapter 3, Frederix et al. [41] and Fan et al. [37] reproduced the Fabre et al. [35] experiments with the Wilocox's $k-\omega$ model. They observed that the turbulence model tends to overestimate interfacial turbulence levels in stratified shear flows because of the presence of high velocity gradients across the interface, which leads to inaccuracies in the prediction of the velocity field. The thick-film case and smooth interface case Fabre 250 investigated in chapter 4 in LES and qDNS was reproduced here using the standard Wilcox's model. Using the VOF method, the RANS simulation was performed on the simplified geometry presented in section 4.2 of chapter 4 with the same cyclic conditions in the length and width of the channel. A mesh containing 25992 cells and providing the wall distance $y^+ < 1$ was employed for the simulation. The case characteristics remained unchanged from table 4.1. The vertical profiles of the mean axial velocity, TKE and absolute values of the cross Reynolds stress across the channel predicted by the standard $k-\omega$ turbulence model were compared with the qDNS results obtained with the refinement level s1 and the Fabre et al. experiments [35]. The plots are presented in figure 5.1. Results clearly show that the mean axial velocity predicted by the RANS model is completely shifted towards the upper wall in the gaseous phase in comparison with the reference. This confirms the results obtained with the Euler-Euler method by Frederix et al. [41] and Fan et al. [37] presented in chapter 3. The TKE profile was largely over-predicted by the standard turbulence model by

one order of magnitude in the gas and nearly two in the liquid. Besides, the predictions made by the standard model did not show any discontinuity at the interface, where a decrease of the values should be observed in the liquid phase. This illustrate well the need of interfacial turbulence damping. The Reynolds stress was also over-predicted by the standard model in both phases showing also a shift profile towards the upper wall in the gas.

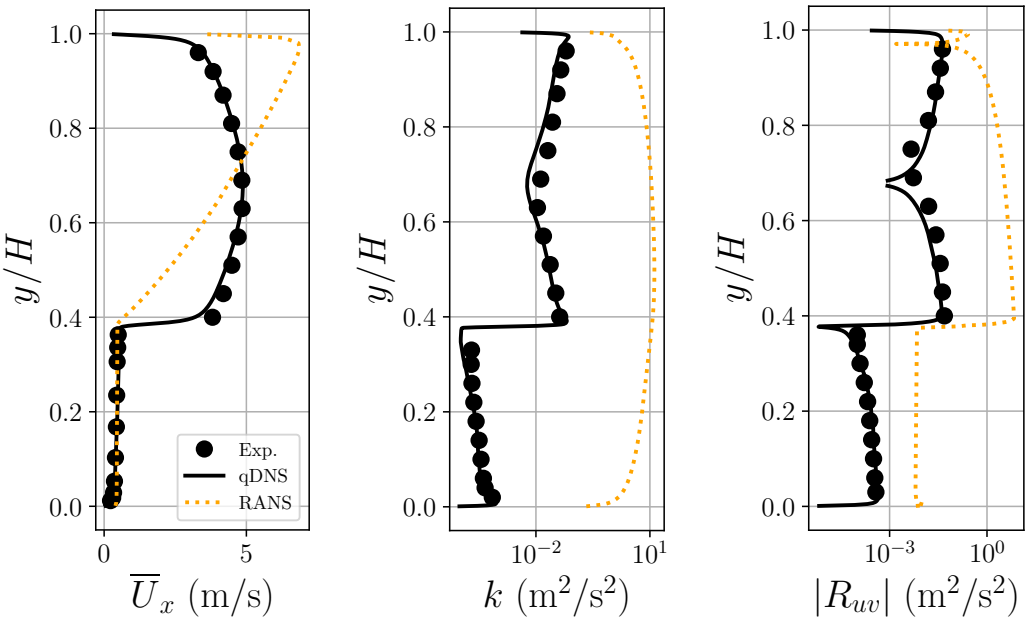


Figure 5.1 Mean axial velocity (left), TKE (centre), and absolute shear stress (right) profiles comparison between Fabre et al. experiments [35], qDNS predictions with mesh s1, and standard RANS $k-\omega$ model predictions

Now that the necessity of a correction of the turbulence production in the standard RANS $k-\omega$ model has been highlighted, a method to re-balance the budget of the specific turbulence dissipation rate ω is suggested next.

5.1.2 Methodology

In this section, a methodology for the correction of the standard Wilcox's RANS $k-\omega$ turbulence model [151] is proposed. The correction can be injected

in one or both of the $k-\omega$ model's equations namely the transport equation of the turbulent kinetic energy k and the transport equation of the specific turbulence dissipation rate ω. According to Singh [118], applying a correction to the ω transport equation may be preferable as both the k and ω terms are present in the transport equation of the specific dissipation rate and $\omega \propto k/\varepsilon$ where ε is the turbulence dissipation rate. It was indeed seen in the literature that most corrections proposed to improve the $k-\omega$ model focused on altering the ω transport equation, which is the case of the widely used Egorov method [32] for instance, in which the turbulence damping term is added to same transport equation.

Correction source term

The correction was introduced in the model ω equation in the form of an additional source term calculated in the high-fidelity qDNS simulation. Similarly to the Egorov method, this additional term acts such as the ω budget is rebalanced and provides informed interfacial turbulence in the $k-\omega$ model. This single correction was found to be adequate and sufficient to drive the RANS solution towards the high-fidelity data. One can rewrite the ω transport equation (eq. 2.26b) for incompressible flow such as:

$$\frac{\partial \rho\omega}{\partial t} + S_{\text{adv}}(\omega, F) = S_{\text{prod}}(\omega, F) - S_{\text{des}}(\omega) + S_{\text{diff}}(\omega, F) \tag{5.1}$$

where:

- $S_{\text{adv}} = \frac{\partial \rho u_i \omega}{\partial x_i}$ is the advection term
- $S_{\text{prod}} = \gamma \frac{\omega}{k} P_\omega$ is the production term
- $S_{\text{des}} = \beta \rho \omega^2$ is the destruction term
- $S_{\text{diff}} = \frac{\partial}{\partial x_i}\left[\left(\mu + \frac{\rho k}{2\omega}\right)\frac{\partial \omega}{\partial x_i}\right]$ is the diffusion term

and where F represents flow field quantities. In order to drive the RANS solution towards the qDNS data, each term from the above description was calculated in the qDNS simulation. This way, a source term S_ω calculated in qDNS and corresponding to an adjusted budget could be introduced in the RANS ω equation. Note that as qDNS velocity quantities were averaged "on the fly" i.e. as the simulation progresses, the local time derivative $\partial\rho\omega/\partial t$ could be taken as 0. Therefore, one obtains the following correction source term to adjust the budget of the specific turbulence dissipation rate ω:

$$S_\omega(\omega, F) = S_{\text{adv}}(\omega, F) - S_{\text{prod}}(\omega, F) + S_{\text{des}}(\omega) - S_{\text{diff}}(\omega, F) \tag{5.2}$$

In order to calculate each term of the S_ω, it was first needed to calculate ω, which is not natively provided for direct numerical simulations in OpenFOAM. The specific turbulence dissipation rate is defined as:

$$\omega = \frac{\varepsilon}{\beta^* k} \tag{5.3}$$

where β^* is a model constant taken at 0.09. It was previously seen in chapter 4 that the turbulent kinetic energy could be calculated in OpenFOAM by calculating the trace of the Reynolds stress tensor, which is obtained while performing the velocity field averaging operation with the function object `fieldAverage`, as shown in appendix A.1.2. The calculation of k was implemented in the simulation control dictionary using a `codedFunctionObject`. The piece of code is available in appendix A.1.2. Following the definition proposed by Hinze [61], the turbulence dissipation rate ε writes:

$$\varepsilon = \frac{1}{2}\nu\overline{\left(\frac{\partial u_i'}{\partial x_j} + \frac{\partial u_j'}{\partial x_i}\right)^2} \tag{5.4}$$

However, ε is not natively calculated by OpenFOAM in qDNS simulations either and had to be implemented. Its calculation consisted of two steps: first, the quantity $q_\varepsilon = \left(\frac{\partial u_i'}{\partial x_j} + \frac{\partial u_j'}{\partial x_i}\right)^2$ was added to the field objects calculated within the VOF solver `interFoam` in the file `createFields.H` as shown in appendix A.1.1; and secondly, the averaging of q_ε was enabled in the simulation control dictionary and ϵ could be calculated using another `codedFunctionObject`, available in appendix A.1.2, such as $\varepsilon = \frac{1}{2}\nu\overline{q}_\varepsilon$.Finally, ω was also added to the coded function objects (c.f. appendix A.1.2) such as $\omega = \varepsilon/(\beta^* k + r_k)$ where r_k was artificially added and taken to be very ($\mathcal{O}(10^{-9})$ to $\mathcal{O}(10^{-10})$) in order to prevent from any division by zero.

The time averaging of the velocity field is activated first, with the field q_ε. Thus, after an averaging period of 10 s, the field ω could be calculated, as well as each term of equation 5.4 in order to obtain the final budget correction source term S_ω. The terms S_{adv}, S_{prod}, S_{des}, S_{diff}, and S_ω were all implemented in the control simulation directory (c.f. appendix A.1.2).

Figure 5.2 shows the profiles of ω, ε, and k after the two averaging operations were performed in qDNS, compared with the standard $k-\omega$ model predictions. While no experimental results are available for the ω and ε profiles, the standard RANS simulation clearly produced over-predicted values for all three quantities in the centre of both of the phases, without picking up the discontinuity at the interface at $y/H = 0.38$, while the qDNS illustrates it very well with peaks in turbulence dissipation near the walls and near the interface. It thus is expected that the correction term calculated in qDNS will add dissipation in those regions, and specifically providing damping of the interfacial turbulence in the RANS $k-\omega$ model.

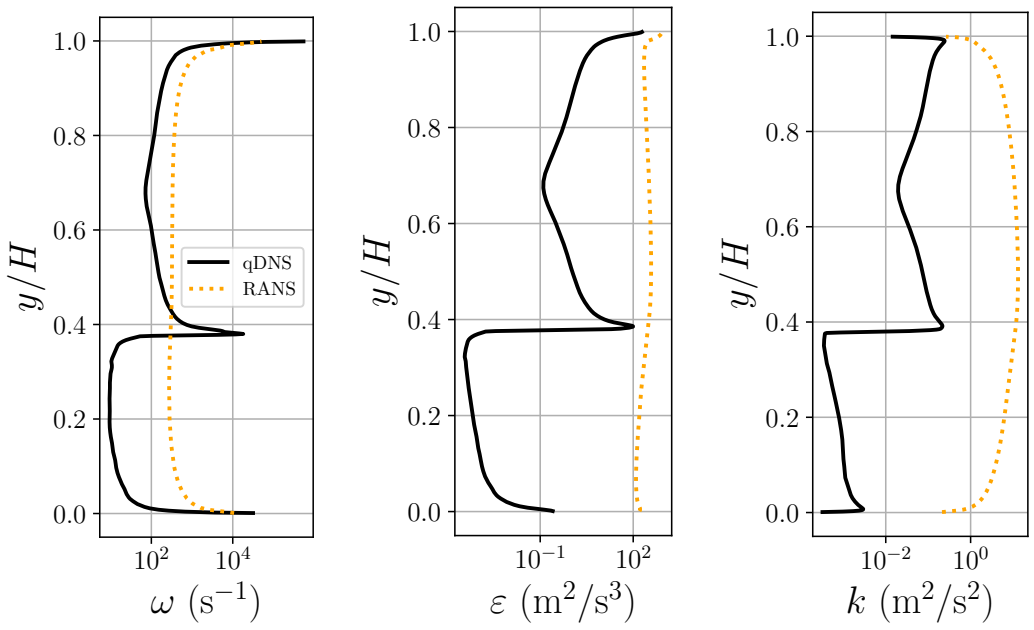


Figure 5.2 Specific turbulence dissipation rate (left), turbulence dissipation (centre), and TKE (right) profiles comparison between qDNS predictions with mesh s1, and standard RANS $k-\omega$ model predictions

Correction for RANS simulations in smooth interface conditions

In order to perform the field correction in the standard $k-\omega$ turbulence model, the budget of specific turbulence dissipation rate calculated in qDNS was injected in the model's ω transport equation as an additional ω source term using the `fvOptions` dictionary. This dictionary also contains the information on the velocity momentum source that is defined prior the simulation in order to set the bulk velocity in each phase, as already described in chapter 4. In order to apply the correction source term, the field S_ω calculated in qDNS was interpolated from the qDNS mesh to the RANS mesh, after all of the average operations were completed. The interpolated values were contained in a file readable by the model, and called in the `fvOptions` dictionary, as shown in appendix A.1.3. The "frozen field" containing the corrected budget of ω values $S_{\omega,i}$ for every cell of the RANS mesh, could then be injected into the transport

equation from the start of the simulation. In this proof of concept, as the correction field S_ω is frozen in time, the additional source term injected in the ω transport equation does not vary in time. This method is thus expected to be most efficient for flows presenting a smooth interface, as it is the case with the Fabre 250 flow regime that is investigated here.

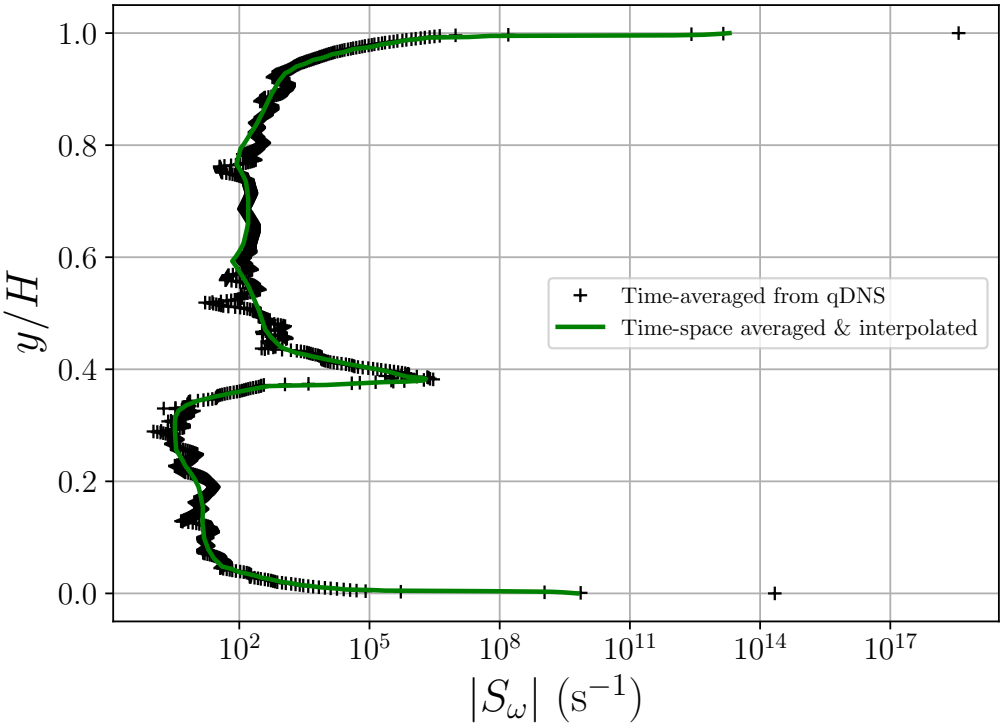


Figure 5.3 Correction source term S_ω calculated and time-averaged in qDNS (black cross) and with additional spatial averaging, mapped on the RANS mesh (green line)

Even with long periods of time averaging, some of the terms still indicated difficulties converging. In order to improve the field average convergence, the correction field S_ω calculated and time-averaged (over 10 s) in qDNS was additionally spatially-averaged in both of the cyclic directions $\vec{x}$ and $\vec{z}$. Therefore, one obtains constant values of S_ω in the two cyclic directions after the two time and spatial averaging operations. Figure 5.3 shows the profiles of the

absolute values of the correction field taken across the channel before and after the space-average and interpolation operations. The final results presented the highest budget correction values in the wall regions and in the interfacial region, which was coherent with the necessity of a turbulence damping in the interface area. Note that the correction source term was added to the transport equation with the negative sign similarly to the destruction term in order to apply turbulence damping to the areas of interest.

5.2 Data-driven thick-film flow turbulence modelling

The methodology described in section 5.1.2 was applied to four test cases in order to test the ability of a "frozen field" correction to drive the model towards the high-fidelity data. The four tests were performed first in qDNS to obtain the appropriate averaged correction field for the ω transport equation, and then in RANS using the additional source term introduced from the qDNS. All of the test geometries were based on the geometry described in chapter 4, section 4.2.1 i.e. a 0.1 m high horizontal channel filled with air and water. The liquid thickness in all the tests was 38 mm corresponding to the thickness employed in the Fabre 250 flow regime. The four tests were performed with the same cyclic conditions in the x and z directions. The first two test cases were performed in the smooth interface condition, obtained with the bulk velocities of the Fabre 250 flow regime: $U_{b,g} = 4.2$ m/s and $U_{b,l} = 0.45$ m/s while the remaining two other tests used a larger liquid bulk velocity $U_{b,g} = 10$ m/s, increasing the gas Reynolds number from $3.5 \cdot 10^4$ to $8.3 \cdot 10^4$. This increase was sufficient to trigger the generation of two-dimensional surface waves of significantly higher

amplitudes than the Fabre 250 flow regime. For each of those flow regimes, two simulations were conducted: the first one with non-slip BC for the top and bottom walls, and the second one with a slip BC for the top wall and non-slip BC for the bottom wall. The two flow regimes are illustrated in figure 5.4 using non-slip BC for both the bottom and top walls. The liquid volume fraction, the instantaneous axial velocity, as well as the vorticity magnitude fields obtained in qDNS are represented. The configuration in which both the top and bottom walls are non-slip BC will be named the "closed channel" configuration, while the configuration using a slip-wall BC at the top wall will be named the "open channel" configuration.

5.2.1 Application to the smooth interface flow regime

In the closed channel configuration, the smooth interface flow regime was performed in qDNS, as previously presented in chapter 4, section 4.2.1 with the refinement level s1. This simulation was used to generate a correction field introduced in figure 5.3, which was applied to the corresponding RANS simulation. Note that in the closed channel configuration, mesh refinements were performed in the wall and interface regions. Using the same geometry base, the open channel configuration employed a slightly different mesh with the same refinement levels to s1 at the bottom wall and in the interface region, although, the slip-wall BC at the top was coarsened as no refinement was needed in this region anymore. This coarsening of the top wall region led to a reduction of the overall mesh size by 20% in comparison with the refinement level s1.

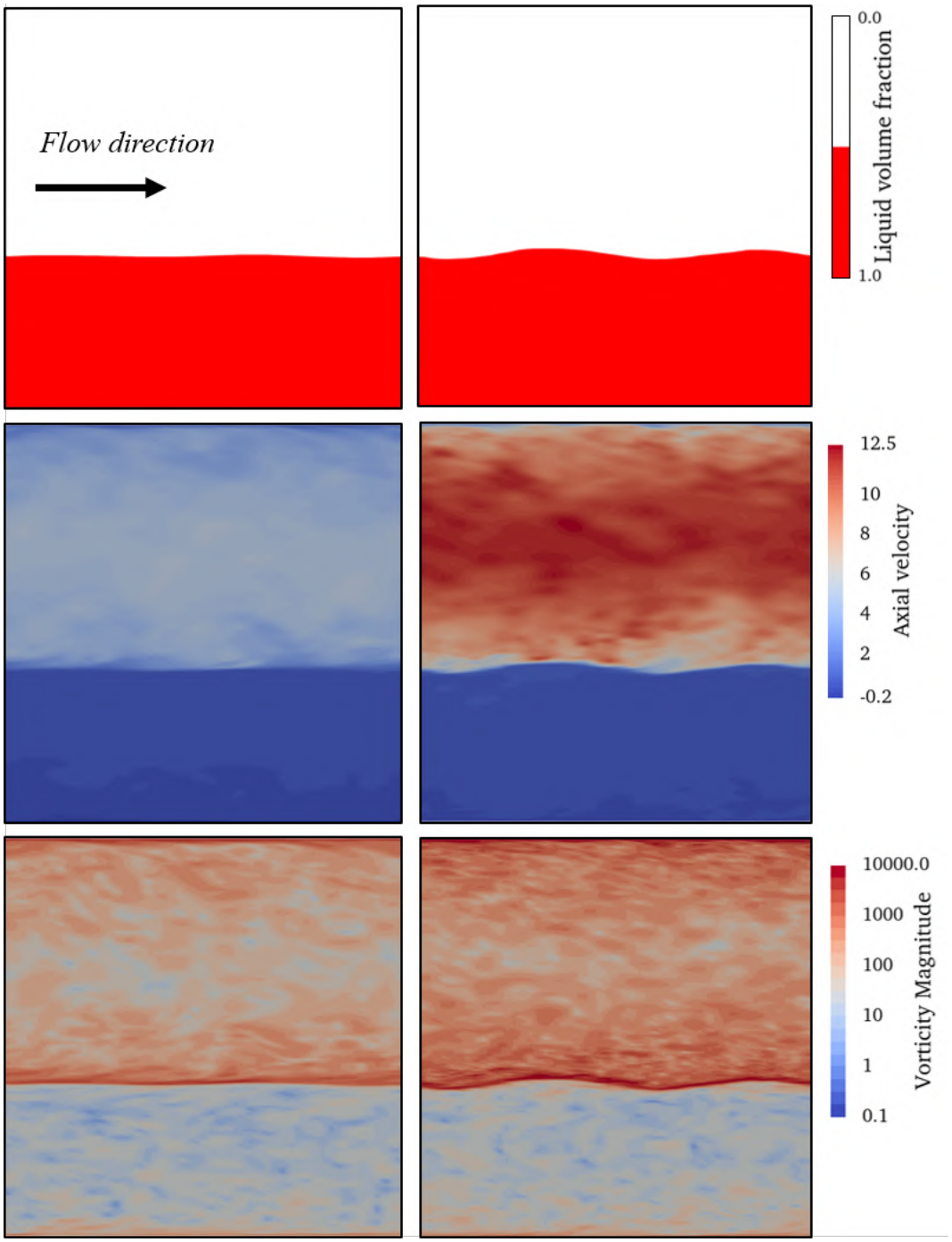
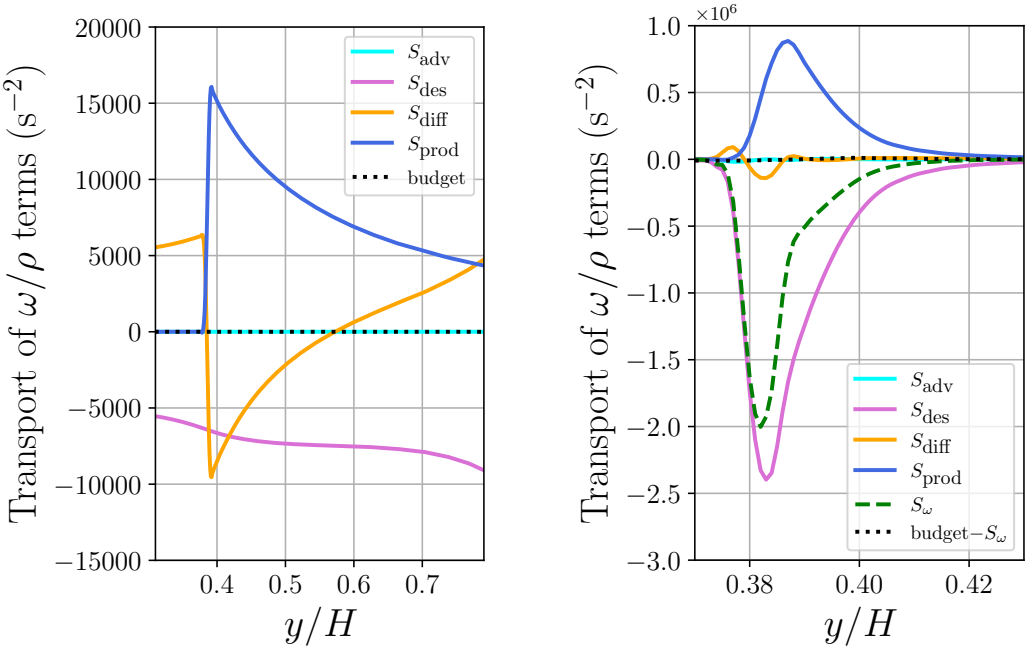


Figure 5.4 Smooth interface regime (left column) compared with the two-dimensional wavy interface regime (right column) in terms of liquid volume fraction (top row), instantaneous axial velocity in $m \cdot s^{-1}$ (centre row), and vorticity magnitude in s^{-1} (bottom row)

Applying the additional source field S_ω led to a correction of the budget of specific turbulence dissipation rate. This budget correction resulted in more turbulence dissipation as intended in order to damp the interfacial turbulence. This adjustment was highlighted in figure 5.5, in which each term of the ω transport equation was plotted across the interface using the standard RANS $k-\omega$ model and the qDNS-informed $k-\omega$ model, in the closed channel configuration.



(a) Standard $k-\omega$ model (b) Corrected $k-\omega$ model

Figure 5.5 Specific turbulence dissipation budget across the interface

The corrected model adapted in increasing its destruction term S_{des}, resulting in an amplification of the interfacial turbulence damping. While no change in the destruction term across the interface is observed in the standard model with values lower than $-1.0 \cdot 10^4$ s^{-2}, the corrected model added a strong gradient of dissipation of energy with destruction values reaching $-2.2 \cdot 10^6$ s^{-2}. However, as the other transport terms produced by the standard model seemed to pick up the interface by showing sharp discontinuities around $y/H \approx 0.38$,

this sharpness was found to be attenuated by the corrected model. This is a consequence of the use of a frozen field correction, whereas the interface is not really "frozen" and still presents some disturbances as shown previously in figure 5.4. As a consequence, some additional turbulence dissipation was involuntary added in the region below the interface, in the liquid phase. From this observation, one could expect to obtain better results using the frozen field method in the conditions of a smooth interface rather than in the conditions of a wavy interface, particularly for the TKE predictions, which might be underestimated in the interfacial region of the liquid phase.

Deploying the frozen correction field method in the $k-\omega$ model provided appropriate results for the fields of interest in comparison to the standard turbulence model. Both configurations namely the closed channel and the open channel were investigated and the corrected RANS $k-\omega$ model produced predictions in very good agreement with the corresponding qDNS simulation as seen in figure 5.6 and figure 5.7.

The addition of a correction source term in the ω transport equation was sufficient to drive the RANS predictions towards the qDNS solutions as it can be seen in the plots of the mean axial velocity, TKE and cross-Reynolds stress. In both configurations, the standard $k-\omega$ model over-predicted the TKE by a factor of at least 10^2 in the two phases in comparison with the qDNS reference. The standard RANS did not predict any discontinuity of the TKE profile in the interfacial region, indicating an important overproduction of turbulence levels in the gaseous phase above the interface. This was passed on to the mean velocity profile, which was shifted upwards in the gas in both the closed channel configuration as previously observed in section 5.1.1, and the open

channel configuration. An underestimation of the TKE levels by the corrected $k-\omega$ model below the interface in the liquid phase was observed in the closed channel configuration and can be explained by the application of a frozen correction field while the interface was not perfectly smooth, as anticipated with the observation of the ω budget in figure 5.5. This underestimation was also slightly reported in the open channel configuration, in which the TKE profile seemed smoother in the interfacial region. The mean velocity profile was also slightly underestimated by the corrected model right above the interface in the gas in the open channel configuration. The shear stress profiles predicted by the corrected model matched the reference very well in both configurations.

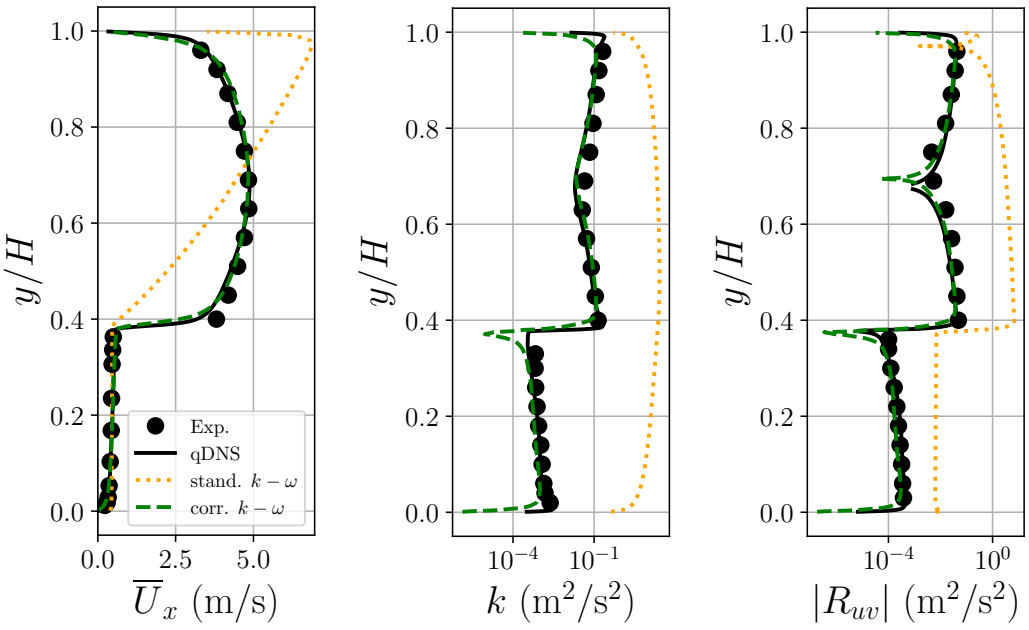


Figure 5.6 Mean axial velocity (left), TKE (centre), and cross Reynolds stress profiles comparison between the qDNS, the standard $k-\omega$ model (stand.), and the corrected $k-\omega$ model (corr.) predictions in the closed channel configuration and smooth interface regime

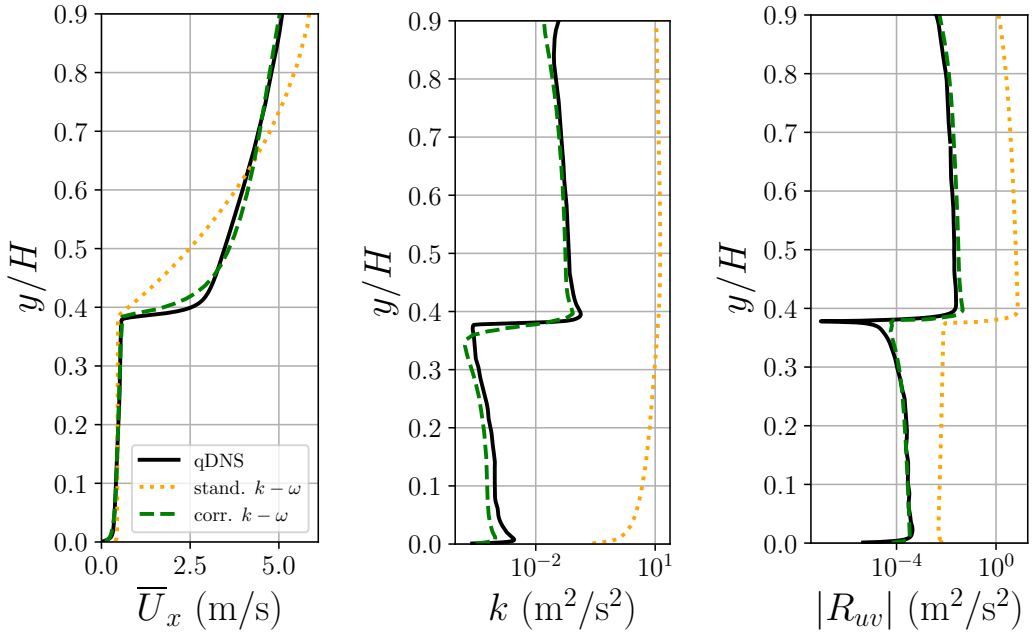


Figure 5.7 Mean axial velocity (left), TKE (centre), and cross Reynolds stress profiles comparison between the qDNS, the standard $k-\omega$ model (stand.), and the corrected $k-\omega$ model (corr.) predictions in the open channel configuration and smooth interface regime

The predictions of ω and ε are presented in figure B.1 for the closed channel configuration and figure B.2 for the open channel configuration, in appendix B.1. Both ω and ε predicted by the corrected turbulence model were in good agreement with the qDNS.

5.2.2 Limitations of the frozen film correction: application to wavy-films

Similar mesh qualities and refinement levels were employed for the additional two test cases in which a higher gas velocity was simulated. The new bulk velocity of 10 m/s allowed for the generation of small 2D waves, putting us in the wavy-film flow regime. In this regime, one would expect to observe results of poorer quality when using the frozen correction field method with the RANS

$k-\omega$ model, due to the application of additional non-necessary turbulence damping in the liquid phase below the interface.

The predictions made by the standard and corrected $k-\omega$ models were compared with the corresponding qDNS results in figure 5.8 for the closed channel configuration and in figure 5.9 for the open channel configuration.

Overall, the corrected turbulence model performed much better than the standard model when comparing with the qDNS predictions. It was noted that better results were obtained by the corrected model in the open channel configuration as only slight underestimations were observed for the TKE and Reynolds stress profiles in the gaseous phase.

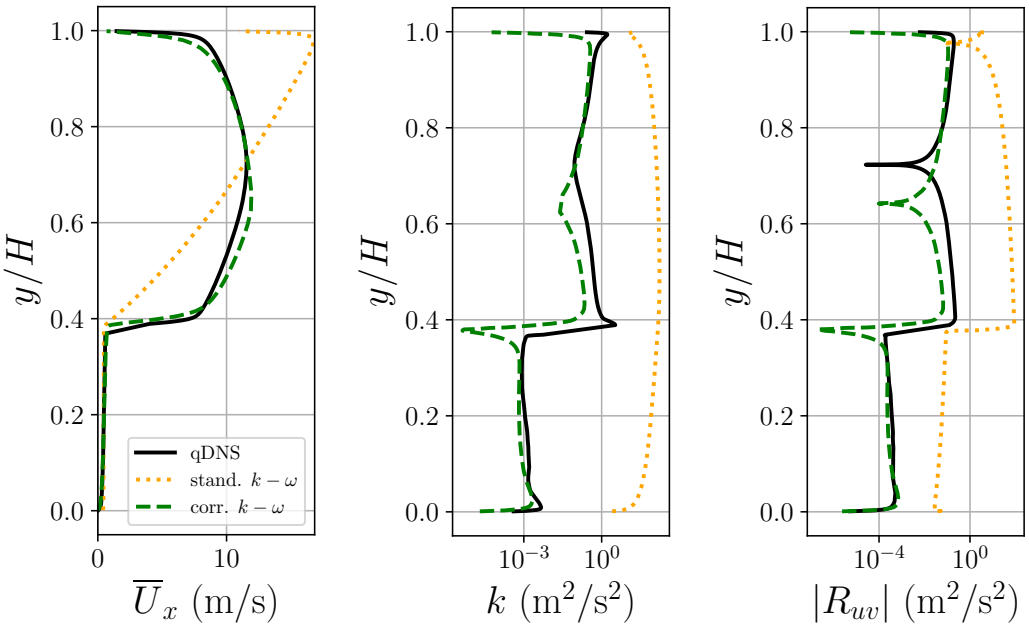


Figure 5.8 Mean axial velocity (left), TKE (centre), and cross Reynolds stress profiles comparison between the qDNS, the standard $k-\omega$ model (stand.), and the corrected $k-\omega$ model (corr.) predictions in the closed channel configuration and wavy interface regime

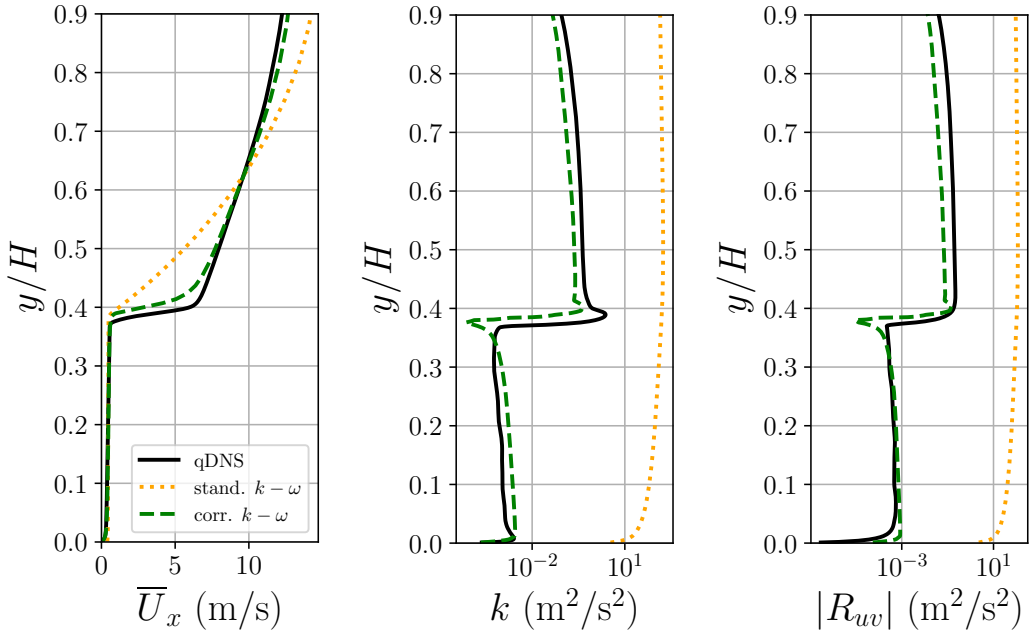


Figure 5.9 Mean axial velocity (left), TKE (centre), and cross Reynolds stress profiles comparison between the qDNS, the standard $k-\omega$ model (stand.), and the corrected $k-\omega$ model (corr.) predictions in the open channel configuration and wavy interface regime

In the closed channel configuration, the underestimation of the TKE levels were more noticeable especially in interfacial region, as well as for the shear stress in the same area. Moreover, while the qDNS predicted asymmetrical TKE and Reynolds stress profiles, which were slightly shifted upwards in the gaseous phase in the closed channel configuration, the corrected RANS predictions were rather shifted downwards the interface. This difference in the lower part of the gaseous phase and the under-prediction in the upper part of the liquid film could be explained by the non-varying character of the frozen correction. In fact, the correction profile should ideally shift upwards as a wave comes (crest) to provide more turbulence damping in the gas and less in the liquid, and reciprocally it should shift downwards as a wave has passed (trough), as illustrated in figure 5.10.

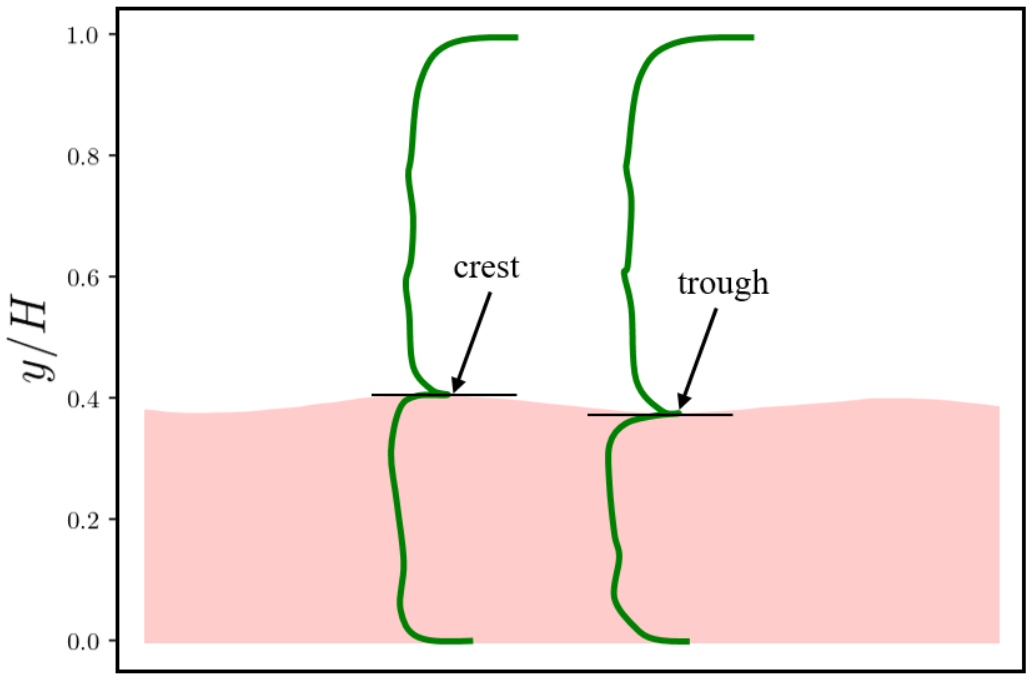


Figure 5.10 Adjustment of the correction field S_ω according to the interface position. S_ω profile is represented in green, the liquid film is represented in light red

In both configurations, the mean axial velocity profiles were well predicted by the corrected model. An underestimation of the velocity was observed in the region just above the interface in the open channel configuration. The predicted profiles of ω and ε are visible in figures B.3 and B.4 of appendix B.1. The corrected model performed overall very well to predict the turbulence dissipation rates, in the two configurations.

5.3 Turbulence correction prediction by a machine learning model

In order to finalise this proof of concept, a simple machine learning (ML) model noted "M1" was implemented using a feed-forward neural network

(FFNN) multilayer perceptron (MLP). The idea was to train the ML model to predict the frozen correction field S_ω prior the beginning of the simulation, given a few number of initial known flow conditions and geometry setup. The predicted correction field is called the output of the ML model, while the flow and geometry features are inputs of the model. In this proof of concept, only the closed channel configuration in the smooth interface condition was tested.

5.3.1 Structure of the ML model

The choice of the inputs has an important impact on the performance of the ML model. They can affect the accuracy and duration of the model training, the accuracy of the model when making predictions, the capacity of the model to adapt to new cases in new flow conditions and flow configurations, etc. In order to make the model more versatile, one can for example use non-dimensional inputs that are based easily accessible flow features. They should represent the flow as a whole. Choosing too many inputs might also create an overfitted model. The model overfitting is usually obtained when the training accuracy is too high, resulting in poor abilities of the model to predict new scenarios as it is only performing well at reproducing the training data [149]. However, for the purpose of this proof of concept, as the training and testing cases are the same, the choice of input did not have to be made regarding the portability of the model, or its overfitting. The model M1 was implemented and used only in the configuration described: closed channel and Fabre 250 flow regime. Four easily accessible input features were selected: a Reynolds number-like noted η_1 based on the phase height h_{phase} in the channel, the liquid volume fraction η_2, the distance from the mean interface level η_3, and the distance from the wall η_4. The one output of the model β was the budget of ω, that is added to the ω transport equation in the Wilcox's RANS

$k-\omega$ model under the form of the source term. Those features are all known and chosen prior starting the simulation. The inputs and output of the model M1 are:

- $\eta_1 = U_b \cdot h_{\text{phase}}/\nu = \alpha U_{b,l} \cdot \bar{h}/\nu_l + (1-\alpha)U_{b,g}(H-\bar{h})/\nu_g$
- $\eta_2 = \alpha_l$
- $\eta_3 = d_{\text{interf}}$
- $\eta_4 = d_{\text{wall}}$
- $\beta = S_\omega$

The simple FFNN implemented for the proof of concept consisted of a first input layer of 4 input neurons, two hidden layers of 256 neurons and one output layer of 1 output neuron. The ReLU non-linear activation function was used. More details on the ML methods are given in chapter 2, section 2.4.

5.3.2 Model training

The training dataset was taken from the qDNS results obtained with the refinement level s1 in the two regimes investigated previously, in the configuration of the closed channel. The model was tested on the configuration of the closed channel in the smooth interface flow regime, which is part of its training. The training dataset was fairly small as each of the training inputs/output consisted of 200 values. Therefore, as one qDNS provided five lists of 200 values, the training dataset contained 2000 values in total. The distribution of the training values for each input and output was represented in histograms, in figure 5.11. For the training of the ML model, the mean squared error function was employed for the loss and 256 epoch were carried out, with

a batch size of 64. The Adam optimiser algorithm was used for our gradient descent optimisation, with a learning rate of 10^{-4}. The data was scaled with the standardisation method described in equation 2.70.

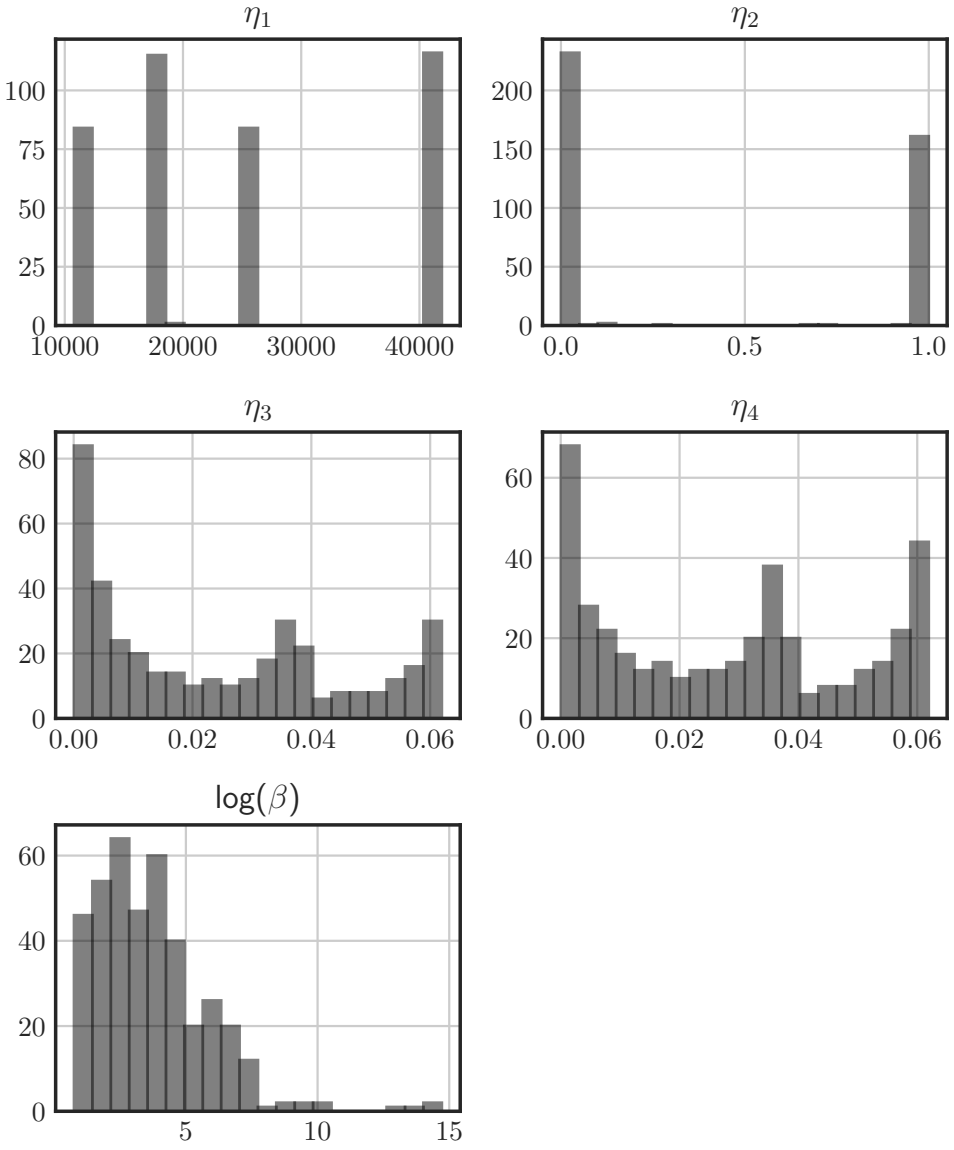


Figure 5.11 Histogram of the inputs and output used in the training of the model M1

In order to improve further the performance of the model training, the logarithm of the output S_w was taken during the training in order to better

homogenise the data. S_ω contains values varying from approximately 10^0 to 10^{13}, and without this operation, the model would have struggled to perform the gradient descent over such an non-homogeneous dataset, even after standard scaling.

The dataset was split randomly such as 80% of its data was used for the training of the model, 20% for the validation. The training reached a prediction accuracy of 97.3%, which was intentionally set high in order to overfit the training data. One would expect from such a training accuracy that the model M1 will be fully capable of reproducing the correction source term calculated in qDNS. Figure 5.12 illustrates the training process, showing the loss and accuracy progress against the number of epochs carried out. The best model was obtained at the 249^{th} epoch. According to figure 5.12, good accuracy values were attained quickly.

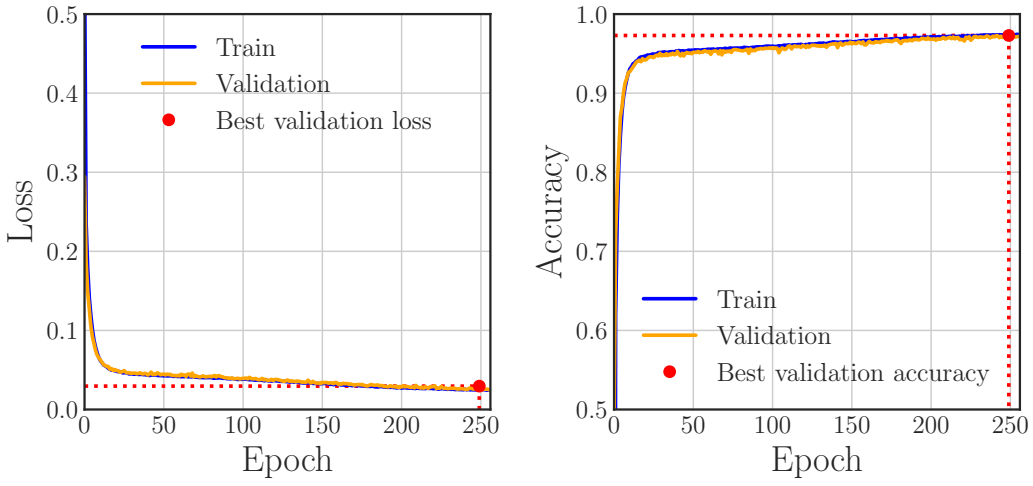


Figure 5.12 Training and validation loss (left) and accuracy (right) against the number of epochs obtained during the training of the ML model M1

5.3.3 ML model predictions

The simulation setup employed for the test of the implemented model M1 is the same as the one introduced in section 5.1.1, also referred as the closed channel configuration in the smooth interface flow regime.

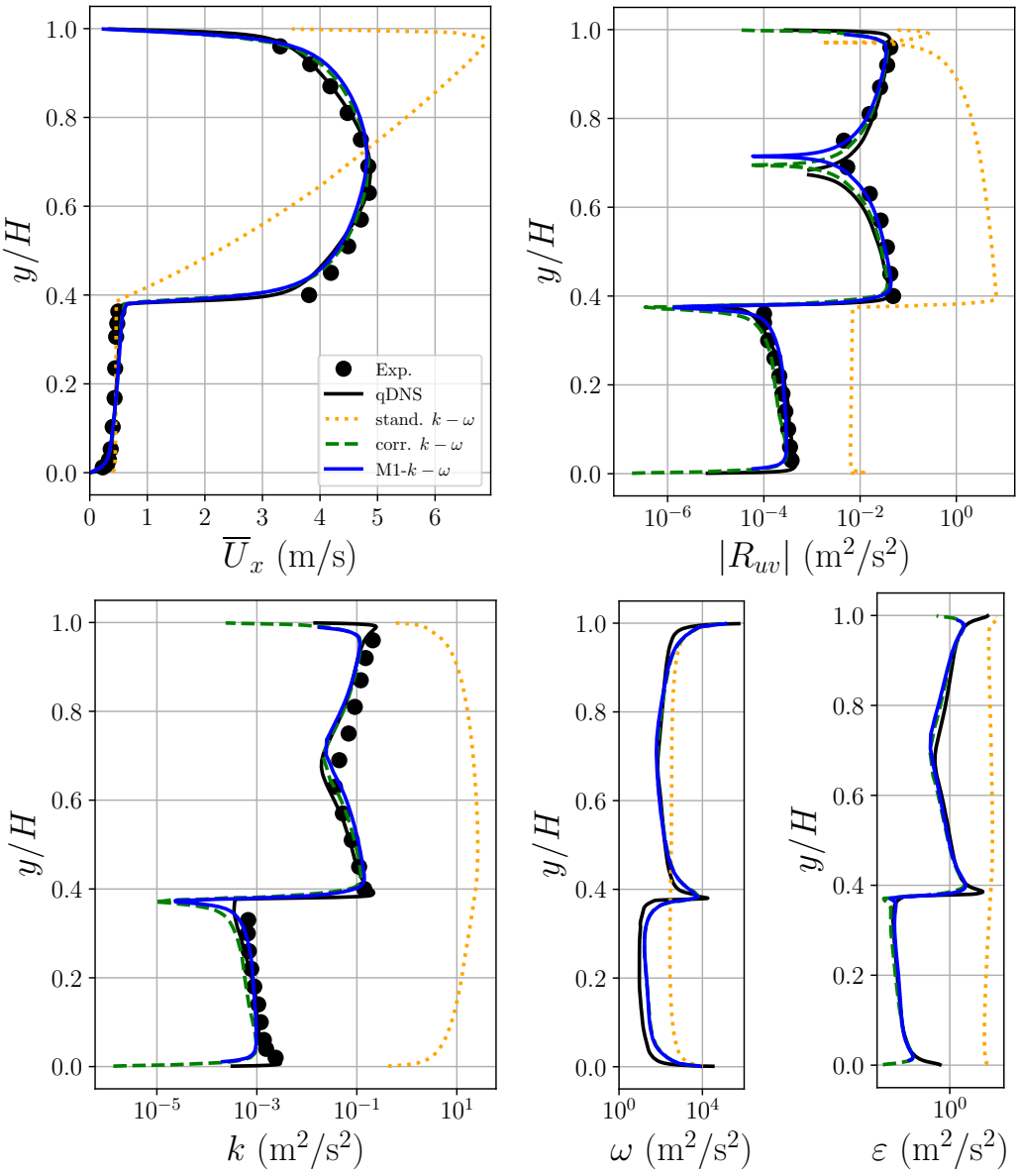


Figure 5.13 Profile predictions obtained with the ML-informed $k-\omega$ model (blue line), the former corrected $k-\omega$ model (green dash line), the standard $k-\omega$ model (orange point line) and the qDNS (black line) in the closed channel configuration, smooth interface regime

Once the machine learning model was trained and saved, the RANS simulation was initialised in order to produce the necessary input features for the prediction of the frozen correction field S_ω. After the prediction was made by the model M1, the predicted correction field was written in the relevant location to be read as a source term the same way as presented in the previous section 5.2.

The results obtained with the ML-informed $k-\omega$ model were compared with the previous results in figure 5.13. As expected, the trained model managed to predict a correction field that was able to drive the informed turbulence model's predictions towards the qDNS solutions, similarly to the method introduced in the previous section 5.2. No important difference is noticeable except in the prediction of the shear-stress profiles where the ML-informed turbulence model produced results closer to the experiments and farther from the qDNS than the former corrected model.

5.4 Conclusions and discussions

This chapter introduced a methodology to inform the standard RANS $k-\omega$ model's interfacial turbulence and more specifically to adjust the budget of the transport of specific turbulence dissipation rate ω by adding a correction source term calculated in high-fidelity simulation. The additional source term was here applied as a frozen correction field and results showed that this correction achieved to drive the model predictions towards the reference solutions for all the quantities observed, namely the mean velocity, the turbulent kinetic energy, the cross-shear stress, the specific turbulence dissipation rate, and the turbulence dissipation rate. Two channel configurations and two flow regimes were tested for a total of four test cases: the closed channel and open channel

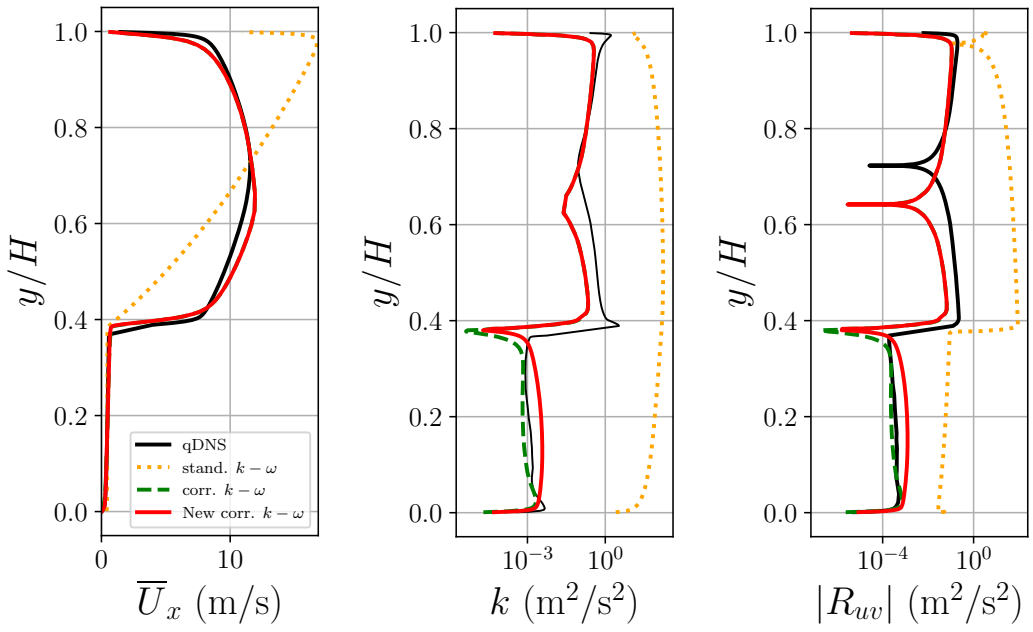


Figure 5.14 Mean axial velocity (left), TKE (centre), and cross Reynolds stress profiles predictions compared between the qDNS, the standard $k-\omega$ model (stand.), the corrected $k-\omega$ model (corr.) and the corrected $k-\omega$ model without correction in the liquid denoted as "New", in the closed channel configuration and wavy interface regime

configurations, and the smooth interface and wavy film flow regimes. Very satisfying results were obtained with the corrected model for both channel configurations in the smooth interface condition, and great improvements were shown in comparison to the standard model's predictions. Increasing the flow rate resulted in overall much better predictions than the standard model. However the corrected model started to show some weaknesses from its under-prediction of the TKE and Reynolds stress in the interfacial region, due to the lack of adaptation of the frozen correction field in wavy film conditions.

In order to account for the under-prediction of those quantities in the liquid interface region, a first simple patch would consist in cancelling the correction field in the liquid phase and let the standard model take over in this phase. This was implemented and tested in the wavy film regime, and in the closed

channel configuration in which the under-predictions were the most noticeable. Results are shown in figure 5.14. Despite the slight improvement in the estimation of the TKE and shear-stress in the interfacial region using the new corrected model, both quantities were overestimated in most the liquid phase in comparison with the qDNS predictions. Elsewhere, the new corrected model predicted identical solutions as the unaltered corrected model.

The treatment of the interface has proven to be a difficult task in waviness conditions. Applying a frozen field to the Wilcox's model resulted in augmented predictions in RANS, however, improvements were still needed regarding the way the correction was applied. The necessity of an adaptive correction was highlighted in this chapter, and the next step of this research was to find a way to update the correction field "on the fly" in time, as often as needed, regarding the wave period of the simulated flow regime.

Finally, a simple machine learning model employing a FFNN trained with the two qDNS cases performed in the closed channel configuration was implemented in order to reproduce the results obtained in the closed channel configuration for the smooth interface flow regime. The model proved to be able to produce nearly identical results, demonstrating that it was capable of predicting the appropriate frozen field correction as it was taught to. This completed the proof of concept, showing that a machine learning can be potentially trained by high-fidelity data from a range of different flow conditions in order to predict appropriate corrections for the budget of specific turbulence dissipation rate in the $k - \omega$ turbulence model.

In the next chapter, thin-film flows were investigated in gas regimes that triggered the generation of waves of lower or higher amplitudes and frequencies. A novel methodology based on the introduction of a correction source term in the ω transport equation was developed in order to account the strong variations of the interface level. This methodology relied on an adaptive correction of the ω equation predicted by a newly trained ML model producing updated corrections as the simulation progresses.

Chapter 6

Application to thin-film two-phase channel flows

Predicting thin-film flows is particularly important in the framework of bearing chambers: it is part of a larger work scope concerning the aero-engine's thermal management. As demonstrated by the experiments of Hann et al. [54] and Kim et al. [76], the superficial gas velocity significantly impacts the liquid film thickness in co-current stratified shear flows. Moreover, standard RANS models, such as the $k-\omega$ model, perform poorly when it comes to predict the superficial gas velocity (c.f. chapter 5, section 5.1.1). Therefore, developing new methods for their improvement is essential, as the liquid film thickness prediction plays a major role in anticipating the oil distribution in the lubrication system of the aero-engine by the bearing chamber, as well as predicting risks of oil burning and coking in the bearing chamber itself.

In this final chapter, the methodology developed in chapter 5 was employed in order to inform the $k-\omega$ turbulence model's interfacial turbulence in thin-film channel flows. Two machine learning models were implemented and trained

with high-fidelity qDNS simulations presented in section 6.1, which setups were based on the experimental work of Hann and Kim [54][76] investigating thin-film stratified flows in a horizontal and rectangular closed channel containing water for the liquid phase and air for the gaseous phase (c.f chapter 3, section 3.1.1). The results of a first ML model "M2" based on the frozen correction field method and employed with the $k-\omega$ model are presented in section 6.2. Then, an adaptive correction method was developed for a new ML model "M3", which was implemented and coupled with the $k-\omega$ turbulence model. Its results are presented in section 6.3.

6.1 Simulation setup and creation of the training dataset

6.1.1 Geometry and flow characteristics

Thin-film flow reference experiments

Hann and Kim [54][76] carried out a range of experiments on thin-film flows in a 2 m long, 0.026 m high and 0.166 wide horizontal and rectangular channel. Despite the large aspect ratio width/height of this channel i.e. more than 6, it was observed that the presence of the side walls had an impact on the waves' dimensionality, allowing for the generation of 3D waves in high flow rates, while 2D waves were observed in the lower regimes. A map identifying the wave patterns in the experiments and based on the flow pattern map of Andritsos and Hanratty [5] is presented in figure 6.1, in which $U_{b,l}$ stands for the liquid bulk velocity and $U_{b,g}$ the gas bulk velocity.

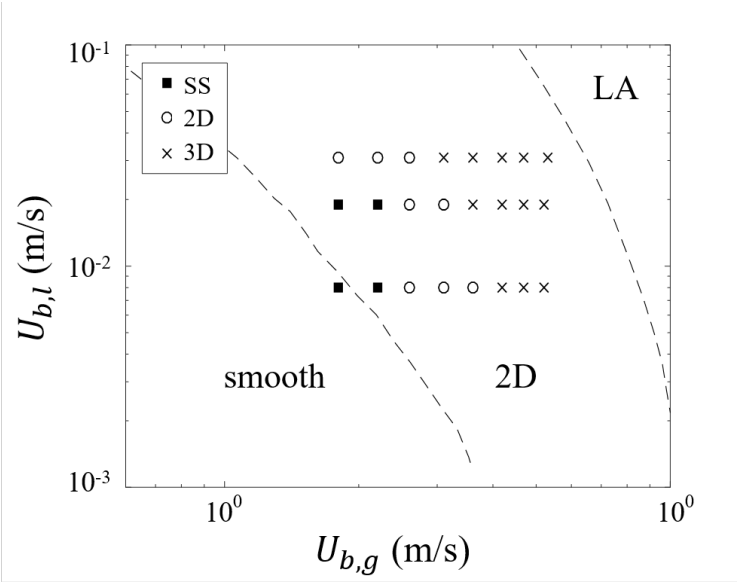


Figure 6.1 Map of flow patterns identified by Andritsos and Hanratty [5]: smooth interface (smooth), 2D waves (2D), and large amplitude waves (LA), overlaid with the experimental conditions of Hann and Kim [54][76] (marked symbols) identifying wave patterns: stratified smooth (SS), 2D small amplitude (2D), and 3D small amplitude(3D)

Hann and Kim investigated a range of 24 flow conditions, including 3 different liquid bulk velocities $U_{b,l}$: 0.008 m/s, 0.019 m/s, and 0.031 m/s and 8 different gas bulk velocities $U_{b,g}$ for each liquid flow regime ranging from 3.1 m/s to 5.2 m/s. The high-fidelity qDNS simulations used to train the developed machine learning models of this chapter were based on the 15 most turbulent flow regimes investigated by the experimental researchers i.e. 5 cases for each of the 3 liquid flow regimes. The 15 cases used as a reference for the qDNS simulations are shown in table 6.1. Every studied case presented different film thicknesses ranging from 8.9% to 24.0% of the channel height. The thinnest films were investigated in the lowest liquid flow regime ($U_{b,l} = 0.008$ m/s).

Table 6.1 Flow regimes description and names

$U_{b,g}$ \ $U_{b,l}$	0.008 m/s	0.019 m/s	0.031 m/s
3.1 m/s	1.a	2.a	3.a
3.6 m/s	1.b	2.b	3.b
4.2 m/s	1.c	2.c	3.c
4.7 m/s	1.d	2.d	3.d
5.2 m/s	1.e	2.e	3.e

Computational domain, simplifications and flow characteristics

Similarly to the computational domain's simplifications made in the thick-film case that was investigated previously in chapters 4 and 5, the channel of the reference experiments was simplified by the use of periodic conditions in the flow stream and cross-stream directions. As many qDNS simulations were needed to build a sufficiently large dataset for the training of the machine learning models, employing periodic conditions allowed for important savings in terms of computational resources. The present study used similar case configurations as presented previously in figure 4.1, chapter 4. Non-slip wall boundary conditions were used at the bottom and top of the channel, and periodic conditions were set in the x and z directions of the channel. The periodic length was set to 0.04 m in the flow stream ($\vec{x}$) and 0.008 m in the spanwise direction ($\vec{z}$). The same height as the experiments was used.

In order to prevent from any unphysical results in qDNS, the periodic simplifications were backed up by the computations of the fluctuation velocity autocorrelations in the two periodic directions, using measurement probes in the

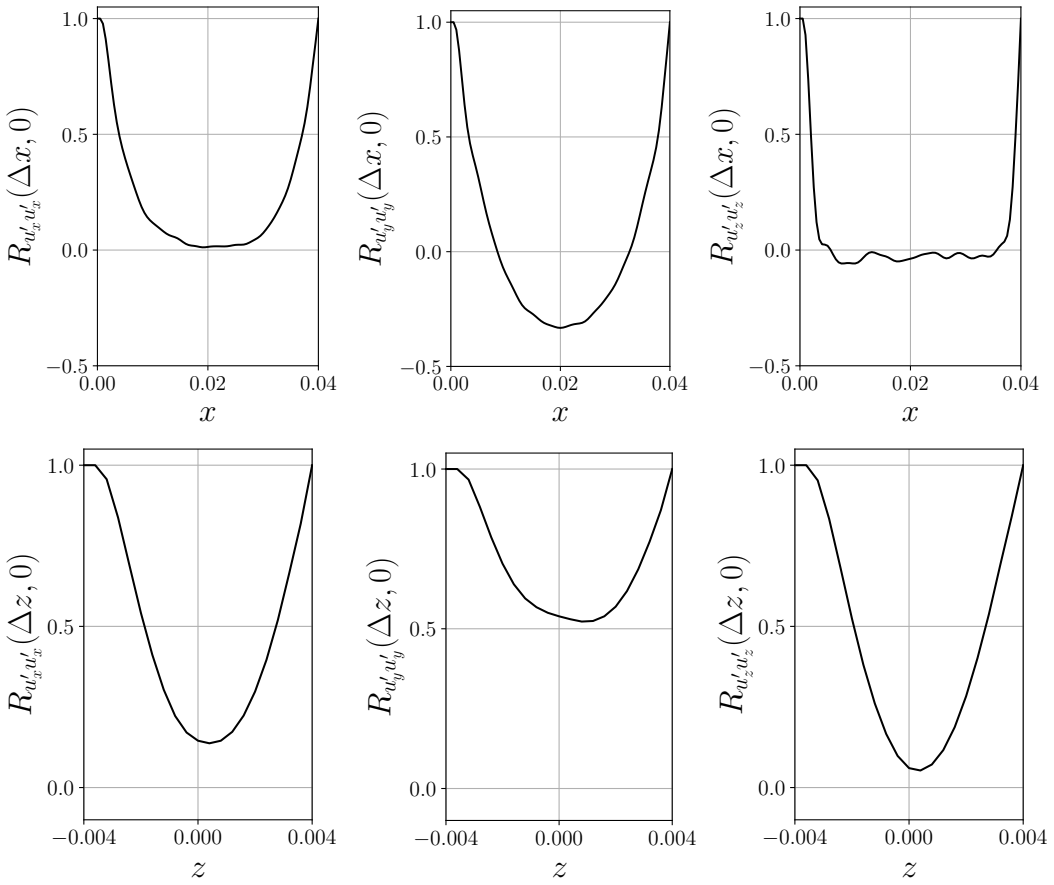


Figure 6.2 Spatial autocorrelations of the x, y and z-components of the fluctuation velocity measured in the centre of the gaseous phase in the flow direction (top) and in cross-flow direction (bottom)

centre of the gaseous phase, where the largest turbulent structures are observed. Figure 6.2 shows the spatial autocorrelation of the axial fluctuation velocity averaged over 10 seconds. According to the plots of the spatial autocorrelation in the streamwise direction $R_{u'_i u'_i}(\Delta x)$, the periodic length was found to be sufficient as all quantities dropped to zero at or before the half of the length. In the spanwise direction, very small values were also observed halfway through the width of the channel, indicating a decorrelation of the flow.

The flow conditions with corresponding ratios of film thickness over channel height r_{film}, and Reynolds numbers based on the hydraulic diameter in the

liquid phase $Re_l = 4\bar{h}U_{b,l}/\nu_\mathrm{l}$ and in the gaseous phase $Re_g = 2(H-\bar{h})U_{b,g}/\nu_\mathrm{g}$, are presented in table 6.2. While all of the gaseous Reynolds numbers indicated a turbulent gas flow, the liquid film flow conditions were laminar [21]. Despite the fact that only one of the two phases was turbulent, waviness is triggered by the gas shear entrainment. The investigated cases employed gas bulk velocities 100 to 650 times higher than the liquid bulk velocities, creating high velocity gradients and strong shear stresses across the interface, highlighted by a discontinuity of the vertical profiles of all flow quantities at the interface level. As the $k-\omega$ turbulence model was not design to predict laminar flows, the results presented in this chapter focus on the gaseous phase only, using the scaling $(y-\bar{h})/(H-\bar{h})$ where y is the vertical position, $\bar{h}$ the mean interface level or mean film thickness, and H the channel height.

Table 6.2 Description of the flows conditions of the 15 qDNS cases

Bulk vel.	$U_{b,l} = 0.008$ m/s			$U_{b,l} = 0.019$ m/s			$U_{b,l} = 0.031$ m/s		
$U_{b,g}$	Re_g	Re_l	r_film	Re_g	Re_l	r_film	Re_g	Re_l	r_film
3.1 m/s	$9.39{\cdot}10^3$	118	13.8%	$9.01{\cdot}10^3$	346	17.3%	$8.52{\cdot}10^3$	700	24.0%
3.6 m/s	$1.10{\cdot}10^4$	112	13.2%	$1.06{\cdot}10^4$	330	16.5%	$1.00{\cdot}10^4$	668	21.8%
4.2 m/s	$1.32{\cdot}10^4$	82	10.3%	$1.28{\cdot}10^4$	266	13.2%	$1.23{\cdot}10^4$	526	16.4%
4.7 m/s	$1.49{\cdot}10^4$	76	9.5%	$1.44{\cdot}10^4$	252	12.5%	$1.39{\cdot}10^4$	498	15.5%
5.2 m/s	$1.66{\cdot}10^4$	72	8.9%	$1.61{\cdot}10^4$	236	11.7%	$1.59{\cdot}10^4$	462	14.4%

Each of the 15 cases simulated in qDNS employed a mesh size of 320000 cells. The meshes were refined near the walls providing the wall distance $y^+ < 1$ and around the interface in order to capture the small turbulent scales and maintain physical results by using wall and interface cell sizes smaller than five times the Kolmogorov length scales. 200 cells were employed to discretise the domain vertically in all cases, using different mesh refinements adapted

to the artificially set interface levels. In the liquid phase, Δy was taken at approximately $\eta/12$ at the walls and $\eta/8$ at the interface, while in the gaseous phase Δy was η at the top wall and 2η at the interface. In terms of wall units, $\Delta y^+ \approx 0.3$, $\Delta x^+ \approx 8$ and $\Delta z^+ \approx 6$.

6.1.2 Quasi-DNS simulations and comparison with experiments

All qDNS simulations were carried out using the VOF method, and the numerical discretisation schemes and solution algorithms employed in the previous chapters, introduced in chapter 4, section 4.1.2. Similarly as before, each phase's bulk velocity was set to its value by use of a velocity momentum source. For each of the 15 cases, the budget of transport of the specific turbulence dissipation rate was calculated following the methodology developed in chapter 5. This term noted S_ω was part of the training dataset in both the new machine learning models implemented in the present chapter. It was employed as the output of the developed machine learning models. Figure 6.3 shows the mean axial velocity $\overline{U_x}$ and ω transport budget S_ω vertical profiles across the channel obtained in the 15 simulated cases.

It was observed from the budget plots that the highest values were found near the top wall (up to 10^{13} s^{-2}), the bottom wall (up to 10^6 s^{-2}) and in the interfacial region (up to 10^9 s^{-2}). According to the results obtained in chapter 5, those locations correspond to regions where the turbulent kinetic energy reaches its highest levels. Applying the budget correction S_ω to the ω transport equation of the $k-\omega$ model would consequently result in an increase of the turbulence dissipation in those areas, especially in the interface region, and

thus would lead to the desired damping of the interfacial turbulence levels.

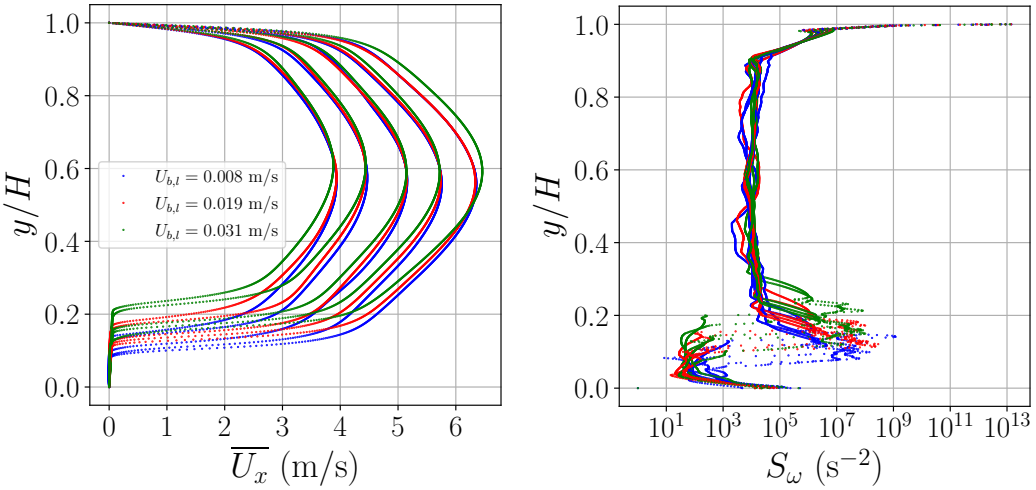


Figure 6.3 Range of mean axial velocity and correction source term profiles obtained in qDNS in cases "1." ($U_{b,l} = 0.008$ m/s), "2." ($U_{b,l} = 0.019$ m/s), and "3." ($U_{b,l} = 0.031$ m/s)

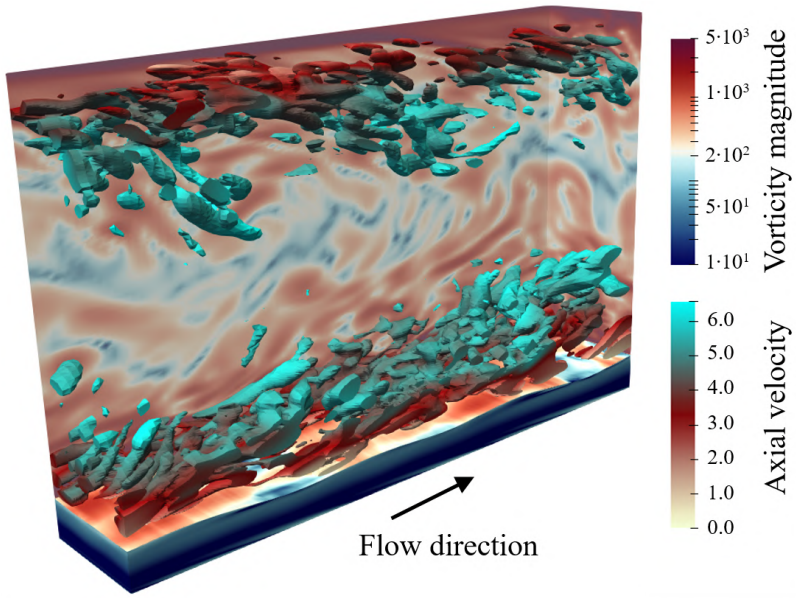


Figure 6.4 Q-criterion isosurfaces with axial velocity contours & vorticity magnitude contour map obtained in qDNS, case 1.e

Figure 6.4 highlights the highest turbulence levels across the channel with the Q-criterion isosurfaces and a map of the vorticity magnitude obtained in qDNS in case 1.e. Once again, the figure clearly identified the above-mentioned interfacial and wall regions as locations showing the highest levels of turbulence.

According to table 6.2, case 1.e corresponds to the configuration with the highest Reynolds number ($1.66 \cdot 10^4$), and the smaller film thickness (8.9% of the channel height). Despite figure 6.1 indicates that 3D wave patterns must be observed the flow regime of case 1.e, 2D wave patterns were observed in the corresponding qDNS. Note that only 2D waves were observed in all the 15 cases, as no side walls were used for the qDNS simulations. Figure 6.5 shows side views (x, y) of the instantaneous liquid volume fraction, axial velocity (m/s) and vorticity magnitude (s^{-1}) fields obtained with qDNS in cases 1.a, to 1.e, at the liquid film bulk velocity 0.008 m/s.

The results obtained using the two other liquid film bulk velocities 0.019 m/s and 0.031 m/s are shown in appendix B.2, in figure B.10 for cases 2.a to 2.e, and in figure B.11 for cases 3.a to 3.e. In the lowest film velocity configuration, case 1.a presented quasi-smooth interface patterns. All other cases presented at least small wave amplitude patterns, while the wave amplitudes increased with the increasing gas superficial velocity. The largest wave amplitudes were observed for the highest gas Reynolds numbers and thinnest films. The waves observed in those cases were identified as 3D according to the pattern map introduced in figure 6.1.

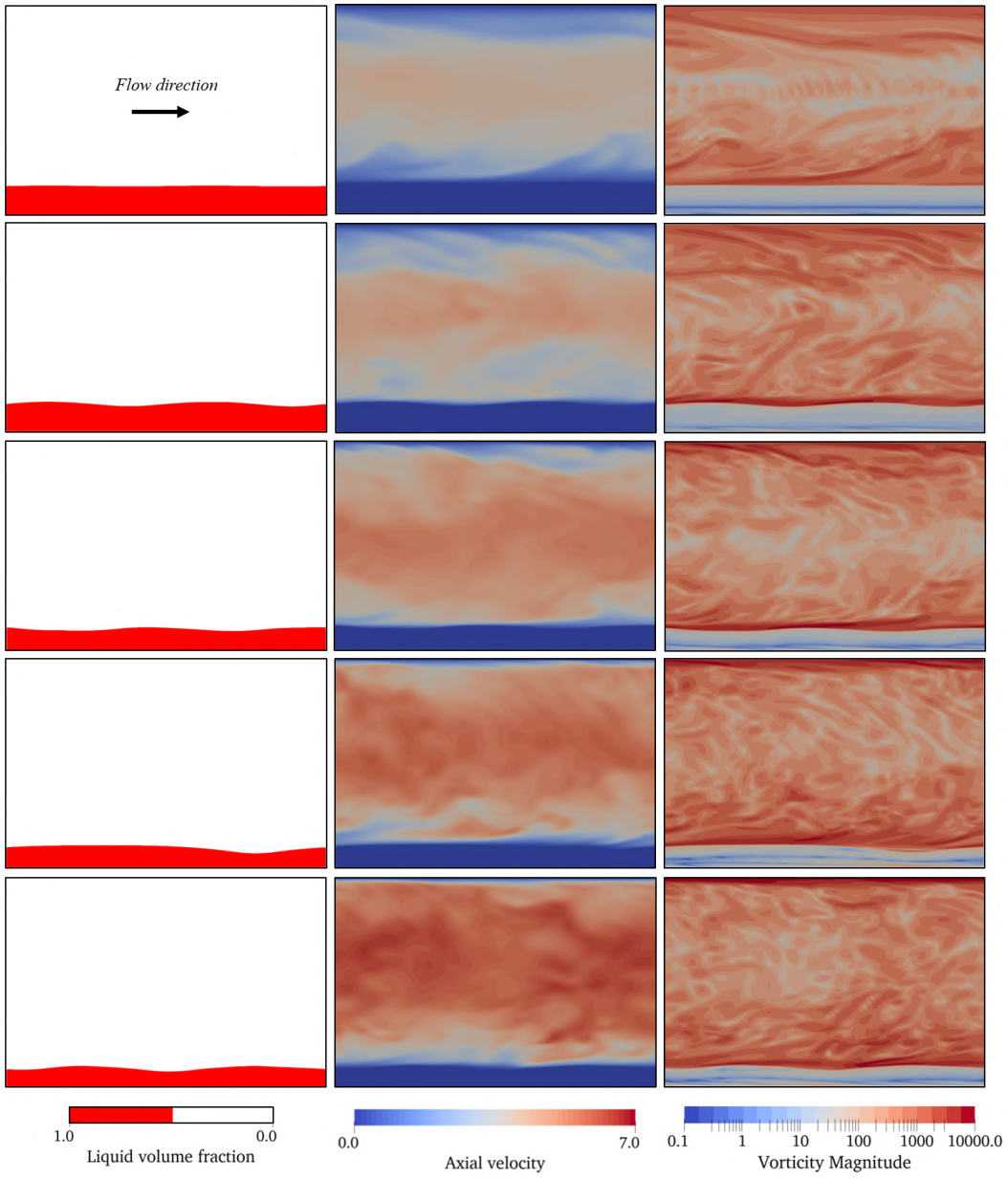


Figure 6.5 qDNS simulations performed at the liquid film velocity 0.008 m/s

Among the 15 simulations, 12 were used for the training of both machine learning models presented in the present chapter. The remaining 3 simulations corresponded to cases 1.e, 2.c and 3.d. The three simulations were kept out of the machine learning training and used as test data for comparison with the results obtained with the ML-informed RANS simulations, which were

carried out in the flow condition of the test cases 1.e, 2.c and 3.d. Those three cases were used for the tests as they presented various flow rates and interface heights belonging to the framework of the training of the ML model. The qDNS data of the test cases were compared with the experiments in terms of mean axial velocity $\overline{U_x}$, Reynolds stress $\overline{u'v'}$, and mean fluctuation velocity $\overline{u'}$ and $\overline{v'}$ profiles. The comparison plot of those quantities in the gaseous phase between qDNS and the experiments are shown in figures 6.6, 6.7, 6.8, and 6.9. In the test cases 1.e, 2.c, and 3.d, the mean axial velocity profiles and mean fluctuation velocity profiles obtained in qDNS matched the experiments very well. The absolute value of the Reynolds stress profiles were found to slightly overestimate the experimental results in the region just above the interface. Such differences might be due to a sharpness diffusion in the velocity discontinuity across the interface. In fact, cases 1.e, 2.c and 3.d showed relatively high amplitude interface wave patterns and the velocity could only be averaged in time.

The mean axial velocity profiles obtained in qDNS in the 12 training cases were also compared with the experimental results and are shown in appendix B.2, in figures B.6, B.7, B.8, B.9. The mean velocity profiles obtained in qDNS in the 12 other cases were also in very good agreement with the experiments. The best fits were obtained for the cases employing the highest gas bulk velocities.

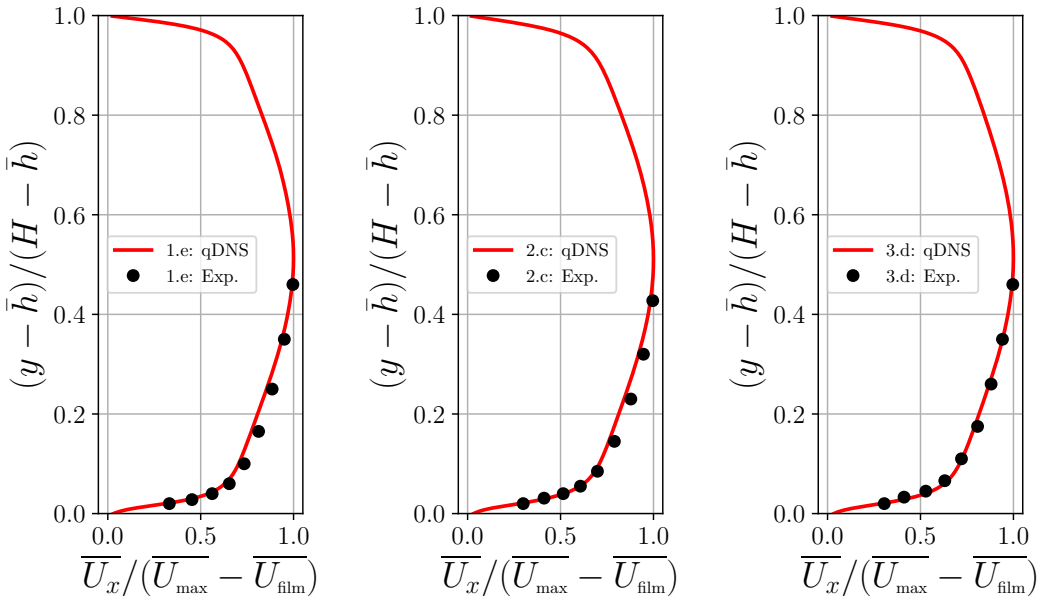


Figure 6.6 Mean axial velocity profiles obtained in qDNS and compared with the experiments in case 1.e (left), 2.c (centre), and 3.d (right)

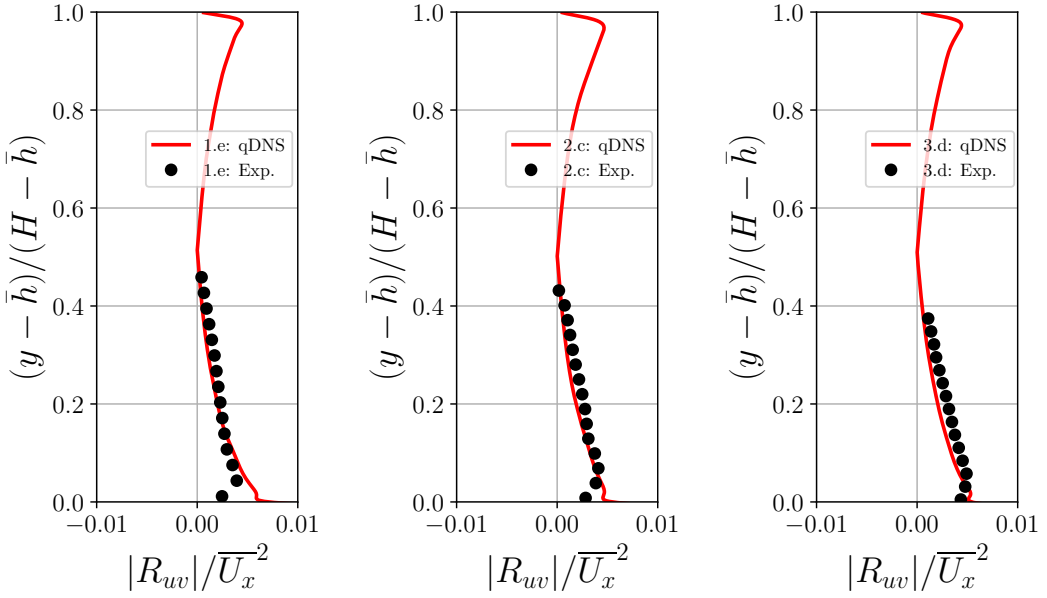


Figure 6.7 Absolute value of the Reynolds stress profiles obtained in qDNS and compared with the experiments in case 1.e (left), 2.c (centre), and 3.d (right)

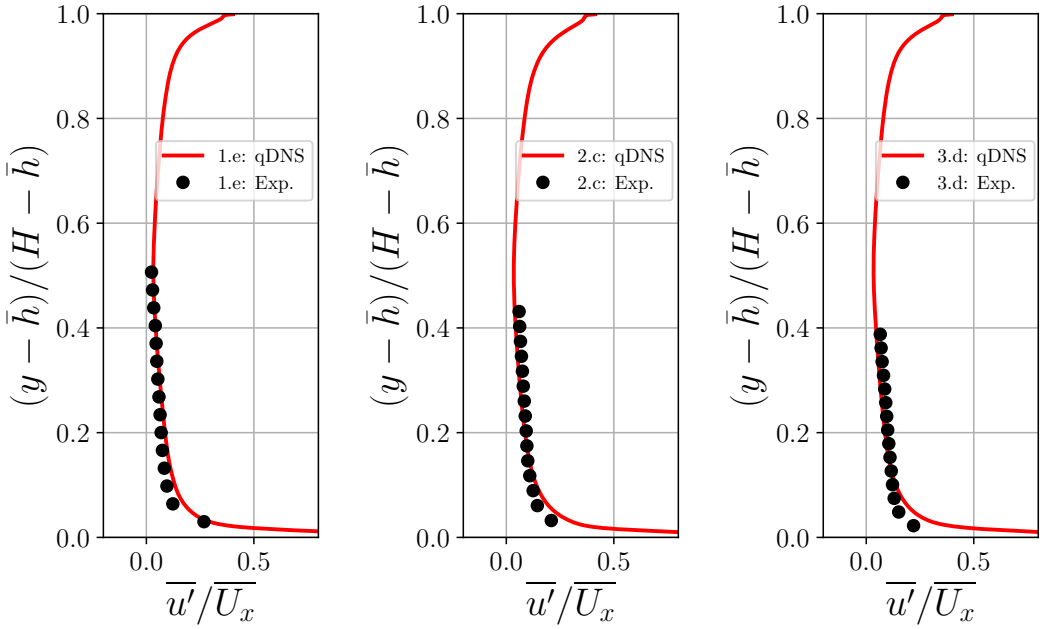


Figure 6.8 Mean axial fluctuation velocity profiles obtained in qDNS and compared with the experiments in case 1.e (left), 2.c (centre), and 3.d (right)

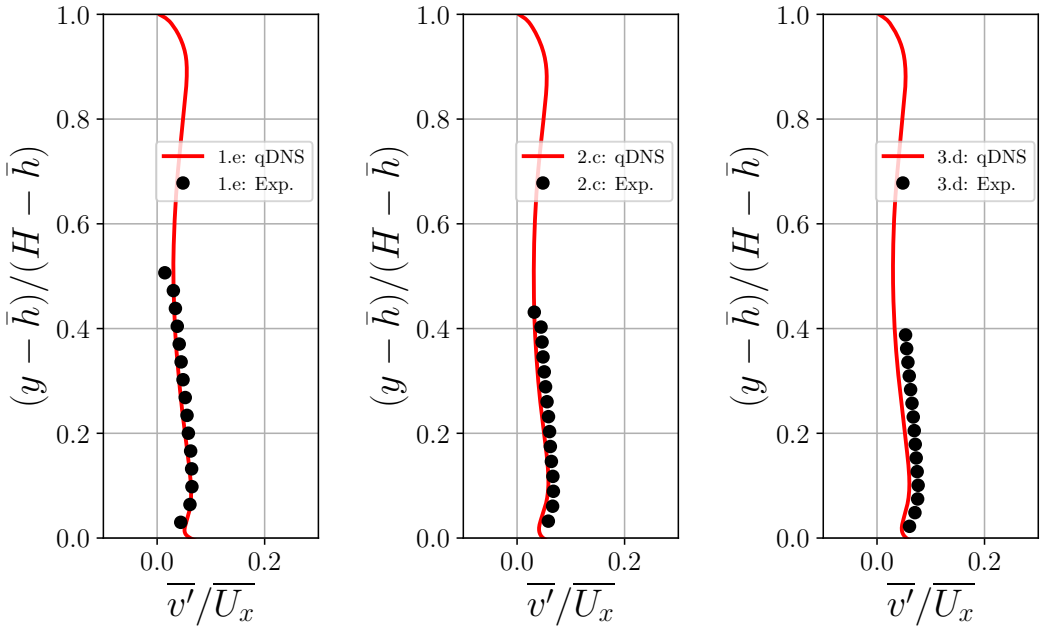


Figure 6.9 Mean vertical fluctuation velocity profiles obtained in qDNS and compared with the experiments in case 1.e (left), 2.c (centre), and 3.d (right)

The high-fidelity qDNS simulations carried out on 15 different flow conditions provided decent detailed datasets for the training of the two machine learning models developed to produce appropriate corrections for the $k-\omega$ turbulence model. As previously mentioned, 12 of the those 15 simulations formed the training dataset of the ML models, while the remaining 3 simulations were not included in the training and used for the testing of the ML model coupled with the RANS turbulence model. The training dataset included the flow conditions of the three different film velocities 0.008 m/s, 0.019 m/s, and 0.031 m/s. The five gas bulk velocities were also represented in the training dataset. The test cases 1.e, 2.c and 3.d were chosen in order to test the ML models using the three film velocities and different gas velocities. They account 20% of the whole qDNS dataset. Indeed, in order to obtain effective ML models the training dataset must include a sufficient number of different flow conditions.

6.2 Frozen correction field method

6.2.1 Process description

Similarly to the method developed in chapter 5, section 5.3, a new machine learning model "M2" was trained using the 12 cases described previously in order to be able to provide a frozen correction field that can be applied to the transport equation of the specific turbulence dissipation rate ω in the $k-\omega$ turbulence model. However, this time the training cases and the testing cases were separated so that the machine learning model M2 could provide predictions in new flow conditions.

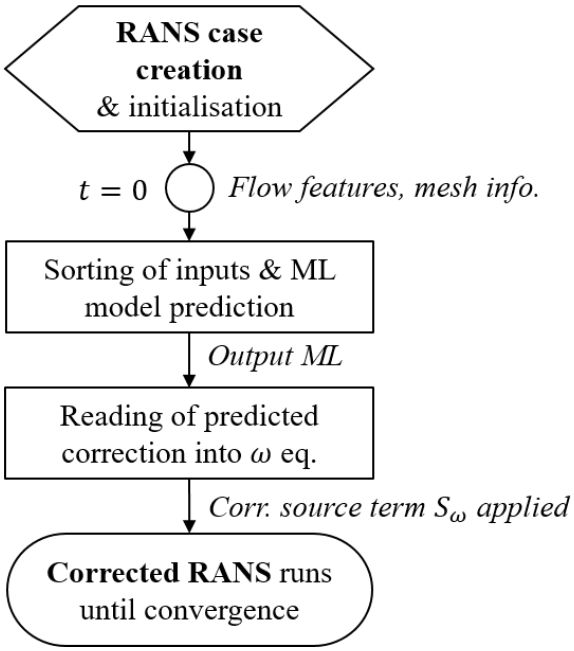


Figure 6.10 Flowchart of the process of a RANS simulation carried out with the M2-informed $k-\omega$ turbulence model and the frozen correction field method

Model M2 was trained to make predictions of the correction source term S_ω right after the initialisation of the RANS case, based on the initial flow conditions. In the process of informing the $k-\omega$ model with the ML model M2, the treatment of the initial flow features was automatised in order to obtain data files that were readable by the ML model. A script was used to automatically add the predicted correction source to the ω transport equation and then resume the now corrected RANS simulation. The flowchart figure 6.10 illustrates that process. Additionally, the predicted frozen correction field was set to only apply in the gas. In fact, the application of large values of turbulence dissipation below the interface level in wavy flows in the present context of thinner films appeared to increase the risk of numerical instabilities during the simulation. Moreover, the $k-\omega$ model is only relevant for turbulent flow modelling, whereas the film is laminar in the three liquid flow regimes.

6.2.2 Implementation of the machine learning model M2

Model M2 structure and training

The training Python notebook of model M2 is available in appendix A.3. M2 used a similar structure to the one used for model M1 in chapter 5. Once again, a simple feed-forward neural network (FFNN) multilayer perceptron (MLP) was used to implement the ML model using the PyTorch library. Four similar flow features were used for the input layer of the neural network: the Reynolds number noted η_1 based on the hydraulic diameter of each phase (c.f. table 6.2), the liquid volume fraction η_2, the distance from the mean interface level η_3, and the distance from the wall η_4. The one output of the model, noted β, was still the correction budget of ω. The four input features are all known after the initialisation of the RANS simulation, which makes model M2 rather easy to operate. The inputs and output of the model M2 are:

- $\eta_1 = \alpha Re_l + (1-\alpha) Re_g$
- $\eta_2 = \alpha$
- $\eta_3 = d_{\text{interf}} = |y - \bar{h}|$
- $\eta_4 = d_{\text{wall}} = \alpha y + (1-\alpha)|H - y|$
- $\beta = S_\omega$

Where α indicates the volume fraction of the primary phase: it is 1 in the liquid and 0 in the gas. Figure 6.11 shows the histogram of the four inputs and one output (log scale) of model M2. For each feature, 200 profile values were collected from each of the 12 qDNS training cases, corresponding to time and space-averaged values of the feature measured in qDNS in the discretised volumes vertically in the channel.

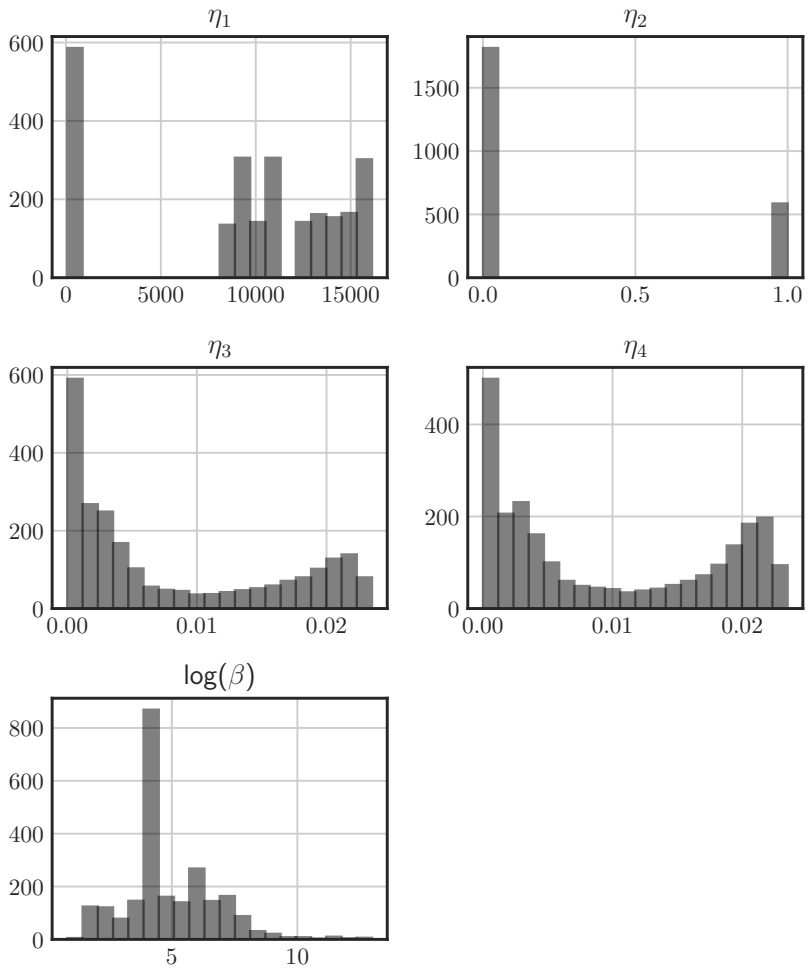


Figure 6.11 Histogram of the inputs and output used in the training of the ML model M2

One can observe from the distribution of the phase volume distribution η_2 that the training data measured in the liquid accounted for one fourth of the total training data, which matched the average film thickness to channel height. A slightly unbalanced distribution of the training data is also noticed in for η_1, suggesting that some flow conditions were covered twice by the training. Increasing the number of cases with different flow conditions would smooth the distribution. Additionally, as previously seen in chapter 5, the correction source term ranges from very small to very large values. In the present context, it ranged from 7 to 10^{13}, and once again the logarithm with base 10 was applied to

β in order to homogenise the data a first time. It is also seen that all data did not belong to the same range from an input feature to an other. For instance, the wall and interface distances ranged from 0 to 0.024, while the Reynolds number ranged from 76 to $1.6 \cdot 10^4$. This is highlighted in the statistics of the inputs and output given in table 6.3. The need of a scaling was indeed well illustrated and a standard scaling was applied to the data prior the training as previously described in eq. 2.70. This operation allowed to obtain all data in the same range with a common mean of approximately 0 and a common standard deviation of approximately 1. The statistics after the standardisation of the data are available in table 6.4. In total, the training dataset contained $5 \times 200 \times 12$ values i.e. 5×2400 values.

Table 6.3 M2 training data statistics before standardisation

Feature	**Count**	$\boldsymbol{\mu}$	$\boldsymbol{\sigma}$	**Min**	**Max**
η_1	2400	9299.5	5573.3	76.56000	16125
η_2	2400	0.2442	0.4297	0.000000	1.000
η_3	2400	0.0082	0.0080	0.000009	0.023
η_4	2400	0.0096	0.0084	0.000010	0.024
$\log(\beta)$	2400	4.8718	1.7995	0.838317	12.97

Table 6.4 Standardised data statistics used for the training of model M2

Feature	**Count**	$\boldsymbol{\mu}$	$\boldsymbol{\sigma}$	**Min**	**Max**
η_1	2400	$-2.5 \cdot 10^{-7}$	1.000210	-1.63	1.19
η_2	2400	$-5.6 \cdot 10^{-8}$	1.000200	-0.57	1.76
η_3	2400	$-1.0 \cdot 10^{-8}$	1.000208	-1.02	1.92
η_4	2400	$-5.2 \cdot 10^{-8}$	1.000209	-1.14	1.66
$\log(\beta)$	2400	$-1.4 \cdot 10^{-8}$	1.000208	-2.24	4.50

The neural network used in model M2 consisted of a first input layer of 4 input neurons, two hidden layers of 256 neurons and one output layer of 1 output neuron, similarly to the neural network used in model M1. ReLU activation functions were used. For the training of model M2, the gradient descent method was employed with the Adam optimiser. The mean squared error (MSE) function was used to compute the model loss. 512 epochs with a batch size of 64 were enough to reach satisfying accuracy. A learning rate of 10^{-4} was selected for the training. The training dataset split was the following: 78% for the training, 17% for the validation, and the remaining 5% for the testing phase. In the context of the model training, the testing data corresponds to training data that was taken out of the training process in order to directly test the model efficiency. This gives a first glance at how effective is the trained model, before using it for predictions in the 3 test cases 1.e, 2.c, and 3.d. In order to increase the efficiency of the training, the data was shuffled randomly at each epoch over the batch size.

The R^2 score was used to compute the accuracy of the training. The latter provided the satisfying training accuracy of 90.9% and validation accuracy of 90.4%. The training progress is shown in figure 6.12, including the loss and accuracy evolution against the number of epochs during the training. The best model was obtained at the 508^{th} epoch. According to the graphs' trends, a longer training would have provided a slightly better training accuracy, however, the model was intentionally not over-trained to avoid any over-fitting of the training data.

After the training, the best version of model M2 was saved. During the testing phase, the model M2 was reloaded and employed to perform predictions

on the same training, validation and test data. The R^2 scores obtained were 92% with the training data, 89% with the validation data, and 87% on the test data.

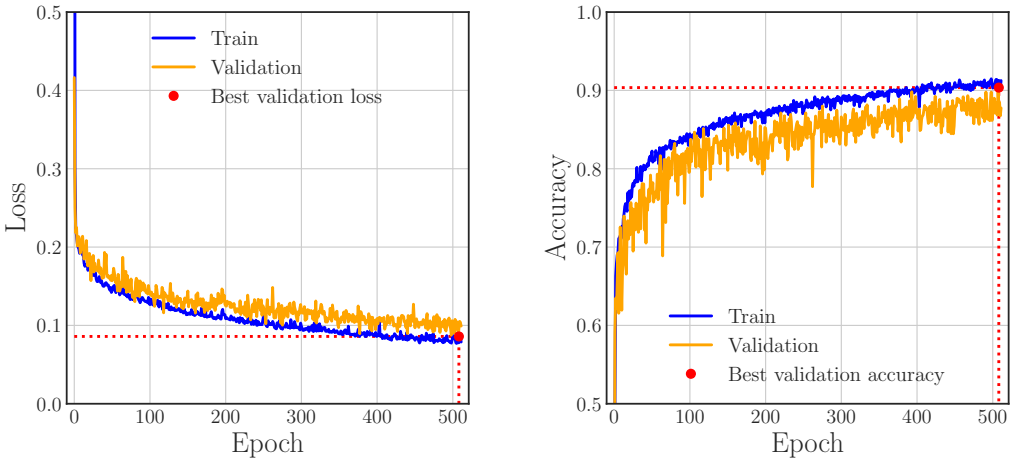


Figure 6.12 Training and validation loss (left) and accuracy (right) against the number of epoches obtained during the training of the ML model M2

6.2.3 Model M2 prediction results and discussions

The model was reloaded again to perform on the three test cases 1.e, 2.c, and 3.d. The results of the correction predictions are shown in figure 6.13 for the three cases, with a representation of the qDNS correction profile ('True') and the profile predicted by model M2. The model M2 predicted the profile 1.e with a R^2 score of 92% and MSE of 0.22, the profile 2.c with a R^2 score of 93% and MSE of 0.24, and the profile 3.d with a R^2 score of 95% and MSE of 0.19. One could then expect to obtain the best results in case 3.d when employing model 2 with the $k-\omega$ model. As the prediction made in case 2.c presented the highest MSE, one could expect to see a less performing model in the flow conditions of this case with the $k-\omega$ model.

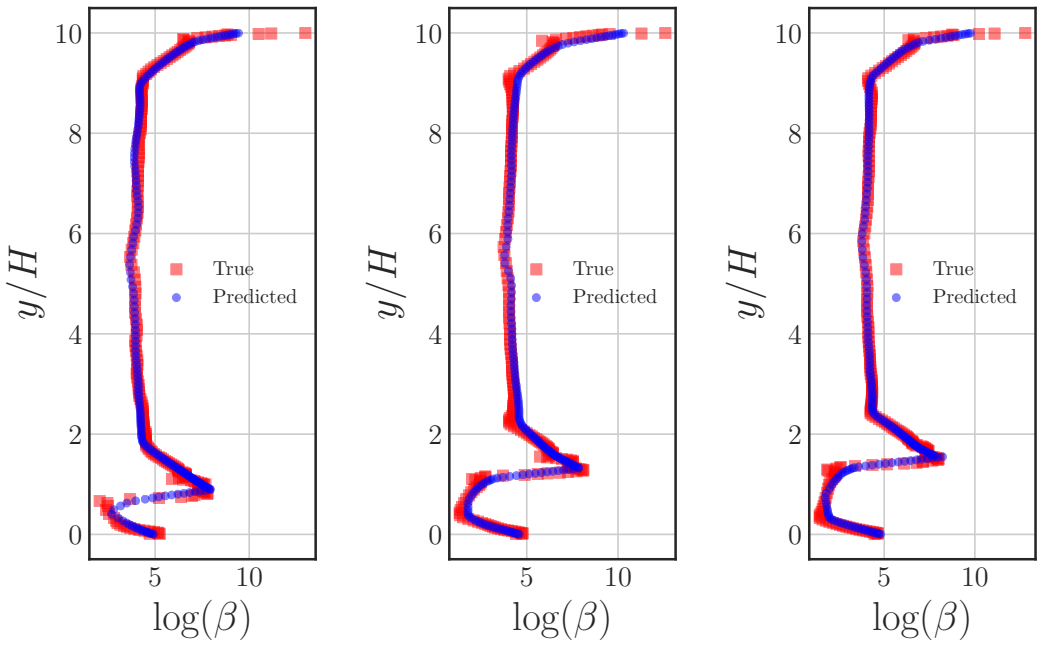


Figure 6.13 Predictions of model M2 in test cases 1.e (left), 2.c (centre), and 3.d (right)

Following the process detailed in the flowchart of figure 6.10, the model M2 was then employed with the $k-\omega$ model in the three test cases 1.e, 2.c, and 3.d. The RANS simulations were performed on the same geometry used for the corresponding qDNS cases but with a coarser mesh resolution. The 3 meshes consisted of 16128 cells. No difference in computational times were observed between the standard RANS simulations and the ML-informed ones. The predictions of the mean axial velocity, the absolute value of Reynolds stress, and turbulent kinetic energy profiles in the gaseous phase are shown in figures 6.14, 6.15, and 6.16. The comparisons between the qDNS, the RANS with the standard $k-\omega$ model, and the RANS with the ML-informed model $k-\omega$ were made.

Overall, the predictions obtained by with the $k-\omega$ model coupled with M2 are in very good agreement with the qDNS results. One observed the best predictions in the case 3.d, for which the mean axial velocity, Reynolds stress,

and TKE predictions seemed the most accurate amongst all test results. In all three cases, the mean axial velocity was slightly underestimated in the wall and interfacial regions, and the same observation was made in the absolute value of Reynolds stress profiles. The absolute value of the Reynolds stress profiles were slightly underestimated across the channel. The TKE profiles were also moderately under-predicted by the corrected model, and some disturbance were noticed just above the interface. These disturbances were due to the way the TKE was calculated and to the waviness of the interface. Indeed, the total TKE was calculated such as: $k_{\text{tot}} = k_{\text{m}} + k_{\text{s}}$ where k_{m} is the modelled TKE and k_{s} is the resolved TKE. While the modelled part of the total TKE is updated after every iteration, the resolved part is obtained by calculating the trace of the tensor $\overline{u_i' u_j'}$ which is averaged over time and does not account for the waviness of the interface.

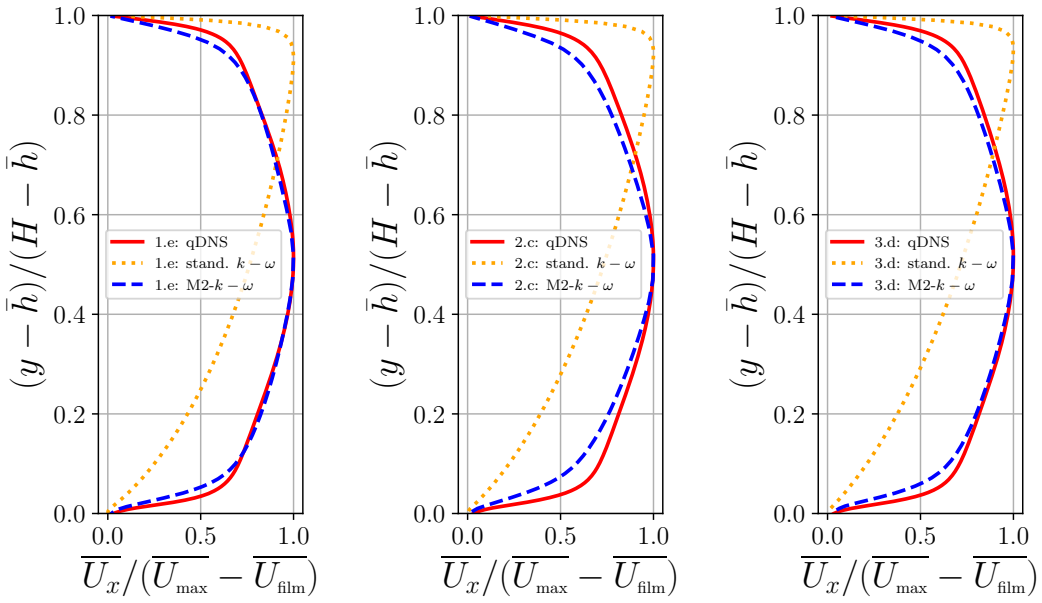


Figure 6.14 Comparison of the mean axial velocity profiles obtained in qDNS, in RANS using the M2-informed $k-\omega$ model, and in RANS using the standard $k-\omega$ model in cases 1.e (left), 2.c (centre), and 3.d (right)

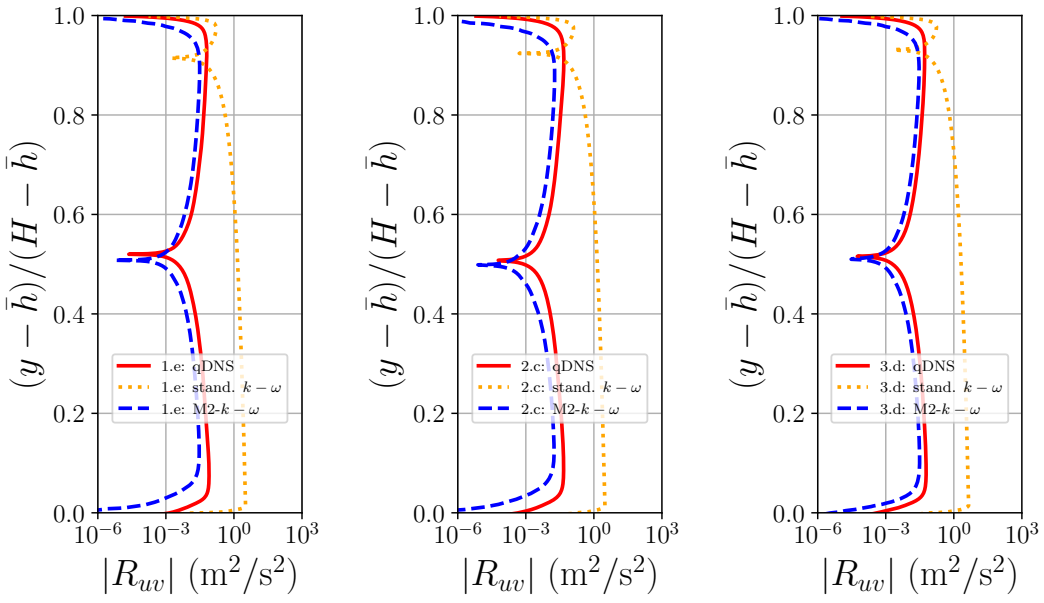


Figure 6.15 Comparison of the absolute values of Reynolds stress profiles obtained in qDNS, in RANS using the M2-informed $k-\omega$ model, and in RANS using the standard $k-\omega$ model in cases 1.e (left), 2.c (centre), and 3.d (right)

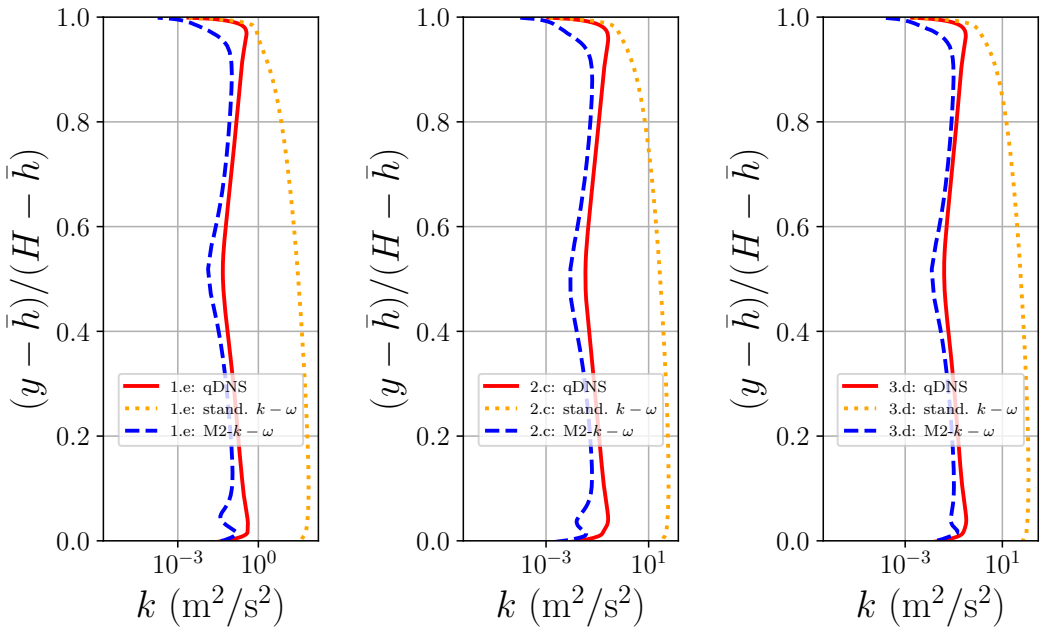


Figure 6.16 Comparison of the TKE profiles obtained in qDNS, in RANS using the M2-informed $k-\omega$ model, and in RANS using the standard $k-\omega$ model in cases 1.e (left), 2.c (centre), and 3.d (right)

The profiles predicted in case 2.c were the less accurate, as it was expected from the tests made prior the simulation. Note that the slightly less accurate results obtained in 2.c might also be due to less accurate predictions of the correction source term by the qDNS obtained for the lowest Reynolds numbers.

The RMSE of the mean axial velocity, Reynolds stress, and TKE profiles in the gaseous phase between the standard $k-\omega$ model and the qDNS predictions, and between the M2-informed $k-\omega$ model and the qDNS predictions were also calculated. The results are shown in table 6.5 and confirmed the previous observations made on the best performance of the model in the test cases.

Table 6.5 RMSEs of the standard ("Stand.") and the ML-informed ("M2") models

Case name	1.e		2.c		3.d	
Model	M2	Stand.	M2	Stand.	M2	Stand.
RMSE $\bar{u}_x$ (m/s)	0.29	1.85	0.35	1.70	0.28	2.00
RMSE R_{uv} (m^2/s^2)	0.0262	1.62	0.0174	1.64	0.0165	2.48
RMSE k (m^2/s^2)	0.11	5.12	0.10	4.89	0.10	7.16

The machine learning model M2 was found to provide appropriate corrections for the three test cases 1.e, 2.c, and 3.d with only quick training on few data using simple inputs flow features that are easily accessible and calculated for the predictions. However, three main weaknesses can be reported through the use of model M2:

- The frozen field method was not adapted for the calculation of the total TKE in the present test cases showing wavy interface patterns

and demonstrated once again the need of an updated correction as the simulation progresses.

- The model can only work in closed channel configurations one of its four inputs includes distance from the top wall.
- The distance from the mean interface level is not an accurate quantity when investigating wavy films.

6.3 Adaptive correction field method

In order to increase the portability of the model M2 and improve the predictions of TKE in the interfacial region, an other machine learning model "M3" was implemented, using an adaptive correction method, and without the use of any geometry-dependant input features.

6.3.1 Process description

As it was discussed in the section 5.4 of chapter 5, providing an updated correction field for the ω transport equation would account for the waviness of the interface. The machine learning model M3 was developed and employed in order to do so, thanks to automatisation scrips (c.f. appendix A.2), which made the ML model produce new predictions at regular intervals in order to adapt to the change in interface levels, and apply the correction in the ω transport equation. Every n iterations, the model M3 makes new predictions of the correction budget S_ω to inform the $k-\omega$ model, as the simulation progresses. Note that more frequent ML predictions are needed for flows with shorter wave periods and larger wave amplitudes. In the present study, no methodology was developed to automatically set up the prediction frequency and it was set

manually depending on the case investigated and the flow pattern obtained in qDNS. Every time a new prediction is made, the model corrects itself as the simulation progresses, and all modelled quantities adapt and converge towards the corrected solution. The process of a RANS simulation using the model M3 is presented in a flowchart in figure 6.17

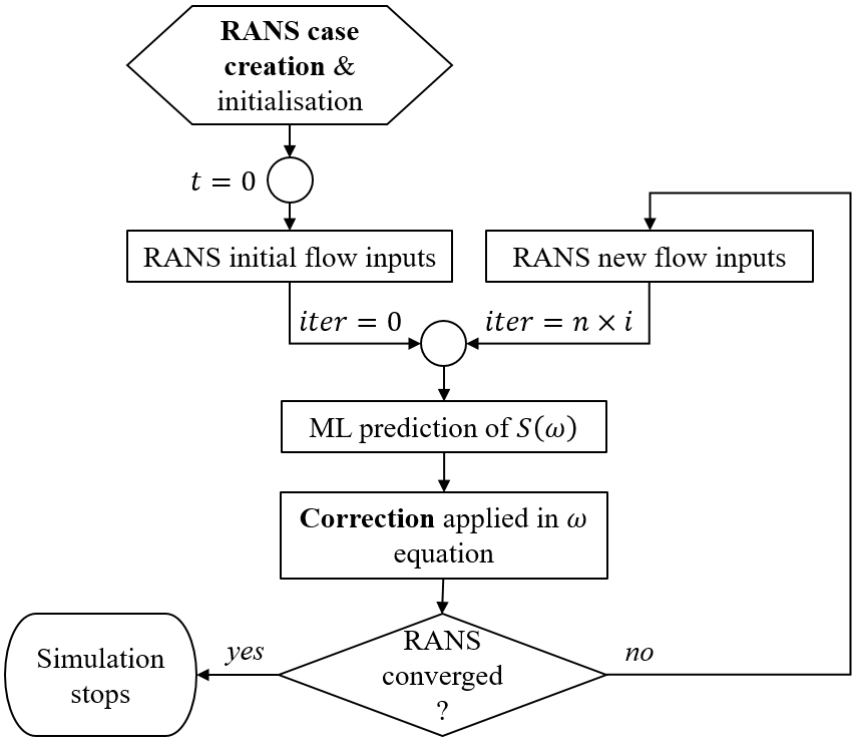


Figure 6.17 Flowchart of the process of a RANS simulation carried out with the M3-informed $k-\omega$ turbulence model and the adaptive correction field method

6.3.2 Implementation of a new machine learning model: model M3

Model M3 structure and training

The training Python notebook of model M3 is available in appendix A.3. M3 was implemented using the PyTorch library with a similar neural network

as model M2, also with four inputs features, and the same correction output. This time the distance from the wall was not used in order to improve the portability of the model, which ideally could also be used in open channel configurations. As the M3 model was used in the context of an adaptive method, the distance from the mean interface levels was also not used, as it is not an accurate measurement in wavy flows and that remains constant in time. The input based on the phase Reynolds number input was replaced by the axial velocity for the same reason, as the phase Reynolds number is calculated by use of the mean thickness of each phase. The input based on the phase volume fraction of the primary phase α remained unchanged. The four inputs and one output of model M3 are:

- $\eta_1 = \alpha u_{x,l} + (1-\alpha) u_{x,g}$
- $\eta_2 = \alpha$
- $\eta_3 = ||\nabla \alpha||$
- $\eta_4 = \alpha k_{\mathrm{tot},l} + (1-\alpha) k_{\mathrm{tot},g}$
- $\beta = S_\omega$

The input η_3 corresponds to the magnitude of the gradient of the primary phase volume fraction. This input was added in order to give the ML model a view on the location of the interface. The distance from the real time interface level would be the base of an ideal input, although, the implementations that were made to solve it appeared to impact the simulation times unreasonably. The input η_4 is the phase turbulent kinetic energy and was used for its sensitivity towards the presence of interfaces and walls. This quantity also adapts to the changes in interface levels as the simulation progresses.

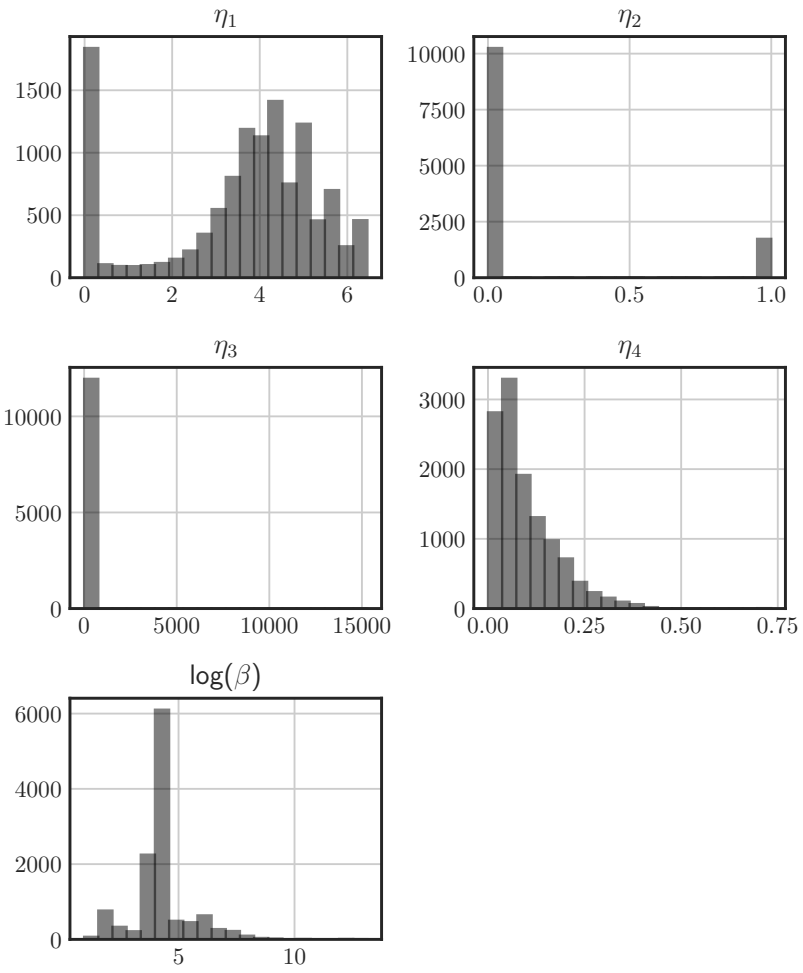


Figure 6.18 Histogram of the inputs and output used in the training of the ML model M3

Figure 6.18 shows the histogram of these four input features and the one output feature (log scale). This time, in order to increase the size of the training dataset, 1001 interpolated point values from the qDNS were employed for each of the five features, and were measured vertically in the channel. Unlike the distribution of the phase Reynolds number inputs of the model M2, the velocity-based new input η_1 allows for a distribution without gaps in the training, fully covering the studied range. As the input η_3 is expected to be zero in the whole domain except around the interface, its distribution is very

unbalanced. η_4 shows a normal-like distribution without gaps. The fifteen qDNS cases are represented in figure 6.19 showing the mean axial velocity profiles and corresponding correction source term profiles calculated in qDNS. The axis of abscissa is the count of interpolated points. The three test cases 1.e, 2.c, and 3.d excluded from the ML training are boxed in red.

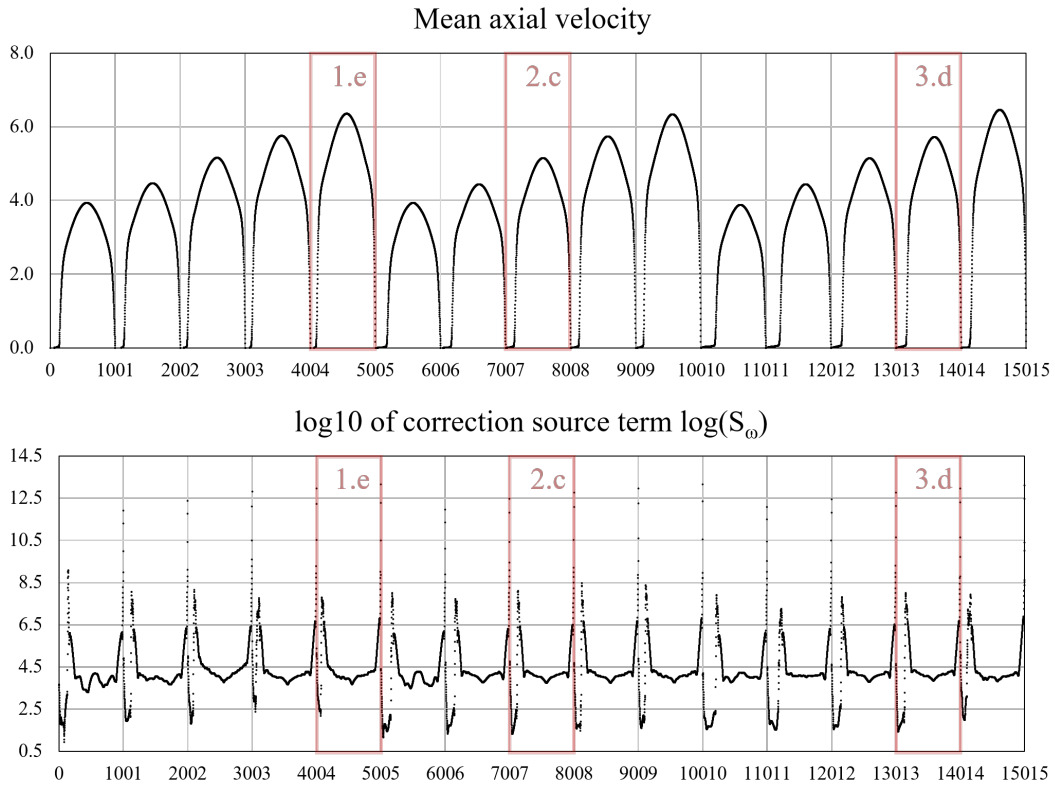


Figure 6.19 Illustration of the qDNS dataset in terms of mean axial velocity (m/s) and log10 correction source term (s^{-2}) including the test cases 1.e, 2.c, and 3.d highlighted in red

The statistics of the training dataset are provided in table 6.6. The training dataset was standardised in order to scale the data in the same range before the training. The statistics after the standardisation for the model M3 are given in table 6.7. In total, the training dataset contained $5 \times 1001 \times 12$ values i.e. 5×12012 values.

Table 6.6 M3 training data statistics before standardisation

Feature	**Count**	$\boldsymbol{\mu}$	$\boldsymbol{\sigma}$	**Min**	**Max**
η_1	12012	3.5420	1.8566	-0.0107	6.459
η_2	12012	0.1455	0.3520	0.0000	1.000
η_3	12012	38.611	613.01	0.0000	15296
η_4	12012	0.0097	0.0068	0.0000	0.024
$\log(\beta)$	12012	4.2372	1.7995	0.9293	13.14

Table 6.7 Standardised data statistics used for the training of model M3

Feature	**Count**	$\boldsymbol{\mu}$	$\boldsymbol{\sigma}$	**Min**	**Max**
η_1	12012	$-1.1{\cdot}10^{-7}$	1.000040	-1.91	1.57
η_2	12012	$-2.0{\cdot}10^{-6}$	1.000096	-0.41	2.43
η_3	12012	$-2.1{\cdot}10^{-7}$	1.000018	-0.06	24.9
η_4	12012	$-5.7{\cdot}10^{-8}$	1.000041	-1.13	7.62
$\log(\beta)$	12012	$-6.9{\cdot}10^{-8}$	1.000040	-2.65	7.14

The neural network of M3 consisted of a first input layer of 4 input neurons, two hidden layers of 512 neurons and one output layer of 1 output neuron. The number of neurons per hidden layer was doubled in comparison to M2's neural network in order to reach a higher training accuracy faster. Similarly to model M2, ReLU activation functions were used, the gradient descent method was employed with the Adam optimiser and a learning rate of 10^{-4}, and the MSE function was used for the loss. This time, 1024 epochs with a batch size of 64 were carried out in order to reach satisfying accuracy. The training dataset was split such as 78% of the data were used for the training, 17% for the validation, and the remaining 5% for the testing phase, similarly to the split carried out in the training of M2. Moreover, the data was randomly shuffled at each epoch

over the batch size.

The training accuracy obtained was 83.6% and the validation accuracy of 87.6%. The evolution of the training is shown in figure 6.20 using the loss and accuracy against the number of epochs carried out. The best model was obtained epoch number 982. During the testing phase of model M3, the R^2 scores obtained were 86% with the training data, 87% with the validation data, and 81% with the test data.

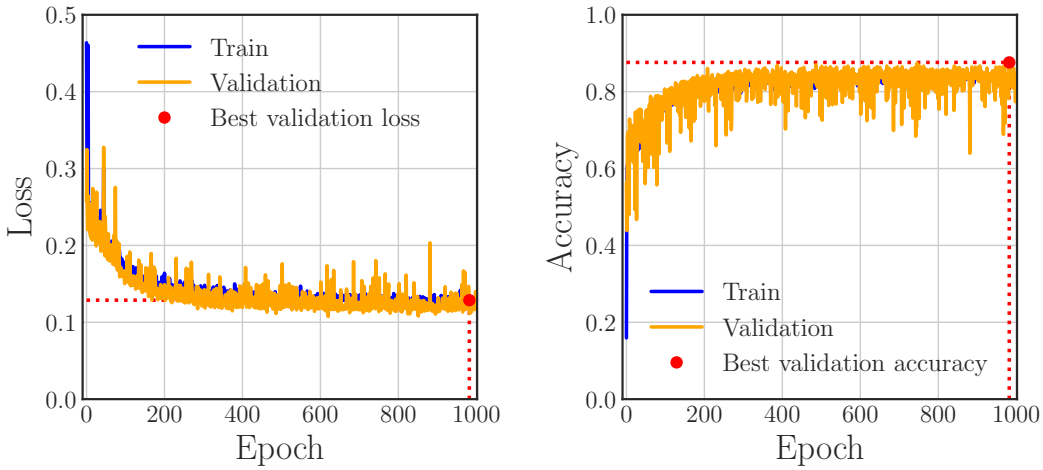


Figure 6.20 Training and validation loss (left) and accuracy (right) against the number of epochs obtained during the training of the ML model M3

6.3.3 Model M3 prediction results and discussions

The model was reloaded to perform on the three test cases 1.e, 2.c, and 3.d. The results of the predictions are shown in figure 6.21 for the three test cases. The profile 1.e was predicted with a R^2 score of 70% and MSE of 0.36, the profile 2.c with a R^2 score of 87% and MSE of 0.21, and the profile 3.d with a R^2 score of 87% and MSE of 0.22.

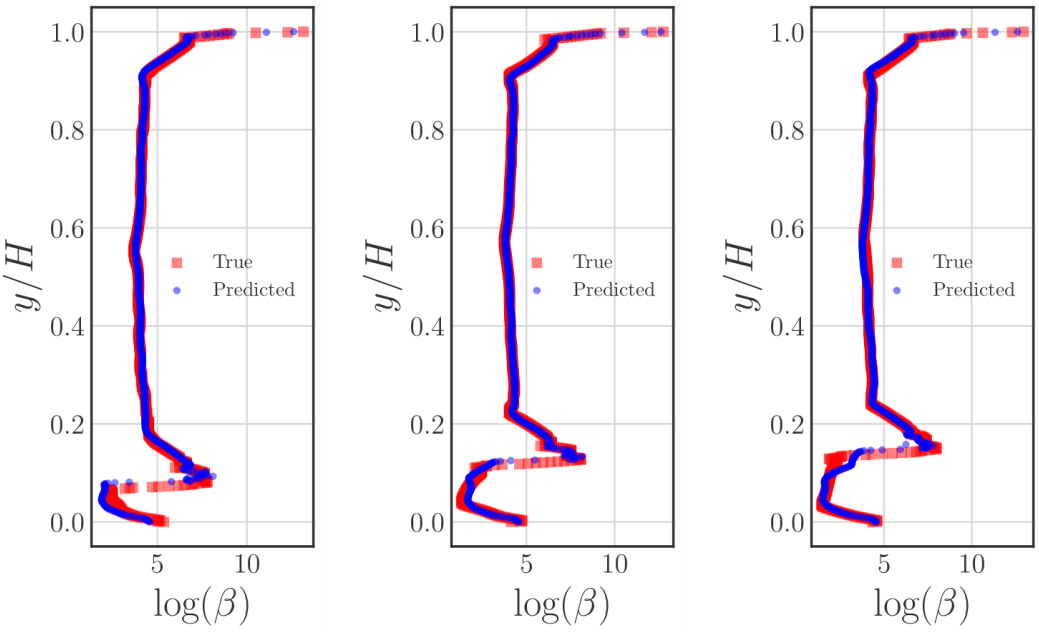


Figure 6.21 Predictions of model M3 in test cases 1.e (left), 2.c (centre), and 3.d (right)

The newly implemented ML model M3 was coupled with the RANS simulations with a Python shell script. The shell script is employed to:

- Call OpenFOAM functions to initialise and run the simulation;
- Execute a secondary script to pre-process the data for the ML model before every new ML prediction;
- Convert the prediction notebook that contains the saved ML model M3 into a Python executable;
- Execute the Python executable to make the new predicted correction field, rescale it, and write it as a readable source in the OpenFOAM simulation directory;
- Resume the simulation after every prediction made by M3.

The process of ML pre-processing, ML prediction, and ML post-processing lasted ≈ 6 seconds during a RANS simulation employing a coarse mesh of 8064 cells, ≈ 8 seconds when using a 16128 cell mesh, and ≈ 12 seconds using a finer mesh of 32256 cells. Therefore, the trend showed that approximately two additional seconds were needed to the process when increasing the mesh by 8064. Within this process, the ML prediction is almost instantaneous, and the most time consuming step is the reading and writing of the data before and after the prediction. Table 6.8 shows the simulation times t_{sim} that the M3-informed $k-\omega$ model took to simulate 1 physical second (p.s.) of the flow using one core for different prediction frequencies f_{pred} of M3, compared to the simulation time of the standard $k-\omega$ model (Std.). Those times were recorded on tests carried out in case 2.c with the 16128 cell mesh.

Table 6.8 Simulation times of the M3-informed model

Simulation/Model	f_{pred} (1/s)	t_{sim} (s)	Comparison to Std.	Comparison to qDNS
qDNS.	-	51463	×29	-
Std.	-	1796	-	÷29
M3	1/0.050	1935	×1.1	÷26
M3	1/0.020	2128	×1.2	÷24
M3	1/0.010	2564	×1.5	÷19
M3	1/0.005	3736	×2	÷14

As expected, increasing the prediction frequency also increased the simulation times. Every time the frequency was doubled, the difference between the standard model simulation time and the M3-informed model simulation time was doubled as well.

Figure 6.22 shows the evolution of the axial velocity and predicted correction source profiles across the channel vertically as the simulation progresses in case 1.e. The first predicted correction source was applied after the initialisation time at $t = 0.02$ s. The plots located to the right of the figure correspond to the converged solution ($t = t_{\text{end}}$).

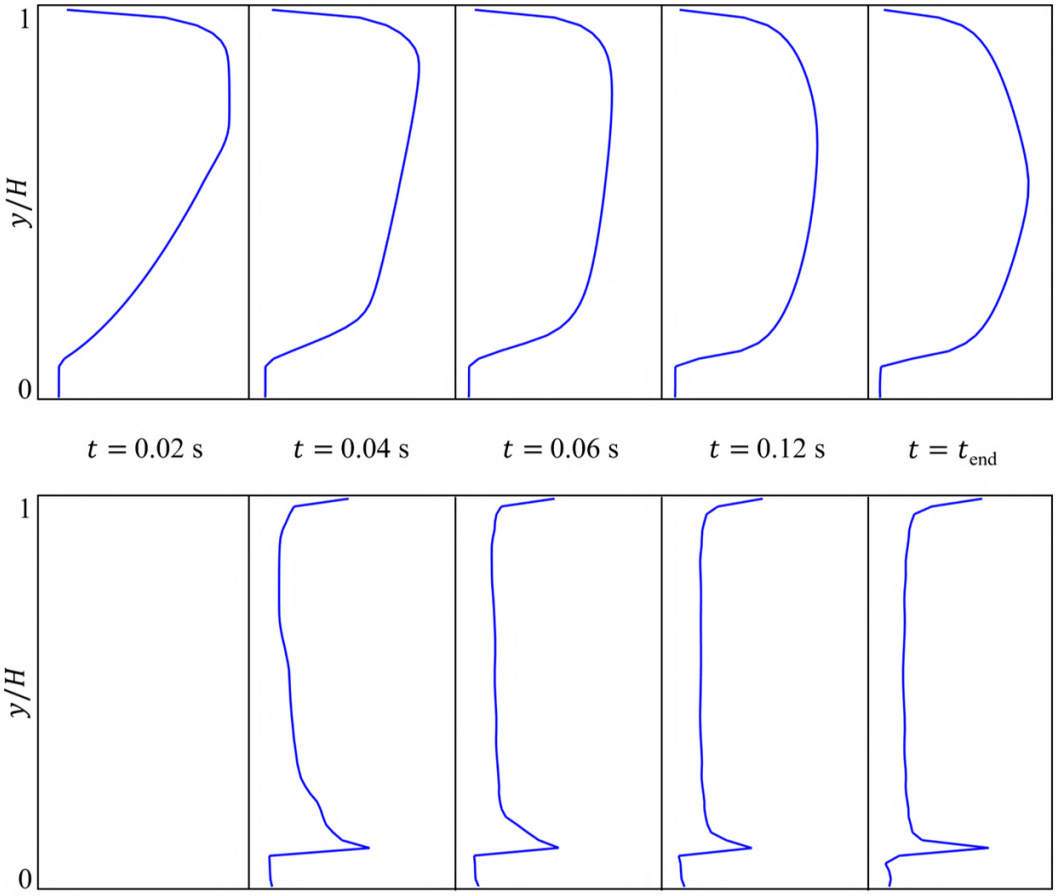


Figure 6.22 Evolution of the corrected axial velocity (first row) and predicted (second row) source using the M3-informed $k - \omega$ model

The coupling of M3 with the $k - \omega$ model was tested in the three test cases 1.e, 2.c, and 3.d using the 16128 cell mesh. A prediction frequency of 1/0.02 s^{-1} was employed for cases 1.e and 2.c and of 1/0.01 s^{-1} for case 3.d. The predictions of the mean axial velocity, the absolute value of Reynolds stress, and

turbulent kinetic energy profiles in the gaseous phase are shown in figures 6.23, 6.24, and 6.25. The predictions obtained with the coupling between the $k-\omega$ model and the ML model M3 appeared to surpass model M2 performances. The M3-informed model's predictions were particularly improved compared to the M2-informed model in the wall and interface regions in which the absolute value of the Reynolds stress and TKE profiles were closer to the qDNS results. The disturbance that was visible in the TKE profiles obtained with the M2 model were no longer visible with M3. The predicted axial velocity agreed particularly well with the qDNS in cases 1.e and 3.d. The Reynolds stress and TKE predictions were also in very good agreement with the qDNS results. The TKE profile was slightly underestimated across the channel height in case 2.c, although, the waviness of the interface appeared to have been accounted thanks to the adaptive correction process, and for the three test cases.

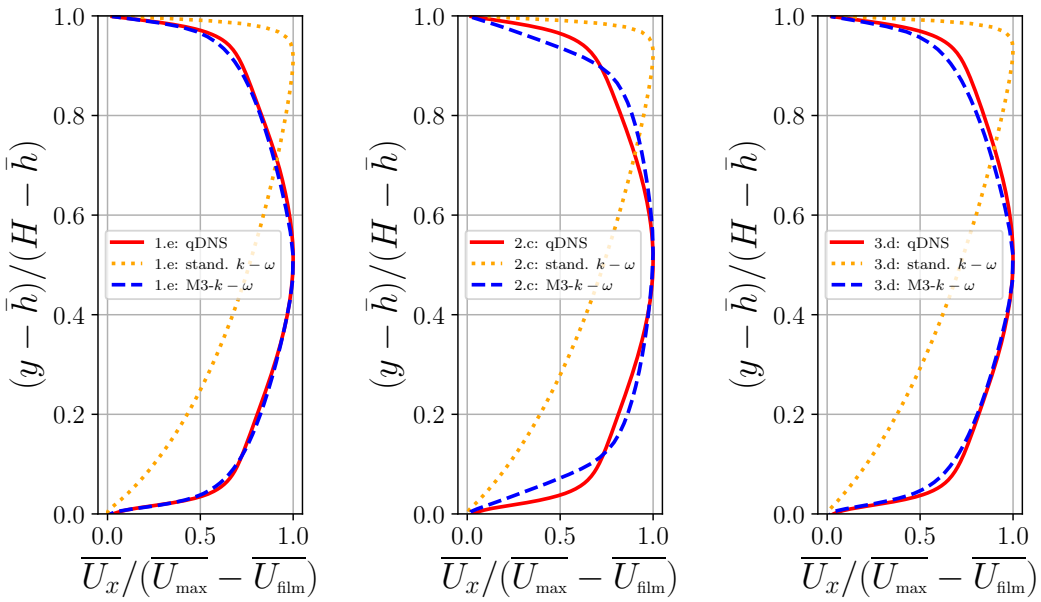


Figure 6.23 Comparison of the mean axial velocity profiles obtained in qDNS, in RANS using the M3-informed $k-\omega$ model, and in RANS using the standard $k-\omega$ model in cases 1.e (left), 2.c (centre), and 3.d (right)

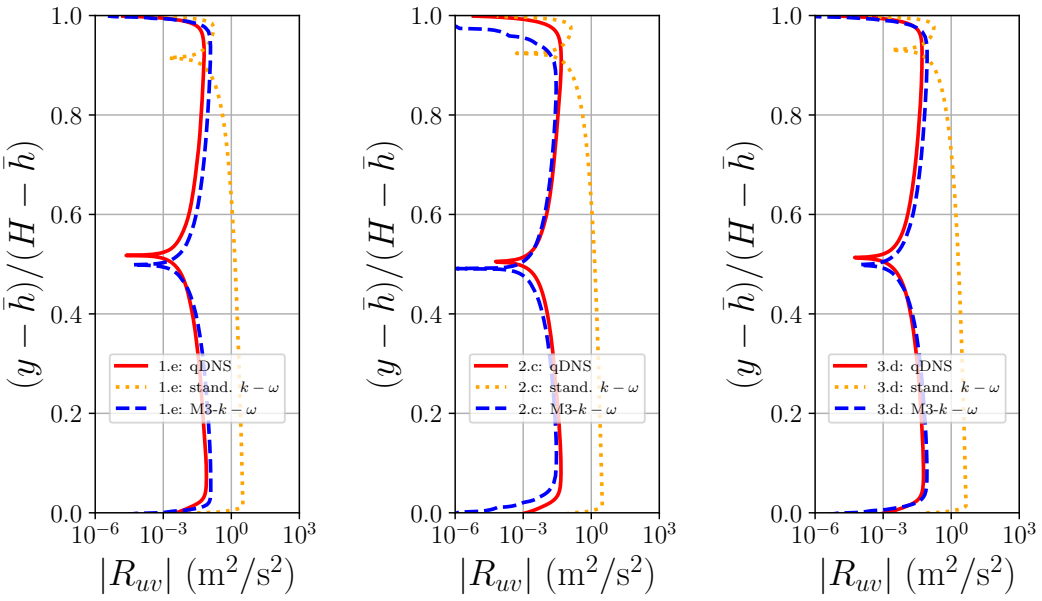


Figure 6.24 Comparison of the absolute value of Reynolds stress profiles obtained in qDNS, in RANS using the M3-informed $k-\omega$ model, and in RANS using the standard $k-\omega$ model in cases 1.e (left), 2.c (centre), and 3.d (right)

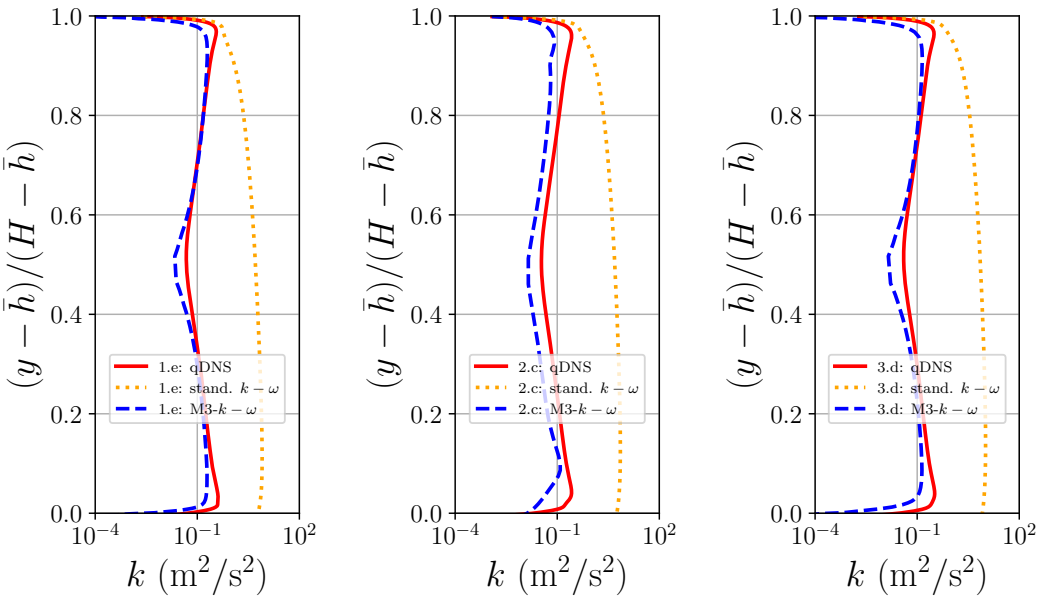


Figure 6.25 Comparison of the TKE profiles obtained in qDNS, in RANS using the M3-informed $k-\omega$ model, and in RANS using the standard $k-\omega$ model in cases 1.e (left), 2.c (centre), and 3.d (right)

The RMSEs of the mean axial velocity, Reynolds stress, and TKE profiles between the qDNS and the two RANS standard and M3-informed turbulence models are presented in table 6.9 for the three test cases. The evolution of the RMSE calculated for each of the previously mentioned quantities is also provided in comparison with the previous RMSEs obtained with the M2-informed model. A down arrow identifies as a smaller error obtained with M3, and an up arrow as an increase of the error.

Table 6.9 RMSEs of the standard ("Stand.") and M3-informed ("M3") models and evolution ("Evol.") of RMSEs in comparison with results of M3

Case name	1.e			2.c			3.d		
Model	Ev.	M3	Stand.	Ev.	M3	Stand.	Ev.	M3	Stand.
RMSE $\bar{u}_x$ (m/s)	$\searrow$	0.10	1.85	$\nearrow$	0.36	1.70	$\searrow$	0.16	2.00
RMSE R_{uv} (m^2/s^2)	$\searrow$	0.007	1.62	$\searrow$	0.016	1.64	$\searrow$	0.009	2.48
RMSE k (m^2/s^2)	$\searrow$	0.07	5.12	$=$	0.10	4.89	$\searrow$	0.07	7.16

Overall, the new M3 model used with the adaptive method enabled better results than with M2 and the frozen field method in the test cases. The case 2.c showed similar results in terms of RMSEs, despite improved predictions near the interface and upper wall. The corrected model performed best in case 1.e, even though this case was predicted with the lowest test accuracy during the test phase of the training. This demonstrates that despite the model ability to predict corrections as it was trained to, the quality of the training is important. Figure 6.19 showed indeed that some S_ω correction profiles obtained in qDNS and used in the training (profiles of cases 1.a, 2.a, and 3.a specifically) showed disturbance and may need to be excluded from the training.

To conclude, the $k-\omega$ turbulence model employed with the adaptive method and coupled with the M3 machine learning model achieved very good predictions of the quantities of interest in comparison with the standard $k-\omega$ model. In order to better appreciate the great improvement provided by the corrected RANS model, figure 6.26 shows the Reynolds stress and TKE profiles obtained in RANS with the standard and corrected models, and in qDNS, using linear scales for the $x-$axis, in case 1.e.

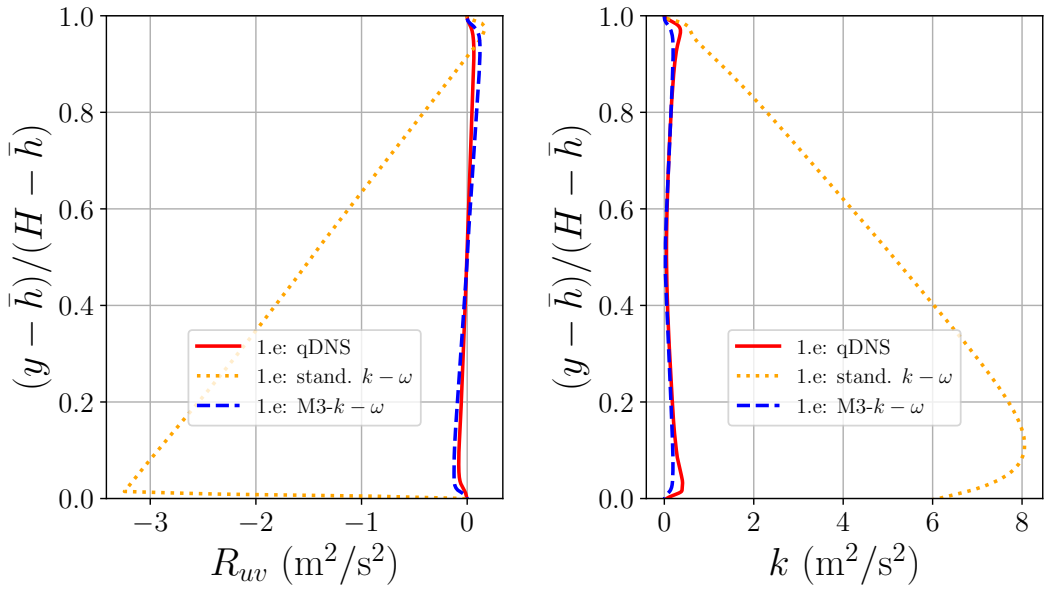


Figure 6.26 Comparison of the Reynolds stress (left) and TKE (right) profiles obtained in qDNS, in RANS with the M3-informed $k-\omega$ model and the standard $k-\omega$ model, for case 1.e using linear scales

6.3.4 Test of model M3 in an open channel

The M3-informed $k-\omega$ model was then employed with the adaptive method in an "open channel" configuration, by setting the top wall BC as a slip-wall BC. The mesh employed in this configuration was similar to the ones used in the previous cases except this time the slip-wall region did not require any grid refinement. The RANS simulation of a thin-film flow using the flow regimes

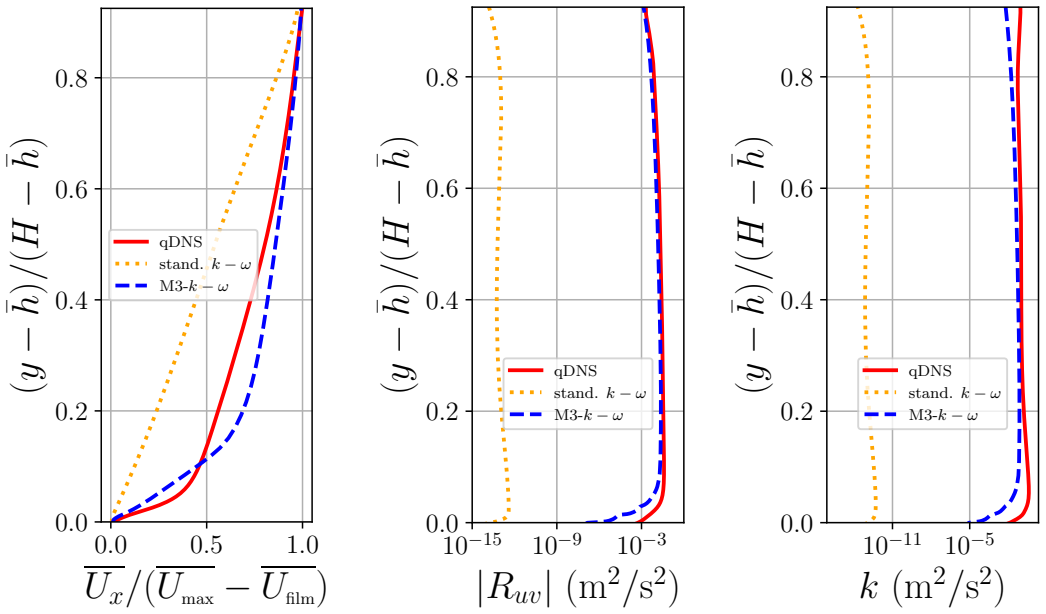


Figure 6.27 Comparison of the mean axial velocity (left), absolute value of Reynolds stress (centre), and TKE (right) profiles obtained in qDNS and in RANS using the M3-informed $k-\omega$ model and the standard $k-\omega$ model cases 3.d with a slip BC at the top wall

described in case 3.d was carried out with the M3-informed $k-\omega$ model and the adaptive correction method. The predictions obtained for the axial velocity, the absolute value of the Reynolds stress, and the TKE profiles are shown in figure 6.27. The M3-informed model under-predicted the three quantities in the interfacial region. The Reynolds stress and TKE were in very good agreement with the qDNS in the rest of the channel. The axial velocity profile was slightly shifted upwards, resulting in an over-prediction of the values in the centre of the gaseous phase.

Despite the significant margin of improvement of the corrected model predictions, the standard $k-\omega$ model was outperformed by the M3-informed model employed with the adaptive correction method. This was highlighted by the calculated RMSEs between the qDNS predictions and the corrected and

standard $k-\omega$ models, presented in table 6.10.

Table 6.10 RMSEs of the M3-informed model ("M3") compared to the standard model ("Stand.") in the open channel configuration

Model	M3	Stand.
RMSE $\bar{u}_x$ (m/s)	0.33	1.24
RMSE R_{uv} (m^2/s^2)	0.01	0.02
RMSE k (m^2/s^2)	0.07	0.11

6.4 Conclusions and discussions

In this chapter, two machine learning models to inform the $k-\omega$ turbulence model were implemented in order to carry out RANS simulations of thin-film flows. The high-fidelity qDNS simulations which were carried out to obtain the training dataset of the two machine learning models that were introduced. The qDNS results were compared against the existing experiments of Hann and Kim [54][76] on thin-film flows in a horizontal, rectangular, and closed channel. The high-fidelity results were in very good agreement with the experiments. This demonstrated that the computational domain simplifications with periodic boundary conditions enabled the production of high-quality qDNS data at least computational cost.

The new machine learning models were employed to inform RANS $k-\omega$ simulations in three test cases of thin-film flows in a horizontal and closed channel. The qDNS solutions of these test cases were excluded from the training of the two machine learning models, and they featured flow conditions that

were covered by the range of the training.

The frozen field correction method introduced in chapter 5 was then reemployed with a new machine learning model "M2" and performed for the RANS $k-\omega$ simulations of the three test cases. The model M2 used four flow input features to predict the ω budget correction S_ω, which were based on flow characteristics that are known at the simulation initialisation or based on the geometry of the channel. Good agreement between the M2-informed $k-\omega$ model and the qDNS predictions were obtained. However, as this method does not account for the waviness of the interface, some disturbances were observed in the region above the interface in the TKE profiles. Moreover, the M2-informed model under-predicted the axial velocity, absolute value of the Reynolds stress, and the TKE profiles in both the top wall and interfacial regions. The predictions obtained by the corrected model were still showing great improvements in comparison with the standard $k-\omega$ turbulence model.

In order to account for the disturbances observed in the interfacial regions and the slight under-predictions of the profiles by the corrected model in the interface and top wall regions, an adaptive correction field method was implemented as well as a new machine learning model "M3". The model was also implemented in a way that no input features depended on geometry characteristics such as the distance from the wall in the model M2. It was also ensured that no input depended on inaccurate quantities such as the distance from the mean interface level in wavy film conditions, which was used in the model M2. The adaptive correction field method was employed for RANS simulations of the three test cases using the $k-\omega$ turbulence model coupled with the machine learning model M3. The new method enabled the elimination of the disturbances originally observed in the interfacial region and the predictions of the

axial velocity, Reynolds stress, and TKE profile across the channel were overall in better agreement with the qDNS than the predictions made using the frozen field correction method. It was observed that the use of the adaptive method increased the RANS simulation times in different ways depending on the frequency of the model prediction needed by the simulation. The test cases showed reasonable increases in the simulation times when using the M3-informed model, in comparison with a standard RANS: from approximately 18 to 43% of increase.

Despite the high accuracy obtained during the testing phase of the machine learning models, some differences with the qDNS were still observed for one of the three test cases, in which the Reynolds employed was the lowest. The RMSEs of the velocity, Reynolds stress and TKE profiles remained significantly larger in this case than the ones calculated in the two others cases. This observation might indicate that some of the training data was not adapted, particularly in low Reynolds cases.

The adaptive method with the model M3 were also tested for the RANS simulation in one of test cases in an "open channel" configuration. The predictions made by the corrected model showed significant improvements in comparison with the standard model predictions. Although results were not perfect, the test demonstrated the potential of the adaptive method coupled with a machine learning trained with geometry-independent inputs. In order to improve the results in the open channel configuration, some cases with larger channel heights could be included in the training, or even cases using both the slip and non slip wall BC at the top of the channel.

Chapter 7

Conclusions and future work

7.1 Conclusions

7.1.1 Summary of the work

The aim of this research project was to develop new strategies for the prediction of two-phase flows and more particularly for the prediction of co-current stratified gas-liquid shear flows with large scale sharp interfaces encountered in aero-engine bearing chambers. The present work focused on improving an existing low order RANS CFD model, the $k-\omega$ turbulence model, which is widely used in the industry to serve engineering and design purposes. The main objective of the research was to address the inability of this averaged turbulence model to predict appropriate interfacial turbulence levels in two-phase shearing flows, due to the high velocity gradients across the interface. The over-prediction of the interfacial turbulence levels by the standard $k-\omega$ model leads to inaccuracies in the prediction of the film thickness. Predicting the film distribution in aero-engines bearing chambers is required to ensure the good lubrication of the engine and to avoid oil burning and coking, which could have important repercussions on the engine operation. Therefore, it is

essential to address the CFD averaged model's weakness to correctly predict interfacial turbulence levels, wall shear, and hence transfer.

After having carried out a review of the existing numerical methods to account for the standard RANS model's overestimation of the interfacial turbulence production, the Egorov method was found to address this issue, by adding destruction source term in the specific turbulence dissipation rate ω transport equation of the RANS $k-\omega$ turbulence model. However, this method requires to manually set a damping parameter in order to adjust the turbulence dissipation intensity, without any proper guidelines, despite many different recommendations have been produced by researchers in various contexts. Moreover, the Egorov method is based on the assumption that the heavier phase can be seen as a wall by the lighter phase. However, this method was found to be valid in smooth interface flow configurations only, while the film flows observed in bearing chambers show wavy interface patterns.

In order to be able to develop new methods to inform the averaged CFD models' interfacial turbulence, the open source CFD code OpenFOAM was chosen to perform all the simulations needed. The two main multiphase methods available in OpenFOAM, namely the volume of fluid method and the Euler-Euler method, were compared in thick-film flow configurations by means of large eddy simulations. Results were compared to the existing experimental results obtained by Fabre et al. on the stratified air-water flow in a horizontal and rectangular channel. The VOF method appeared to perform better than the Euler-Euler method for this type of flow, providing predictions closer to the experiments and significantly shorter computational times. The VOF method

was then employed for all the simulations carried out in the research.

In order to investigate the two-phase flow more in detail, quasi-DNS simulations of the thick-film were conducted in the horizontal channel. A simplification of the channel using cyclic boundary conditions in the flow stream and spanwise directions was performed, based on an autocorrelation study of the fluctuation velocities in the two cyclic directions. This allowed for the saving of computational resources, reducing computational times significantly without altering the quality of the solution. A vortex identification analysis was also carried out using Q-criterion and vorticity magnitudes and results illustrated well the turbulent cascade of energy dissipation from the large turbulent scales, near the walls and above the interface towards the centre of the gaseous phase, and near the bottom wall towards the centre of the liquid phase. This supported the predicted turbulent kinetic energy profiles.

Next, a methodology to inform the standard RANS $k-\omega$ model's interfacial turbulence levels was developed. It consisted in the adjustment of the budget of the transport of ω in the standard $k-\omega$ model by the addition of a correction source term calculated in high-fidelity simulation. The correction source term was injected in the transport equation as a "frozen correction field". The high-fidelity-informed added source term showed to be sufficient to drive the model predictions towards the reference solutions in the thick-film flow configuration studied beforehand. A total of four cases were tested using this method, at low and high gas speed, in both a closed channel (top wall non-slip boundary condition) and an open channel configuration (top wall slip boundary condition). The low speed gas regime generated a smooth interface pattern and the faster regime generated a wavy film pattern. The results obtained in the

smooth interface regime were very satisfying, supporting the frozen correction field method to be employed with the RANS $k-\omega$ model. In the wavy film regime, the results were overall in very good agreement with the high-fidelity solution, however, more inaccuracies were detected near the interface. In comparison to the standard model predictions, great improvements were shown in both regimes and both channel configurations. The decrease in accuracy of the qDNS informed model in the interfacial region in the wavy film regime was assumed to be due to the absence of adaptation of the implemented method. A simple machine learning model based on a feed forward neural network was trained to predict the frozen correction field using simple flow features inputs that are available from the initialisation of the RANS simulation. The model was trained with the qDNS data obtained in both flow regimes in the closed channel configuration. The machine learning model performed a correction field prediction in the smooth interface flow regime, which was applied into the ω transport equation. The $k-\omega$ model informed by the ML model's correction prediction produced nearly identical results as the qDNS-informed $k-\omega$ model, demonstrating the feasibility of the concept i.e. being able to predict an appropriate correction field for the budget of specific turbulence dissipation rate in the $k-\omega$ turbulence model, within a particular training framework.

The following step consisted in building a reasonably extensive dataset for the training of future machine learning models to be used in thin-film flow conditions, including wavy film patterns. The quasi-DNS of 15 cases based on the experimental work of Hann and Kim on thin-film flows in a horizontal and rectangular closed channel were carried out to provide the training data. The 15 cases covered 5 different gas flow rates and 3 liquid flow rates, constituting

a fairly mixed set of flow conditions. The computational domain simplification methodology carried out in the thick-film configuration was reproduced for the 15 cases in order to perform qDNS as reasonable computational costs. The budget of transport of ω was calculated in each case and provided the output of the implemented machine learning models during the training phase. The qDNS results were validated by the experiments and the 15 cases were separated in a training group of 12 cases and a test group of 3 cases. This enabled the test of the machine learning models on unseen data. Two machine learning models employed with two methods for the correction of the transport of specific turbulence dissipation rate budget were developed:

- The frozen correction field method was first employed with a newly implemented machine learning model called "M2" based on a feed forward neural network using four inputs known at the start of a RANS simulation: one was based on the phase Reynolds number, one on the liquid volume fraction distribution, one on the distance from the mean interface level, and one on the distance from the walls. A high training and validation accuracy was achieved during the training phase of M2 allowing for very good predictions of the frozen correction field to be applied into the ω transport equation. The results obtained when informing the $k-\omega$ model with M2 predictions as a frozen correction field showed great improvements in comparison with the standard $k-\omega$ model. More specifically, quasi-symmetric axial velocity profiles were predicted with the corrected model, matching the qDNS results, unlike the profiles predicted by the standard turbulence model which are shifted towards the upper part of the channel in the gas. Underestimations of the axial velocity, absolute value of Reynolds stress, and turbulent kinetic energy profiles were observed in the M2-informed model in the interfacial region and in the upper wall

region. Besides, some disturbances in the interfacial turbulent kinetic energy profiles predicted by the corrected model were also detected. The cause of the under-predicted values and the disturbances observed in the interfacial region was assumed to be associated with the frozen character of the correction applied to the model which does not account for the waviness of the interface as previously noticed in the proof of concept of the same method, but also the use of time averaging methods and unreliable inputs used by M2 like the distance from the mean interface level, which are not adapted to wavy films.

- An adaptive correction field method was therefore implemented in order to provide updated predictions for the ω transport equation as frequently as required according to the wave frequency and amplitude of the film. The method was developed along with a new ML model "M3", which input features are computable at any time while the simulation progresses for the ML model M3 to be able to perform the prediction update of the correction field. The profiles predicted by the M3-informed $k-\omega$ model used with the adaptive correction field method were in very good agreement with the qDNS results. The former disturbances seen in the turbulent kinetic energy profiles in the interfacial region were no longer observed and less underestimation of the quantities were noticed in the top wall and interface areas. The calculated RMSEs between the qDNS and the M2-informed $k-\omega$ model, and between the qDNS and the M3-informed $k-\omega$ model highlighted this improvement and also emphasised the $k-\omega$ enhancement in comparison with the standard model. The newly developed adaptive method appeared to increase the simulation times accordingly to the chosen frequency of the predictions to be made by the ML model. For the tested cases, a prediction frequency of one

every 2 ms to one every 1 ms was enough to obtain satisfying results, resulting in an increase of the simulation time by 18 to 43% in comparison with the standard $k-\omega$ model. Finally, the method was employed in an open channel configuration in order to test the portability of the ML model M3, as it was implemented free of geometry-dependent inputs. Results showed significant improvements in comparison with the standard model performance, despite flawed predictions of the velocity profile, which appeared slightly shifted upwards in comparison with the qDNS.

The implemented adaptive correction field method employed with a more portable machine learning model like M3 demonstrated the potential of the use of machine learning to inform averaged models without exacerbating the computational times.

7.1.2 Key contributions

The key contributions provided in this thesis can be listed as follows:

- Experimentally validated high-fidelity qDNS simulation datasets were generated for the training of machine learning models to inform lower order RANS models.
- Two correction methods namely the "frozen field" and "adaptive" correction methods were implemented in order to inform the interfacial turbulence in the standard RANS $k-\omega$ model
- The two methods were employed for applications with several trained machine learning models to inform the interfacial turbulence in stratified two-phase channel flows simulations carried out with the RANS $k-\omega$ model for two flow configurations (thick and thin films) in two channel geometries (closed and open)

- Excellent agreements with the experimental and high-fidelity data were obtained using both methods in the two film configurations, however limitations with still good results were noticed in the open channel geometry as it falls outside of the machine learning model's training framework. Note also that the frozen field correction method allows for the best results in the smooth interface regime.

7.2 Recommendations for future improvements

Optimising the coupling OpenFOAM–PyTorch

Despite the higher training and validation accuracy of M2 employed with the frozen correction field method, improved results were obtained using M3 with the adaptive correction method, demonstrating that it is essential to account for the film waviness. Therefore, a method in which a prediction update of the correction field at each iteration would be ideal. However, this would imply an excessive increase in computational costs. The prediction update phases carried out during the adaptive correction method include the pre-processing data reading and writing, the model prediction, and the post-processing data writing. Those pre and post-processing steps account for the greater computational times of the prediction update phases. Therefore, the optimisation of those steps is essential in order to reduce the simulation cost.

The integration of the machine learning model within the OpenFOAM turbulence model's source code was considered by using the C++ API of PyTorch instead of the currently used Python API. The embedded model might perform the whole prediction process faster than when using a python shell script. However, this method was not yet tested because of a lack of time.

Enhancing the machine learning model performances

In order to improve the portability of the machine learning models, dimensionless inputs could be used. This would allow for the use of the model in similar flow regimes as the ones that were included in the training, but in new flow and geometry configurations.

An input based on the distance from the real time interface level would be a real asset to the machine learning model, as it is a very well distributed quantity as seen on the similar distribution of the distance from the mean interface level. Attempts to compute the real time distance from the interface in OpenFOAM were made using the `interfaceHeight` function object with measurement probes generated in every mesh cell in order to capture the interface distances in all mesh coordinates. However, the estimation of the real time interface distances from every mesh cell with this method was an excessively long task and consequently, it was omitted. Instead, the distances from the mean interface level were used.

Improving the training data

It was observed that despite the very high test accuracy produced by the implemented machine learning models, the predicted corrections made in one of the test cases did not produce results quite as good as the ones obtained in the other cases. This could easily be explained by a poorer quality training data provided by part of the qDNS simulations. A different averaging methodology is proposed in order to improve the quality of the training for future research. It consists in carrying out a phase averaging instead of a time averaging, and the process would be the following:

- The wave period and wave length of the wavy interface must acquired first, using for example the Fourier decomposition of the time series of interface level variations to find the physical natural frequency
- Multiple vertical measurement probes are then placed in the flow stream direction over one wave length, in order to write the data of interest every wave period. This allows for the generation of multiple datasets of a "frozen image" of the two-phase flow, corresponding to the same phase
- As the written data are all supposed to capture the same wave phase, the averaging operation can now be applied over the written data on the quantity of interest.

Consequently, it will prevent from any diffusion of the fields when time averaging around the interface, and result in an improvement of the quality of the training data. Note that this methodology would only work for 2D wave flow regimes.

Even though it was observed that only 2D wave patterns could be obtained when using cyclic condition in the spanwise direction, 3D wave flow regimes could be incorporated to the training data by carrying out high-fidelity simulations in rectangular channels with non-slip side walls. Experimental data could theoretically be added to the training dataset but one must beforehand implement an accurate and easy method to calculate the ω transport budget currently used as the output of the developed machine learning models.

Another way to improve the training data is evidently to expend the range of flow conditions to include in the dataset. This means that more flow geometries

(channel heights, film thicknesses) and/or more flow regimes (lower and higher gas and liquid Reynolds) must be investigated in high-fidelity simulations.

Further work on the adaptive correction method

In the adaptive correction method, the frequency of prediction of the correction field is currently set manually in the simulation settings: the Python shell script triggers the OpenFOAM solver and the latter stops after the selected writing interval using the keywork `nextWrite`. A guidelines table could be produced in order to know in advance the optimal update frequency depending on the flow regime, similarly to the pattern map introduced by Andritsos and Hanratty for the identification of the interface pattern according to the film and gas superficial velocities.

Application to other averaged models

It was reviewed in the numerical work of Federix et al. that the weakness of the $k-\omega$ model in over-predicting the interfacial turbulence levels in two-phase shearing flows is also shared by the $k-\varepsilon$ model. One could then apply the methods developed for the $k-\omega$ model to the $k-\varepsilon$ model by introducing a correction source term for the transport of the turbulence dissipation rate ε.

Similar developments could also be provided to the $k-\omega$ SST model in order to benefit from the modelling advantages of both the $k-\omega$ model in the near wall regions and of the $k-\varepsilon$ in the flow stream.

The possibility in providing relevant corrections for transition models such as the $\gamma-Re_\theta$ model was not yet investigated, although could be also of interest for two-phase flow regimes in which both turbulent and laminar flow conditions

are present, while the standard $k-\omega$ and $k-\varepsilon$ models are unable to provide appropriate predictions in laminar flows.

Application to new geometries

The test of model M3 in an open channel configuration investigated in section 6.3.4 demonstrated the potential of the methodology developed and the ability of the ML-enhanced model to adapt to unseen geometries. Therefore, it would be interesting to employ the same methods to model two-phase flows in other types of geometry, such as modelling annular flows or stratified flows in circular pipe channels for instance.

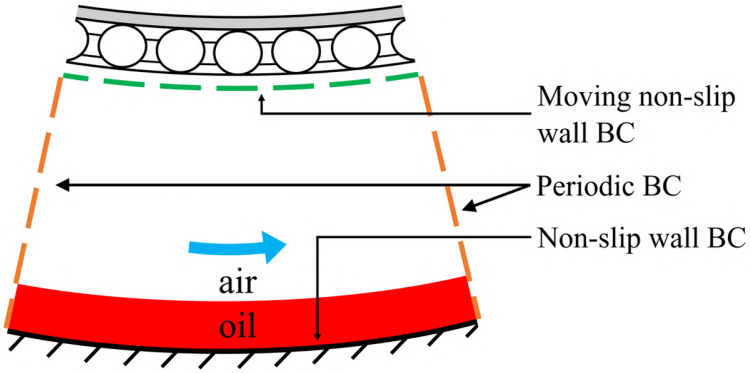


Figure 7.1 Two-phase flow in periodic section of bearing chamber with boundary conditions (BC)

Finally, with the aim to correctly model the shear flows present in a true aero-engine bearing chamber, the developed methods could be tested in intermediate geometries. More elaborate versions of the horizontal channel investigated in this thesis could be tested, such as a periodic section of a rectangular channel with curved bottom and top walls and without gravity force, as illustrated in figure 7.1. Next, the ML-enhanced model could be tested in a simplified

version of a full-scale bearing chamber similar to the geometry employed by Bristot et al. [15] presented in chapter 3 (figure 3.7).

List of publications

Published work:

Bertolotti, L, Jefferson-Loveday, R, Ambrose, S, and Korsukova, E. "A Comparison of VOF and Euler-Euler Approaches in CFD Modelling of Two-Phase Flows With a Sharp Interface." Proceedings of the ASME Turbo Expo 2020: Turbomachinery Technical Conference and Exposition. Volume 2C: Turbomachinery. Virtual, Online. September 21–25, 2020. V02CT35A018. ASME. (2021).
http://doi.org/10.1115/GT2020-14681.

Bertolotti, L., Jefferson-Loveday, R., Ambrose, S., and Korsukova, E. "A Comparison of Volume of Fluid and Euler–Euler Approaches in Computational Fluid Dynamics Modeling of Two-Phase Flows With a Sharp Interface." ASME. J. Turbomach. December 2021; 143(12): 121005. (2021).
http://doi.org/10.1115/1.4051554.

Accpeted paper:

Bertolotti, L., Jefferson-Loveday, R., Ambrose, S., and Korsukova, E. “High Fidelity CFD-Trained Machine Learning to Inform RANS-Modelled Interfacial Turbulence” Proceedings of Global Power and Propulsion Society. (2022).

Future submissions:

Bertolotti, L., Jefferson-Loveday, R., Ambrose, S., and Korsukova, E. "Two-Phase Stratified Flow Modelling Using a Machine Learning-Driven RANS Model" to be submitted to the International Journal of Multiphase Flow.

Korsukova, E., Ambrose, S., Bertolotti, L., and Jefferson-Loveday, R. "Scale Resolving Simulations of Wavy Shear Driven Stratified Gas-Liquid Flow in a Horizontal Channel" to be submitted to the ICMF 2023 (11th International Conference on Multiphase Flow, Kobe, Japan, April 2-7, 2023).

$\star$

References

[1] Al-Wahaibi, T., Yusuf, N., Al-Wahaibi, Y., and Al-Ajmi, A. (2012). Experimental study on the transition between stratified and non-stratified horizontal oil–water flow. *International Journal of Multiphase Flow*, 38(1):126–135. https://doi.org/10.1016/j.ijmultiphaseflow.2011.08.007.

[2] Andreussi, P. and Persen, L. (1987). Stratified gas-liquid flow in downwardly inclined pipes. *International Journal of Multiphase Flow*, 13(4):565–575. https://doi.org/10.1016/0301-9322(87)90022-X.

[3] Andritsos, N. (1986). Effect of pipe diameter and liquid viscosity on horizontal stratified flow.

[4] Andritsos, N. and Hanratty, T. (1987a). Influence of interfacial waves in stratified gas-liquid flows. *AIChE Journal*, 33(3):444–454. https://doi.org/10.1002/aic.690330310.

[5] Andritsos, N. and Hanratty, T. (1987b). Interfacial instabilities for horizontal gas-liquid flows in pipelines. *International Journal of Multiphase Flow*, 13(5):583–603. https://doi.org/10.1016/0301-9322(87)90037-1.

[6] Arteaga-Arteaga, H., Mora-Rubio, A., Florez, F., Murcia-Orjuela, N., Diaz-Ortega, C., Orozco-Arias, S., delaPava, M., Bravo-Ortíz, M., Robinson, M., Guillen-Rondon, P., and Tabares-Soto, R. (2021). *Machine learning applications to predict two-phase flow patterns.* PeerJ Computer Science 7:e798 https://doi.org/10.7717/peerj-cs.798.

[7] Ashton, N. and Revell, A. (2014). Investigation into the predictive capability of advanced reynolds-averaged navier-stokes models for the drivaer automotive model. In Park, H., editor, *The International Vehicle Aerodynamics Conference*, pages 125–137. Woodhead Publishing, Oxford. https://doi.org/10.1533/9780081002452.4.125.

[8] Batchelor, G. (1967). *An introduction to fluid dynamics*. Cambridge, England: Cambridge University Press. Applied Mathematics, University of Cambridge, England.

[9] Benz, M. and Schulenberg, T. (2015). A novel approach for turbulence modelling of wavy stratified two-phase flow. *16th International Topical Meeting in Nuclear Reactor Thermal Hydraulics (NURETH-16)*. Chicago, Ill, August 30-September 4.

[10] Bernard, P. (1986). Limitations of the near-wall k-epsilon turbulence model. *AIAA Journal*, 24(4):619–622. https://doi.org/10.2514/3.9316.

[11] Bertolotti, L., Jefferson-Loveday, R., Ambrose, S., and E., K. (2020). A comparison of VOF and Euler-Euler approaches in CFD modelling of two-phase flows with a sharp interface. volume Volume 2C: Turbomachinery. Virtual. of *Proceedings of the ASME Turbo Expo 2020: Turbomachinery Technical Conference and Exposition*, Online. ASME. http://doi.org/10.1115/GT2020-14681.

[12] Bertolotti, L., Jefferson Loveday, R., Ambrose, S., and Korsukova, E. (2021). A Comparison of Volume of Fluid and Euler–Euler Approaches in Computational Fluid Dynamics Modeling of Two-Phase Flows With a Sharp Interface. *Journal of Turbomachinery*, 143(12). https://doi.org/10.1115/1.4051554.

[13] Biberg, D. (2005). *Mathematical models for two-phase stratified pipe flow*. Ph.D. Thesis, University of Oslo, Norway.

[14] Boussinesq, J. (1877). Essai sur la théorie des eaux courantes. *Mémoires présentées par divers savants à l'Académie des Sciences de l'Institut National de France*, 13:1–680. Imprimerie Nationale, Paris.

[15] Bristot, A., Morvan, H., Simmons, K., and Klingsporn, M. (2017). Effect of turbulence damping in VOF simulation of an aero-engine bearing chamber. volume Volume 2B: Turbomachinery, pages 1–10, Charlotte, North Carolina, USA. ASME. http://doi.org/10.1115/GT2017-63436.

[16] C., G. and L., W. (2021). Deep learning to replace, improve, or aid CFD analysis in built environment applications: A review. *Building and Environment*, 206:108315. https://doi.org/10.1016/j.buildenv.2021.108315.

[17] Cantwell, B. and Coles, D. (1983). An experimental study of entrainment and transport in the turbulent near wake of a circular cylinder. *Journal of Fluid Mechanics*, 136:321–374. http://doi.org/10.1017/S0022112083002189.

[18] Caretto, L. S., Gosman, A. D., Patankar, S. V., and Spalding, D. B. (1973). Two calculation procedures for steady, three-dimensional flows with recirculation. In Cabannes, H. and Temam, R., editors, *Proceedings of the Third International Conference on Numerical Methods in Fluid Mechanics*, pages 60–68, Berlin, Heidelberg. Springer Berlin Heidelberg. http://doi.org/10.1007/BFb0112677.

[19] Chen, B., Chen, G., Sun, H., and Zhang, Y. (2014). Effect of oil droplet deformation on its deposited characteristics in an aeroengine bearing chamber. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 228(2):206–218.

[20] Chourushi, T. (2017). Effect of fluid elasticity on the numerical stability of high-resolution schemes for high shearing contraction flows using openfoam. *Theoretical and Applied Mechanics Letters*, 7(1):41–51. https://doi.org/10.1016/j.taml.2017.01.005.

[21] Cioncolini, A., Del Col, D., and Thome, J. R. (2015). An indirect criterion for the laminar to turbulent flow transition in shear-driven annular liquid films. *International Journal of Multiphase Flow*, 75:26–38.

[22] Cohen, L. and Hanratty, T. (1965). Generation of waves in the concurrent flow of air and a liquid. *AIChE Journal*, 11(1):138–144. https://doi.org/10.1002/aic.690110129.

[23] Cohen, L. and Hanratty, T. (1968). Effect of waves at a gas—liquid interface on a turbulent air flow. *Journal of Fluid Mechanics*, 31(3):467–479. http://doi.org/10.1017/S0022112068000285.

[24] Courant, R., Friedrichs, K., and Lewy, H. (1928). Über die partiellen Differenzengleichungen der mathematischen Physik. *Math. Ann.*, 100:32–74. 10.1007/BF01448839.

[25] Crank, J. and Nicolson, P. (1947). A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Mathematical Proceedings of the Cambridge Philosophical Society*, 43(1):50–67.

[26] Cruz, M. A., Thompson, R. L., Sampaio, L. E., and Bacchi, R. D. (2019). The use of the reynolds force vector in a physics informed machine learning approach for predictive turbulence modeling. *Computers and Fluids*, 192:104258.

[27] Delhaye, J., Collier, J., Herwitt, G., Mayinger, F., and Bergles, A. (1981). *Two-phase flow and heat transfer in the process and power industries*. Hemisphee Publishing Corporation, Edition McGraw-Hill, New York.

[28] Deshpande, S. (2012). Evaluating the performance of the two-phase flow solver interfoam. *Computational Science and Discovery*, 5(1). https://doi.org/10.1088/1749-4699/5/1/014016.

[29] Dong, Z., Bürgler, M., Hohermuth, B., and Vetsch, D. (2022). Density-based turbulence damping at large-scale interface for reynolds-averaged two-fluid models. *Chemical Engineering Science*, 247:116975. https://doi.org/10.1016/j.ces.2021.116975.

[30] Drazin, P. G. and Riley, N. (2006). *The Navier-Stokes Equations: A Classification of Flows and Exact Solutions*. London Mathematical Society Lecture Note Series. Cambridge University Press. https://doi.org/10.1017/CBO9780511526459.

[31] Eastwood, S., Tucker, P., Xia, H., and Klostermeir, C. (2009). Developing large eddy simulation for turbomachinery applications. *Philosophical Transaction of the Royal Society*, 367:2999–3013.

[32] Egorov, Y., Martin, A., Boucker, M., Pigny, S., Scheurer, M., and Willemsen, S. (2004). Validation of CFD codes with pts-relevant test cases. volume 5th Euratom Framework Programme ECORA project, pages 91–116.

[33] Evans, M., Harlow, F., and Bromberg, E. (1957). *The Particle-In-Cell Method for Hydrodynamic Calculations*. Los Alamos National Lab, New Mexico, Oak Ridge, TN, USA. Technical Report.

[34] Fabre, J., Masbernat, L., Fernandez-Flores, R., and Suzanne, C. (1987a). *Stratified flow, Part II: interfacial and wall shear stress*. Multiphase Science and Technology, Volume 3, G. F. Hewitt, J. M. Delhaye and N. Zuber, eds. Hemisphere.

[35] Fabre, J., Masbernat, L., and Suzanne, C. (1987b). Experimental data set No. 7: Stratified flow, Part I: Local structure. *Multiphase Science and Technology*, 3(1-4):285–301.

[36] Faghri, A. and Zhang, Y. (2006). Solid-liquid-vapor phenomena and interfacial heat and mass transfer. In *Transport Phenomena in Multiphase Systems*,

pages 331–420. Academic Press, Boston. https://doi.org/10.1016/B978-0-12-370610-2.50010-6.

[37] Fan, W. and Anglart, H. (2019). Progress in phenomenological modeling of turbulence damping around a two-phase interface. *Fluids*, 4:136. http://doi.org/10.3390/fluids4030136.

[38] Farhadi, A., Mayrhofer, A., Tritthart, M., Glas, M., and Habersack, H. (2018). Accuracy and comparison of standard k-epsilon with two variants of k-omega turbulence models in fluvial applications. *Engineering Applications of Computational Fluid Mechanics*, 12(1):216–235. https://doi.org/10.1080/19942060.2017.1393006.

[39] Ferziger, J. and Peric, M. (2002). *Computational Methods for Fluid Dynamics*. Springer, Berlin, 3 edition. http://dx.doi.org/10.1007/978-3-642-56026-2.

[40] Foundation, T. O. (2018). *OpenFOAM User Guide version 6*.

[41] Frederix, E., Mathur, A., Dovizio, D., Geurts, B., and Komen, E. (2018). Reynolds-averaged modeling of turbulence damping near a large-scale interface in two-phase flow. *Nuclear Engineering and Design*, 333:122–130. https://doi.org/10.1016/j.nucengdes.2018.04.010.

[42] Fröhlich, J., Mellen, C., RODI, W., Temmerman, L., and LESCHZINER, M. (2005). Highly resolved large-eddy simulation of separated flow in a channel with streamwise periodic constriction. *Journal of Fluid Mechanics*, 526:19 – 66.

[43] Fulgosi, M., Lakehal, D., Banerjee, S., and De Angelis, V. (2003). Direct numerical simulation of turbulence in a sheared air–water flow with a deformable interface. *Journal of Fluid Mechanics*, 482:319–345. http://doi.org/10.1017/S0022112003004154.

[44] Gada, V., Tandon, M., Elias, J., Vikulov, R., and Lo, S. (2017). A large scale interface multi-fluid model for simulating multiphase flows. *Applied Mathematical Modelling*, 44:189–204. https://doi.org/10.1016/j.apm.2017.02.030.

[45] Gauss, C. (1903). *Werke*, volume 9 of *Werke*. Springer, Königlichen Gesellschaft der Wissenschaften, Göttingen.

[46] Gingold, R. A. and Monaghan, J. J. (1977). Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181(3):375–389. https://doi.org/10.1093/mnras/181.3.375.

[47] Global Carbon Project (2019). *Supplemental data of Global Carbon Budget 2019.* (Version 1.0).

[48] Godino, D., Corzo, S., and Ramajo, D. (2020). Two-phase modeling of water-air flow of dispersed and segregated flows. *Annals of Nuclear Energy*, 149:107766.

[49] Golub, G. and van Loan, C. (1996). *Matrix computations*. Springer, University of John Hopkins, Baltimore, 3 edition.

[50] Gomez-Fernandez, M., Higley, K., Tokuhiro, A., Welter, K., Wong, W., and Yang, H. (2020). Status of research and development of learning-based approaches in nuclear science and engineering: A review. *Nuclear Engineering and Design*, 359:110479.

[51] Greenshields, C. (2015). Openfoam user guide. *Openfoam Foundation Ltd*, 3(1).

[52] Greenshields, C. (2018). *Funding OpenFOAM in 2019.* [accessed May 27, 2022].

[53] Hammond, J., Pepper, N., Montomoli, F., and Michelassi, V. (2022). Machine learning methods in CFD for turbomachinery: A review. *International Journal of Turbomachinery, Propulsion and Power*, 7(2). http://doi.org/10.3390/ijtpp7020016.

[54] Hann, D., Loizou, K., Vasques, J., Tokarev, M., and Cherdantsev, A. (2018). *The use of POD filtering to study the transition from 2D to 3D in stratified two-phase flow.* 19th International Symposium on the Application of Laser and Imaging Techniques to Fluid Mechanics. 2018: Lisbon, Portugal.

[55] Harlow, F. (1955). *A Machine Calculation Method for Hydrodynamic Problems.* Los Alamos Scientific Laboratory report LAMS-1956.

[56] Harlow, F. and Welch, J. (1965). Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids*, 8(12):2182–2189. http://doi.org/10.1063/1.1761178.

[57] Harten, A. (1983). High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics*, 49(3):357–393. https://doi.org/10.1016/0021-9991(83)90136-5.

[58] Hashmi, A. (2012). *Oil Film Dynamics in Aero Engine Bearing Chambers - Fundamental Investigations and Numerical Modelling.* PhD thesis. Forschungsberichte aus dem Institut für Thermische Str"omungsmaschinen. KIT.

[59] Hashmi, A., Dullenkopf, K., Koch, R., and Bauer, H. (2010). CFD methods for shear driven liquid wall films. volume Volume 4: Heat Transfer, Parts A and B of *Turbo Expo: Power for Land, Sea, and Air*, pages 1283–1291, Glasgow, UK. ASME. http://doi.org/10.1115/GT2010-23532.

[60] Hestenes, M. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409 – 435. http://dx.doi.org/10.6028/jres.049.044.

[61] Hinze, J. (1975). *Turbulence*. 2nd edition. McGraw-Hill.

[62] Hirsch, C. (2007). *Numerical Computation of Internal and External Flows*. Butterworth-Heinemann. The Fundamentals of Computational Fluid Dynamics. 2nd edition. https://doi.org/10.1016/B978-0-7506-6594-0.X5037-1.

[63] Hirt, C. and Nichols, B. (1981). Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201–225. https://doi.org/10.1016/0021-9991(81)90145-5.

[64] Höhne, T. and Vallée, C. (2009). Modelling of stratified two phase flows using an interfacial area density model. *Proc. Multiphase Flow*, pages 123–133. http://doi.org/10.2495/MPF090111.

[65] Hu, Z. and Sandham, N. (2001). *Large–domain simulation of couette and poiseuille flow*. Proc. 2nd Intr. Symposium on Turbulent and Shear Flow Phenomena, (Ed. E. Lindlorg et al.), KTH, Stockholm, Sweden.

[66] Hughes, T. R., Oberai, A., and Mazzei, L. (2001). Large eddy simulation of turbulent channel flows by the variational multiscale method. *Physics of Fluids*, 13(6):1784–1799. http://doi.org/10.1063/1.1367868.

[67] Hunt, J., Wray, A., and Moin, P. (1988). *Eddies, stream, and convergence zones in turbulent flows*. Center for Turbulence Research Report CTR-S88.

[68] IATA (2021). *Net-Zero Carbon Emissions by 2050*. Press Release No: 66. Date: 4 October 2021. [accessed May 17, 2022].

[69] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

[70] Ishii, M. and Hibiki, T. (2010). *Thermo-Fluid Dynamics of Two-PhaseFlow*. Springer, New-York, 2 edition. http://dx.doi.org/10.1007/978-1-4419-7985-8.

[71] Ishii, M. and Mishima, K. (1984). Two-fluid model and hydrodynamic constitutive relations. *Nuclear Engineering and Design*, 82(2):107–126. https://doi.org/10.1016/0029-5493(84)90207-3.

[72] Issa, R. (1986). Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*, 62(1):40 – 65. http://doi.org/10.1016/0021-9991(86)90099-9.

[73] Iungo, G., Viola, F., Ciri, U., Rotea, M., and Leonardi, S. (2015). Data-driven rans for simulations of large wind farms. In *Journal of Physics: Conference Series*, volume 625, page 012025. IOP Publishing. https://doi.org/10.1088/1742-6596/625/1/012025.

[74] Jeffreys, H. and Taylor, G. (1925). On the formation of water waves by wind. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 107(742):189–206. http://doi.org/10.1098/rspa.1925.0015.

[75] Khosravi, A., Pabon, J., Koury, R., and Machado, L. (2018). Using machine learning algorithms to predict the pressure drop during evaporation of r407c. *Applied Thermal Engineering*, 133:361–370. https://doi.org/10.1016/j.applthermaleng.2018.01.084.

[76] Kim, J., Hann, D., and Johnson, K. (2021). *Time-resolved simultaneous PIV measurements of an air-shear-driven thin film flows in a rectangular duct*. RR UTC Tech. report, CornerStone WP 5.2 – Experimental [internal].

[77] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. 3rd International Conference for Learning Representations, San Diego, 2015.

[78] Kochkov, D., Smith, J., Alieva, A., Wang, Q., Brenner, M., and Hoyer, S. (2021). Machine learning-accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118. http://doi.org/10.1073/pnas.2101784118.

[79] Krepper, E., Reddy Vanga, N., Zaruba, A., Prasser, H., and Lopez de Bertodano, M. (2007). Experimental and numerical studies of void fraction distribution in rectangular bubble columns. *Nuclear Engineering and Design*, 237(4):399–408. https://doi.org/10.1016/j.nucengdes.2006.07.009.

[80] Launder, B. and Sharma, B. (1974). Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. *Letters Heat Mass Transfer*, 1:131–137. https://doi.org/10.1016/0094-4548(74)90150-7.

[81] Lee, D., Fahey, D., Skowron, A., Allen, M., Burkhardt, U., Chen, Q., Doherty, S., Freeman, S., Forster, P., Fuglestvedt, J., Gettelman, A., De León, R., Lim, L., Lund, M., Millar, R., Owen, B., Penner, J., Pitari, G., Prather, M., Sausen, R., and Wilcox, L. (2021). The contribution of global aviation to anthropogenic climate forcing for 2000 to 2018. *Atmospheric Environment*, 244:117834.

[82] Leonard, B. (1991). The ultimate conservative difference scheme applied to unsteady one-dimensional advection. *Computer Methods in Applied Mechanics and Engineering*, 88(1):17–74. https://doi.org/10.1016/0045-7825(91)90232-U.

[83] Lin, Z., Liu, X., Lao, L., and Liu, H. (2020). Prediction of two-phase flow patterns in upward inclined pipes via deep learning. *Energy*, 210:118541. https://doi.org/10.1016/j.energy.2020.118541.

[84] Line, A. and Fabre, J. (1997). *Stratified gas-liquid flow*, pages 1097–1101. A-to-Z Guide to Thermodynamics, Heat and Mass Transfer, and Fluids Engineering. CRC Press. http://doig.org/10.1615/AtoZ.s.stratified_gas-liquid_flow.

[85] Ling, J., Kurzawski, A., and Templeton, J. (2016). Reynolds averaged turbulence modelling using deep neural networks with

embedded invariance. *Journal of Fluid Mechanics*, 807:155–166. http://doi.org/10.1017/jfm.2016.615.

[86] Lo, S. and Tomasello, A. (2010). Recent progress in CFD modelling of multiphase flow in horizontal and near-horizontal pipes. volume BHR Group of *7th North American Conference on Multiphase Technology*, Banff, Canada.

[87] Luo, S., Vellakal, M., Koric, S., Kindratenko, V., and Cui, J. (2020). Parameter identification of rans turbulence model using physics-embedded neural network. In Jagode, H., Anzt, H., Juckeland, G., and Ltaief, H., editors, *High Performance Computing*, pages 137–149, Cham. Springer International Publishing.

[88] Maiwald, A. and Schwarze, R. (2011). Numerical analysis of flow-induced gas entrainment in roll coating. *Applied Mathematical Modelling*, 35(7):3516–3526. https://doi.org/10.1016/j.apm.2011.01.004.

[89] Marschall, H. (2011). Towards the numerical simulation of multi-scale two-phase flows.

[90] Menter, F. (1994). Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA Journal*, 32(8):1598–1605. https://doi.org/10.2514/3.12149.

[91] Mirjalili, S., Jain, S., and Dodd, M. (2017). Interface-capturing methods for two-phase flows: An overview and recent developments. *Center for Turbulence Research - Annual research brief*, pages 117–135.

[92] Moin, P. and Mahesh, K. (1998). Direct numerical simulation: A tool in turbulence research. *Annual Review of Fluid Mechanics*, 30(1):539–578. http://doi.org/10.1146/annurev.fluid.30.1.539.

[93] Montanez-Barrera, J. A., Barroso-Maldonado, J. M., Bedoya-Santacruz, A. F., and Mota-Babiloni, A. (2022). Correlated-informed neural networks: a

new machine learning framework to predict pressure drop in micro-channels. http://doi.org/10.48550/ARXIV.2201.07835.

[94] Mouza, A., Paras, S., and Karabelas, A. (2001). CFD code application to wavy stratified gas-liquid flow. *Chemical Engineering Research and Design*, 79(5):561–568. Fluid Flow.

[95] Mukerji, S. (1997). Turbulence computations with 3-d small-scale additive turbulent decomposition and data-fitting using chaotic map combinations. http://doi.org/10.2172/666048.

[96] Noh, W. and Woodward, P. (1976). *SLIC (Simple Line Interface Calculation). [Usable in 1, 2, or 3 space dimensions].* https://doi.org/10.2172/7261651.

[97] Orszag, S. (1970). Analytical theories of turbulence. *Journal of Fluid Mechanics*, 41(2):363–386. https://doi.org/10.1017/S0022112070000642.

[98] Patankar, S. V. (2018). *Numerical heat transfer and fluid flow.* CRC press.

[99] Polansky, J. and Schmelter, S. (2022). Implementation of turbulence damping in the openfoam multiphase flow solver interfoam. *Archives of Thermodynamics*, vol. 43(No 1):21–43. http://doi.org/10.24425/ather.2022.140923.

[100] Pope, S. (2000). *Turbulent flows.* Cambridge University Press, United Kingdom.

[101] Porombka, P. and Höhne, T. (2015). Drag and turbulence modelling for free surface flows within the two-fluid euler–euler framework. *Chemical Engineering Science*, 134:348–359.

[102] Prandtl, L. (1925). *Bericht über Untersuchungen zur ausgebildeten Turbulenz, Z.* Angew. Math, Meth., 5, 136-139.

[103] Prasser, H., Frank, T., Beyer, M., Carl, H., Pietruske, H., and Schütz, P. (2008). Gas-liquid flow around an obstacle in a vertical pipe - experiments and CFD simulation. Nuclear engineering and design, 238(7):1802–1819.

[104] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151. https://doi.org/10.1016/S0893-6080(98)00116-6.

[105] Rakhsha, M., Kees, C., and Negrut, D. (2021). Lagrangian vs. eulerian: An analysis of two solution methods for free-surface flows and fluid solid interaction problems. *Fluids*, 6(12). https://doi.org/10.3390/fluids6120460.

[106] Reboux, S., Sagaut, P., and Lakehal, D. (2006). Large-eddy simulation of sheared interfacial flow. *Physics of Fluids*, 18(10):105105.

[107] Rhie, C. and Chow, W. (1983). Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA Journal*, 21(11):1525–1532. http://doi.org/10.2514/3.8284.

[108] Rios, D. and Muller, P. (1998). Feedforward neural networks for nonparametric regression.

[109] Rolls-Royce Plc (2019). *Advance and UltraFan*. [accessed October 28, 2019].

[110] Rusche, H. (2002). *Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fractions*. PhD thesis.

[111] Russell, T. W. F., Etchells, A. W., Jensen, R. H., and Arruda, P. J. (1974). Pressure drop and holdup in stratified gas-liquid flow. *AIChE Journal*, 20(4):664–669. https://doi.org/10.1002/aic.690200406.

[112] Sagaut, P. and Deck, S. (2009). Large eddy simulation for aerodynamics: Status and perspectives. *Philosophical transactions. Se-*

ries A, Mathematical, physical, and engineering sciences, 367:2849–60. http://doi.org/10.1098/rsta.2008.0269.

[113] Sagaut, P., Deck, S., and Terracol, M. (2013a). *LES, DES and Hybrid RANS/LES Methods: Appliction and Guidelines*. Imperial College Press.

[114] Sagaut, P., Deck, S., and Terracol, M. (2013b). *Multiscale and multiresolution approaches in turbulence (second edition)*, volume 2. Imperial College Press.

[115] Santamaria, A., Mortazavi, M., Chauhan, V., and Benner, J. (2019). Two-phase flow characterization in PEM fuel cells using machine learning. *ECS Meeting Abstracts*, MA2019-01(30):1538–1538. https://doi.org/10.1149/ma2019-01/30/1538.

[116] Santamaria, A., Mortazavi, M., Chauhan, V., Benner, J., Philbrick, O., Clemente, R., Jia, H., and Ling, C. (2021). Machine learning applications of two-phase flow data in polymer electrolyte fuel cell reactant channels. *Journal of The Electrochemical Society*, 168(5):054505. https://doi.org/10.1149/1945-7111/abfa5c.

[117] Schiller, L. and Naumann, A. (1935). A drag coefficient correlation. *Zeitschrift des Vereins Deutscher Ingenieure*, 77:318–320.

[118] Singh, A. (2018). *A Framework to improve Turbulence Models using Full-field Inversion and Machine Learning*. The University of Michigan.

[119] Singh, A., Medida, S., and Duraisamy, K. (2017). Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils. *AIAA Journal*, 55(7):2215–2227. https://doi.org/10.2514/1.J055595.

[120] Smagorinsky, J. (1963). General circulation experiments with the primitive equations. *Monthly Weather Review*, 91(3):99–164. http://doi.org/10.1175/1520-0493(1963)091<0099:GCEWTP>2.3.CO;2.

[121] Sofos, F., Stavrogiannis, C., Exarchou-Kouveli, K., Akabua, D., Charilas, G., and Karakasidis, T. (2022). Current trends in fluid research in the era of artificial intelligence: A review. *Fluids*, 7(3). http://doi.org/10.3390/fluids7030116.

[122] Spalart, P. (2000). Strategies for turbulence modelling and simulations. *International Journal of Heat and Fluid Flow*, 21(3):252–263. https://doi.org/10.1016/S0142-727X(00)00007-2.

[123] Spalart, P. and Allmaras, S. (1992). *A one-equation turbulence model for aerodynamic flows*. 30th Aerospace Sciences Meeting and Exhibit. http://doi.org/10.2514/6.1992-439.

[124] Spalart, P., Jou, W., Strelets, M., and Allmaras, S. (1998). *Comments on the feasibility of LESfor wings and on a hybrid RANS/LES approach*. In Proc. 1st AFSOR Int. Conf. on DNS/LES,Ruston, LA, pp. 137–147.

[125] Stäbler, T. D. (2007). Experimentelle untersuchung und physikalische beschreibung der schichtenströmung in horizontalen kanälen.

[126] Strand, O. (1983). *An Experimental Investigation of Stratified Two-Phase Flow in Horizontal Pipes*. Ph.D. Thesis, University of Oslo, Norway.

[127] Sussman, M., Smereka, P., and S., O. (1994). A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics*, 114(1):146–159. https://doi.org/10.1006/jcph.1994.1155.

[128] Sweby, P. (1984). *High-Resolution Schemes Using Flux Limiters for Hyperbolic Conservation-Laws. Siam Journal on Numerical Analysis*. Siam Journal on Numerical Analysis, 21, 995-1011. http://dx.doi.org/10.1137/0721062.

[129] Taitel, Y. and Dukler, A. (1976). A model for predicting flow regime transitions in horizontal and near horizontal gas-liquid flow. *AIChE Journal*, 22(1):47–55. https://doi.org/10.1002/aic.690220105.

[130] Tekavcic, M., Meller, R., Schlegel, F., and Koncar, B. (2020). *Two-fluid Model Simulations of Isothermal Stratified Counter-current Flow of Air and Water with Interface Compression and Turbulence Damping*. 29th International conference nuclear energy for new europe, Sept. 7-10 2020, Portoroz, Slovenia.

[131] Terzuoli, F., Galassi, M., Mazzini, D., and D'Auria, F. (2008). CFD code validation against stratified air-water flow experimental data. *Science and Technology of Nuclear Installations*, 2008. https://doi.org/10.1155/2008/434212.

[132] Tezduyar, T. E. (2006). Interface-tracking and interface-capturing techniques for finite element computation of moving boundaries and interfaces. *Computer Methods in Applied Mechanics and Engineering*, 195(23):2983–3000. Incompressible CFD.

[133] Tiselj, I., Flageul, C., and Oder, J. (2019). Direct numerical simulation and wall-resolved large eddy simulation in nuclear thermal hydraulics. *Nuclear Technology*, pages 1–15. http://doi.org/10.1080/00295450.2019.1614381.

[134] Tracey, B., Duraisamy, K., and Alonso, J. (2013). *Application of Supervised Learning to Quantify Uncertainties in Turbulence and Combustion Modeling*. http://doi.org/10.2514/6.2013-259.

[135] Tracey, D. (2015). *Machine learning for model uncertainties in turbulence models and Monte Carlo integral approximation*. Stanford University, Department of Aeronautics and Astronautics.

[136] Tryggvason, G., Bunner, B., Esmaeeli, A., Juric, D., Al-Rawahi, N., Tauber, W., Han, J., Nas, S., and Jan, Y.-J. (2001). A front-tracking method for the computations of multiphase flow. *Journal of Computational Physics*, 169(2):708–759. https://doi.org/10.1006/jcph.2001.6726.

[137] Tsoukalas, L., Ishii, M., and Mi, Y. (1997). A neurofuzzy methodology for impedance-based multiphase flow identification. *Engineering Applications of Artificial Intelligence*, 10(6):545–555. https://doi.org/10.1016/S0952-1976(97)00037-7.

[138] Tucker, P. and Davidson, L. (2004). Zonal k–l based large eddy simulations. *Computers and Fluids*, 33(2):267 – 287. http://doi.org/10.1016/S0045-7930(03)00039-2.

[139] Ubbink, O. (1997). Numerical prediction of two fluid systems with sharp interfaces.

[140] Usman, A., Rafiq, M., Saeed, M., Nauman, A., Almqvist, A., and Liwicki, M. (2021). Machine learning computational fluid dynamics. In *2021 Swedish Artificial Intelligence Society Workshop (SAIS)*, pages 1–4. http://doi.org/10.1109/SAIS53221.2021.9483997.

[141] Vallée, C., Höhne, T., Prasser, H.-M., and Sühnel, T. (2008). Experimental investigation and CFD simulation of horizontal stratified two-phase flow phenomena. *Nuclear Engineering and Design*, 238(3):637–646. Benchmarking of CFD Codes for Application to Nuclear Reactor Safety.

[142] van Leer, B. (1979). Towards the ultimate conservative difference scheme. v. a second-order sequel to godunov's method. *Journal of Computational Physics*, 32(1):101 – 136.

[143] Versteeg, H. and Malalasekera, W. (2007). *An introduction to computational fluid dynamics: the finite volume method*. Harlow, England: Pearson Education Ltd.

[144] Vinuesa, R. and Brunton, S. (2021). The potential of machine learning to enhance computational fluid dynamics.

[145] Vlachos, N., Paras, S., and Karabelas, A. (1997). Liquid-to-wall shear stress distribution in stratified/atomization flow. *International Journal of Multiphase Flow*, 23(5):845–863. https://doi.org/10.1016/S0301-9322(97)00007-4.

[146] Von Kármán, T. (1930). *Mechanische Ähnlichkeit und Turbulenz*. Sonderdrucke aus den Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen : Mathematisch-physische Klasse. Weidmannsche Buchh.

[147] Vreman, A. and Kuerten, J. (2014). Comparison of direct numerical simulation databases of turbulent channel flow at $Re_t = 180$. *Physics of Fluids*, 26(1):015102. http://doi.org/10.1063/1.4861064.

[148] Wardle, K. and Weller, H. (2013). Hybrid multiphase CFD solver for coupled dispersed/segregated flows in liquid-liquid extraction. *International Journal of Chemical Engineering*, 2013:13 pages. http://dx.doi.org/10.1155/2013/128936.

[149] Webb, G. (2010). *Overfitting*, pages 744–744. Springer US, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_623.

[150] Weller, H. G., Tabor, G., Jasak, H., and Fureby, C. (1998). A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in physics*, 12(6):620–631.

[151] Wilcox, D. (2008). Formulation of the $k - \omega$ turbulence model revisited. *AIAA Journal*, 46(11):2823–2838. http://doi.org/10.2514/1.36541.

[152] Wintterle, T., Laurien, E., Stäbler, T., Meyer, L., and Schulenberg, T. (2006). *Experimental and Numerical Investigation of Counter-Current Stratified Flows in Horizontal Channels*. CFD4NRS, Garching, Munich, 5-7 September, 2006, Germany.

[153] Wintterle, T., Laurien, E., Stäbler, T., Meyer, L., and Schulenberg, T. (2008). Experimental and numerical investigation of counter-current stratified

flows in horizontal channels. *Nuclear Engineering and Design*, 238(3):627–636. Benchmarking of CFD Codes for Application to Nuclear Reactor Safety.

[154] Wu, J.-L., Wang, J.-X., and Xiao, H. (2016). A bayesian calibration–prediction method for reducing model-form uncertainties with application in rans simulations. *Flow, Turbulence and Combustion*, 97(3):761–786. https://doi.org/10.1007/s10494-016-9725-6.

[155] Xiao, H., Wu, J.-L., Wang, J.-X., Sun, R., and Roy, C. (2016). Quantifying and reducing model-form uncertainties in reynolds-averaged navier–stokes simulations: A data-driven, physics-informed bayesian approach. *Journal of Computational Physics*, 324:115–136. https://doi.org/10.1016/j.jcp.2016.07.038.

[156] Zalesak, S. (1979). Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of Computational Physics*, 31(3):335–362.

$\star$

Appendix A

Code implementations

A.1 C++ pieces of code

A.1.1 Field objects implemented in the solver `interFoam`

Listing A.1 Implementation of q_ε (here called `magsqrDU`) in `createFields.H`

```
volScalarField magsqrDU
(
    IOobject
    (
        "magsqrDU",
        runTime.timeName(),
        mesh,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
    ),
    magSqr(symm(fvc::grad(U)))
);
```

NB: the following line must be added to the VOF solver file `interFoam.C` in order to update the created field:

```
        magsqrDU == magSqr(symm(fvc::grad(U)));
```

A.1.2 Function objects coded in the OpenFOAM simulation control dictionary `controlDict`

Listing A.2 Use of the `fieldAverage` feature

```
fieldAverage
{
    type            fieldAverage;
    libs            ( "libfieldFunctionObjects.so" );
```

```
    cleanRestart    true;
    writeControl    1;

    fields
    (
        U
        {
            mean            on;
            prime2Mean      on;
            base            time;
        }
        magsqrDU
        {
            mean            on;
            prime2Mean      off;
            base            time;
        }
    );
}
```

Listing A.3 Implementation of k

```
tke
{
    functionObjectLibs ("libutilityFunctionObjects.so");
    type            coded;
    name            tke;
    writeControl    runTime;
    writeInterval   1;
    codeWrite
    #{
    const volSymmTensorField& UPrime2Mean = mesh().lookupObject
   <volSymmTensorField>("UPrime2Mean");

    volScalarField tke
    (
        IOobject
        (
            "tke",
            mesh().time().timeName(),
            mesh(),
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        0.5*tr(UPrime2Mean)
        //UPrime2Mean is the Reynolds tensor obtained by field
   averaging
    );
    tke.write();
    #};
}
```

Listing A.4 Implementation of ε

```
epsilon
{
    functionObjectLibs ("libutilityFunctionObjects.so");
    type            coded;
    name            epsilon;
    writeControl    runTime;
    writeInterval   1;
    codeWrite
    #{
    const volScalarField&  nu = mesh().lookupObject<volScalarField>("nu");
    const volScalarField&  magsqrDUMean  = mesh().lookupObject<volScalarField>("magsqrDUMean");

    volScalarField epsilon
    (
        IOobject
        (
            "epsilon",
            mesh().time().timeName(),
            mesh(),
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        2*nu*magsqrDUMean
    );
    epsilon.write();
    #};
}
```

Listing A.5 Implementation of ω

```
omega
{
    functionObjectLibs ("libutilityFunctionObjects.so");
    type            coded;
    name            omega;
    writeControl    runTime;
    writeInterval   1;
    codeWrite
    #{
    const volScalarField&   nu = mesh().lookupObject<volScalarField>("nu");
    const volScalarField&   epsilon = mesh().lookupObject<volScalarField>("epsilon");
    const volScalarField&   tke = mesh().lookupObject<volScalarField>("tke");
    const uniformDimensionedVectorField&  g = mesh().lookupObject<uniformDimensionedVectorField>("g");

    volScalarField omega
    (
    IOobject
        (
```

```
        "omega",
        mesh().time().timeName(),
        mesh(),
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
        ),
       epsilon/(0.09*tke+0.00001*pow(nu*mag(g),2.0/3.0))
       //nu and g used here to add a residual to the ke with the same units (m^2/s^2)
   );
   omega.write();
   #};
}
```

Listing A.6 Implementation of S_{des}

```
Destruction
{
    functionObjectLibs ("libutilityFunctionObjects.so");
    type            coded;
    name            Destruction;
    writeControl    runTime;
    writeInterval   1;
    codeWrite
    #{
    const volScalarField&   omega = mesh().lookupObject<volScalarField>("omega");

    volScalarField Destruction
    (
        IOobject
        (
        "Destruction",
        mesh().time().timeName(),
        mesh(),
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
        ),
        0.072*pow(omega,2) // beta = 0.072
    );
    Destruction.write();
    #};
}
```

Listing A.7 Implementation of S_{prod}

```
Production
{
    functionObjectLibs ("libutilityFunctionObjects.so");
    type            coded;
    name            Production;
    writeControl    runTime;
    writeInterval   1;
    codeWrite
```

```
    #{
    const volVectorField&   U  = mesh().lookupObject<
   volVectorField>("UMean");

    volScalarField Production
    (
        IOobject
        (
            "Production",
            mesh().time().timeName(),
            mesh(),
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        0.52*dev(twoSymm(fvc::grad(U))) && fvc::grad(U)
    );
    Production.write();
    #};
}
```

Listing A.8 Implementation of S_{diff}

```
Diffusion
{
    functionObjectLibs ("libutilityFunctionObjects.so");
    type            coded;
    name            Diffusion;
    writeControl    runTime;
    writeInterval   1;
    codeWrite
    #{
    const volScalarField&   nu = mesh().lookupObject<
   volScalarField>("nu");
    const volScalarField&   k = mesh().lookupObject<
   volScalarField>("tke");
    const volScalarField&   omega = mesh().lookupObject<
   volScalarField>("omega");
    const uniformDimensionedVectorField&  g = mesh().
   lookupObject<uniformDimensionedVectorField>("g");

    volScalarField Diffusion
    (
        IOobject
        (
            "Diffusion",
            mesh().time().timeName(),
            mesh(),
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        fvc::grad((fvc::grad(omega))*(nu+0.5*k/(omega
   +0.0000001*pow(magSqr(g)/nu,1.0/3.0))) ) && I
        //residual must be added to avoid division by zero
    );
```

```
    Diffusion.write();
    #};
}
```

Listing A.9 Implementation of S_{adv}

```
Advection
{
    functionObjectLibs ("libutilityFunctionObjects.so");
    type            coded;
    name            Advection;
    writeControl    runTime;
    writeInterval   1;
    codeWrite
    #{
    const volScalarField&  omega = mesh().lookupObject<
    volScalarField>("omega");
    const volVectorField&  U = mesh().lookupObject<
    volVectorField>("UMean");

    volScalarField Advection
    (
        IOobject
        (
        "Advection",
        mesh().time().timeName(),
        mesh(),
        IOobject::NO_READ,
        IOobject::AUTO_WRITE
        ),
        fvc::grad(omega*U) && I
    );
    Advection.write();
    #};
}
```

Listing A.10 Implementation of S_{ω}

```
budgetOmega
{
    functionObjectLibs ("libutilityFunctionObjects.so");
    type            coded;
    name            budgetOmega;
    writeControl    runTime;
    writeInterval   1;
    codeWrite
    #{
    const volScalarField&  Advection    = mesh().lookupObject<
    volScalarField>("Advection");
    const volScalarField&  Destruction  = mesh().lookupObject<
    volScalarField>("Destruction");
    const volScalarField&  Production   = mesh().lookupObject<
    volScalarField>("Production");
```

```
    const volScalarField&  Diffusion    = mesh().lookupObject<
   volScalarField>("Diffusion");

    volScalarField budgetOmega
    (
        IOobject
        (
            "budgetOmega",
            mesh().time().timeName(),
            mesh(),
            IOobject::NO_READ,
            IOobject::AUTO_WRITE
        ),
        Advection-Production-Diffusion+Destruction
    );
    budgetOmega.write();
    #};
}
```

Listing A.11 Implementation of $||\nabla\alpha||$

```
magGradApha
{
    functionObjectLibs ("libutilityFunctionObjects.so");
    type coded;
    name magGradAlpha;
    writeControl  runTime;
    writeInterval 0.005;
    codeWrite
    #{
        const volScalarField& Alpha1 = mesh().lookupObject<
   volScalarField>("alpha.water");
        volScalarField magGradAlpha
        (
            IOobject
            (
                "magGradAlpha",
                mesh().time().timeName(),
                mesh(),
                IOobject::NO_READ,
                IOobject::AUTO_WRITE
            ),
        mag(fvc::grad(Alpha1))
        );
        magGradAlpha.write();
        #};
}
```

A.1.3 Coded source terms in the `fvOptions` dictionary

Listing A.12 Creation of the velocity sources for each phase

```

```

```
momentumSource0
{
    type            meanVelocityForce;
    active          yes;

    meanVelocityForceCoeffs
    {
        selectionMode   cellSet;
  cellSet   setAir;
        fields    (U);
        Ubar            (4.2 0 0);
  relaxation  1.0;
    }
}

momentumSource1
{
    type            meanVelocityForce;
    active          yes;

    meanVelocityForceCoeffs
    {
        selectionMode   cellSet;
  cellSet   setWater;
        fields    (U);
        Ubar            (0.45 0 0);
  relaxation  1.0;
    }
}
```

Listing A.13 Implementation of the ω correction source term

```
omegaSource
{
    type            scalarCodedSource;
    active          yes;
    name            omegaSource;
    selectionMode   all;

    scalarCodedSourceCoeffs
    {
  selectionMode all;
        fields          (omega);
        codeInclude
        #{
                #include <IFstream.H>
                #include <OFstream.H>
        #};
        codeCorrect
        #{
        #};
        codeAddSup
        #{
          const scalarField& V = mesh_.V();
```

```
23          const vectorField& C = mesh_.C();
24          //reading the correction source field interpolated
    from qDNS:
25          const scalarField& CORR = IFstream("./constant/
    budgetOmega")();
26          scalarField& budgetOmegaSource = eqn.source();
27
28            forAll (C,i)
29            {
30                budgetOomegaSource[i] = -(CORR[i]) * V[i];
31            }
32        #};
33        codeSetValue
34        #{
35        #};
36        code
37        #{
38            $codeInclude
39            $codeCorrect
40            $codeAddSup
41            $codeSetValue
42        #};
43    }
44    sourceTimeCoeffs
45    {
46       selectionMode   all;
47    }
48 }
```

A.1.4 Modified turbulence model

The RANS $k-\omega$ turbulence model was modified in order to be able to print out the different terms of the transport of ω and check the budget and the corrected source evolution progressions during simulations.

Listing A.14 Modified $k-\omega$ model: `newkOmega.C`

```
1  #include "newkOmega.H"
2  #include "fvOptions.H"
3  #include "bound.H"
4
5  namespace Foam
6  {
7  namespace RASModels
8  {
9
10 // * * * * * * * * * * * * * * Protected Member Functions  * * * *
       * * * * * * * * //
11 template<class BasicTurbulenceModel>
12 void pkOmega3<BasicTurbulenceModel>::correctNut()
13 {
14     this->nut_ = k_/omega_;
15     this->nut_.correctBoundaryConditions();
```

```
16      fv::options::New(this->mesh_).correct(this->nut_);
17
18      BasicTurbulenceModel::correctNut();
19  }
20  template<class BasicTurbulenceModel>
21  tmp<fvScalarMatrix> pkOmega3<BasicTurbulenceModel>::kSource()
        const
22  {
23      return tmp<fvScalarMatrix>
24      (
25          new fvScalarMatrix
26          (
27              k_,
28              dimVolume*this->rho_.dimensions()*k_.dimensions()
29              /dimTime
30          )
31      );
32  }
33  template<class BasicTurbulenceModel>
34  tmp<fvScalarMatrix> pkOmega3<BasicTurbulenceModel>::omegaSource
        () const
35  {
36      return tmp<fvScalarMatrix>
37      (
38          new fvScalarMatrix
39          (
40              omega_,
41              dimVolume*this->rho_.dimensions()*omega_.dimensions
        ()/dimTime
42          )
43      );
44  }
45  // * * * * * * * * * * * * * * * * * * Constructors  * * * * * * *
        * * * * * * * //
46  template<class BasicTurbulenceModel>
47  pkOmega3<BasicTurbulenceModel>::pkOmega3
48  (
49      const alphaField& alpha,
50      const rhoField& rho,
51      const volVectorField& U,
52      const surfaceScalarField& alphaRhoPhi,
53      const surfaceScalarField& phi,
54      const transportModel& transport,
55      const word& propertiesName,
56      const word& type
57  )
58  :
59      eddyViscosity<RASModel<BasicTurbulenceModel>>
60      (
61          type,
62          alpha,
63          rho,
64          U,
```

```
65          alphaRhoPhi,
66          phi,
67          transport,
68          propertiesName
69      ),
70      Cmu_
71      (
72          dimensioned<scalar>::lookupOrAddToDict
73          (
74              "betaStar",
75              this->coeffDict_,
76              0.09
77          )
78      ),
79      beta_
80      (
81          dimensioned<scalar>::lookupOrAddToDict
82          (
83              "beta",
84              this->coeffDict_,
85              0.072
86          )
87      ),
88      gamma_
89      (
90          dimensioned<scalar>::lookupOrAddToDict
91          (
92              "gamma",
93              this->coeffDict_,
94              0.52
95          )
96      ),
97      alphaK_
98      (
99          dimensioned<scalar>::lookupOrAddToDict
100         (
101             "alphaK",
102             this->coeffDict_,
103             0.5
104         )
105     ),
106     alphaOmega_
107     (
108         dimensioned<scalar>::lookupOrAddToDict
109         (
110             "alphaOmega",
111             this->coeffDict_,
112             0.5
113         )
114     ),
115     k_
116     (
117         IOobject
```

```
118         (
119             IOobject::groupName("k", alphaRhoPhi.group()),
120             this->runTime_.timeName(),
121             this->mesh_,
122             IOobject::MUST_READ,
123             IOobject::AUTO_WRITE
124         ),
125         this->mesh_
126     ),
127     omega_
128     (
129         IOobject
130         (
131             IOobject::groupName("omega", alphaRhoPhi.group()),
132             this->runTime_.timeName(),
133             this->mesh_,
134             IOobject::MUST_READ,
135             IOobject::AUTO_WRITE
136         ),
137         this->mesh_
138     ),
139     fieldDDT_
140     (
141         IOobject
142         (
143             IOobject::groupName("DDTOmega", alphaRhoPhi.group()
    ),
144             this->runTime_.timeName(),
145             this->mesh_,
146             IOobject::NO_READ,
147             IOobject::AUTO_WRITE
148         ),
149         this->mesh_,
150         dimensionedScalar("0",omega_.dimensions()/dimTime,0)
151     ),
152     fieldADV_
153     (
154         IOobject
155         (
156             IOobject::groupName("AdvectionOmega", alphaRhoPhi.
    group()),
157             this->runTime_.timeName(),
158             this->mesh_,
159             IOobject::NO_READ,
160             IOobject::AUTO_WRITE
161         ),
162         this->mesh_,
163         dimensionedScalar("0",omega_.dimensions()/dimTime,0)
164     ),
165     fieldDIFF_
166     (
167         IOobject
168         (
```

```
169             IOobject::groupName("DiffusionOmega", alphaRhoPhi.
    group()),
170             this->runTime_.timeName(),
171             this->mesh_,
172             IOobject::NO_READ,
173             IOobject::AUTO_WRITE
174         ),
175         this->mesh_,
176         dimensionedScalar("0",omega_.dimensions()/dimTime,0)
177     ),
178     fieldDESTR_
179     (
180         IOobject
181         (
182             IOobject::groupName("DestructionOmega", alphaRhoPhi
    .group()),
183             this->runTime_.timeName(),
184             this->mesh_,
185             IOobject::NO_READ,
186             IOobject::AUTO_WRITE
187         ),
188         this->mesh_,
189         dimensionedScalar("0",omega_.dimensions()/dimTime,0)
190     ),
191     fieldPROD_
192     (
193         IOobject
194         (
195             IOobject::groupName("ProductionOmega", alphaRhoPhi.
    group()),
196             this->runTime_.timeName(),
197             this->mesh_,
198             IOobject::NO_READ,
199             IOobject::AUTO_WRITE
200         ),
201         this->mesh_,
202         dimensionedScalar("0",omega_.dimensions()/dimTime,0)
203     ),
204     fieldPROD2_
205     (
206         IOobject
207         (
208             IOobject::groupName("ProductionOmega2", alphaRhoPhi
    .group()),
209             this->runTime_.timeName(),
210             this->mesh_,
211             IOobject::NO_READ,
212             IOobject::AUTO_WRITE
213         ),
214         this->mesh_,
215         dimensionedScalar("0",omega_.dimensions()/dimTime,0)
216     ),
217     fieldSOURCE_
```

```
218     (
219         IOobject
220         (
221             IOobject::groupName("SourceOmega", alphaRhoPhi.
    group()),
222             this->runTime_.timeName(),
223             this->mesh_,
224             IOobject::NO_READ,
225             IOobject::AUTO_WRITE
226         ),
227         this->mesh_,
228         dimensionedScalar("0",omega_.dimensions()/dimTime,0)
229     )
230 {
231     bound(k_, this->kMin_);
232     bound(omega_, this->omegaMin_);
233 
234     if (type == typeName)
235     {
236         this->printCoeffs(type);
237     }
238 }
239 // * * * * * * * * * * * * * * * * * * Member Functions  * * * * * *
    * * * * * * * * * //
240 template<class BasicTurbulenceModel>
241 bool pkOmega3<BasicTurbulenceModel>::read()
242 {
243     if (eddyViscosity<RASModel<BasicTurbulenceModel>>::read())
244     {
245         Cmu_.readIfPresent(this->coeffDict());
246         beta_.readIfPresent(this->coeffDict());
247         gamma_.readIfPresent(this->coeffDict());
248         alphaK_.readIfPresent(this->coeffDict());
249         alphaOmega_.readIfPresent(this->coeffDict());
250 
251         return true;
252     }
253     else
254     {
255         return false;
256     }
257 }
258 template<class BasicTurbulenceModel>
259 void pkOmega3<BasicTurbulenceModel>::correct()
260 {
261     if (!this->turbulence_)
262     {
263         return;
264     }
265     // Local references
266     const alphaField& alpha = this->alpha_;
267     const rhoField& rho = this->rho_;
268     const surfaceScalarField& alphaRhoPhi = this->alphaRhoPhi_;
```

```
269     const volVectorField& U = this->U_;
270     volScalarField& nut = this->nut_;
271     fv::options& fvOptions(fv::options::New(this->mesh_));
272     eddyViscosity<RASModel<BasicTurbulenceModel>>::correct();
273     volScalarField::Internal divU
274     (
275         fvc::div(fvc::absolute(this->phi(), U))().v()
276     );
277     tmp<volTensorField> tgradU = fvc::grad(U);
278     volScalarField::Internal G
279     (
280         this->GName(),
281         nut.v()*(dev(twoSymm(tgradU().v())) && tgradU().v())
282     );
283     tgradU.clear();
284 
285     // Update omega and G at the wall
286     omega_.boundaryFieldRef().updateCoeffs();
287     // Turbulent frequency equation
288     fvScalarMatrix omegaEqn1
289     (
290         fvm::ddt(alpha, rho, omega_)
291     );
292     fvScalarMatrix omegaEqn2
293     (
294         fvm::div(alphaRhoPhi, omega_)
295     );
296     fvScalarMatrix omegaEqn3
297     (
298         fvm::laplacian(alpha*rho*DomegaEff(), omega_)
299     );
300     fvScalarMatrix omegaEqn4
301     (
302         fvm::Sp(beta_*alpha()*rho()*omega_(), omega_)
303     );
304     fvScalarMatrix omegaEqn5
305     (
306         fvm::SuSp(gamma_*alpha()*rho()*G/k_(), omega_)
307     );
308     fvScalarMatrix omegaEqn52
309     (
310               fvm::SuSp(((2.0/3.0)*gamma_)*alpha()*rho()*divU,
    omega_)
311     );
312     fvScalarMatrix omegaEqn6
313     (
314         (
315                   0.0*fvm::ddt(alpha, rho, omega_) // 0
316             +  omegaSource()
317             + fvOptions(alpha, rho, omega_)
318         )
319     );
320 tmp<fvScalarMatrix> omegaEqn
```

```
321     (
322     omegaEqn1
323     + omegaEqn2
324     ==
325     omegaEqn3
326     - omegaEqn4
327     + omegaEqn5
328     - omegaEqn52
329     + omegaEqn6
330     );
331 omegaEqn.ref().relax();
332     fvOptions.constrain(omegaEqn.ref());
333     omegaEqn.ref().boundaryManipulate(omega_.boundaryFieldRef()
        );
334     solve(omegaEqn);
335     fvOptions.correct(omega_);
336     bound(omega_, this->omegaMin_);
337 //- Update fields
338     fieldDDT_     = omegaEqn1.A()*omega_ - omegaEqn1.H();
339     fieldADV_     = omegaEqn2.A()*omega_ - omegaEqn2.H();
340     fieldDIFF_    = omegaEqn3.A()*omega_ - omegaEqn3.H();
341     fieldDESTR_   = omegaEqn4.A()*omega_ - omegaEqn4.H();
342     fieldPROD_    = omegaEqn5.A()*omega_ - omegaEqn5.H();
343     fieldPROD2_    = omegaEqn52.A()*omega_ - omegaEqn52.H();
344     fieldSOURCE_ = omegaEqn6.A()*omega_ - omegaEqn6.H();
345 // Turbulent kinetic energy equation
346     tmp<fvScalarMatrix> kEqn
347     (
348         fvm::ddt(alpha, rho, k_)
349       + fvm::div(alphaRhoPhi, k_)
350       - fvm::laplacian(alpha*rho*DkEff(), k_)
351      ==
352         alpha()*rho()*G
353       - fvm::SuSp((2.0/3.0)*alpha()*rho()*divU, k_)
354       - fvm::Sp(Cmu_*alpha()*rho()*omega_(), k_)
355       + kSource()
356       + fvOptions(alpha, rho, k_)
357     );
358 kEqn.ref().relax();
359     fvOptions.constrain(kEqn.ref());
360     solve(kEqn);
361     fvOptions.correct(k_);
362     bound(k_, this->kMin_);
363
364     correctNut();
365 }
366 // * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
    * * * * * * * //
367 } // End namespace RASModels
368 } // End namespace Foam
```

A.2 Python scripts

Listing A.15 `shell_input.py`: Pre-processing of the inputs for the ML model M3 prediction phase, to be used with `shell_script.py`

```
##############################################
# Should be automatically imported with iPython
# Uncomment/comment if needed
import numpy as np
import pandas as pd
import os
##############################################
# EDIT BELOW
# numbers of cells:
ny_air      = 57
ny_water    = 15
nx          = 28
nz          = 8
ny          = ny_air+ny_water
# geometry vertically: (only needed if geometry dependant input selected)
y_top       = 0.02600 # top wall y-coordinate
y_bott      = 0.00000 # bottom wall y-coordinate
y_interf    = 0.00233 # mean interface setting y-coordinate

dirpath='./'
dirs = [s for s in os.listdir(dirpath) if os.path.isdir(os.path.join(dirpath, s))]
dirs.sort(key=lambda s: os.path.getmtime(os.path.join(dirpath, s)), reverse=True)
mostRecent_dirPath = dirpath+str(dirs[0])+"/"
print ('LAST WRITTEN TIME_STEP = ', dirs[0], ' s')

# lines to skip in OpenFOAM files access true data
skipFirstLines = 22

file = open('./0/Cy', 'r') #y mesh coordinates
lines = file.readlines()
file.close()
del lines[nx*ny*nz+skipFirstLines:]
del lines[0:skipFirstLines]
new_file = open('Cy', 'w+')
for line in lines:
    new_file.write(line)
new_file.close()

y_line_mesh = np.loadtxt('./Cy').tolist()

#############
# ML FEATURES
#############

# phase volume distribution
```

```
a_file = open(mostRecent_dirPath+'alpha.water', 'r')
lines = a_file.readlines()
a_file.close()
del lines[nx*ny*nz+skipFirstLines:]
del lines[0:skipFirstLines]
new_file = open('alpha', 'w+')
for line in lines:
    new_file.write(line)
new_file.close()

alpha   = np.loadtxt('./alpha')

# ||grad(alpha)||
a_file = open(mostRecent_dirPath+'magGradAlpha', 'r')
lines = a_file.readlines()
a_file.close()
del lines[nx*ny*nz+skipFirstLines:]
del lines[0:skipFirstLines]
new_file = open('magGradAlpha', 'w+')
for line in lines:
    new_file.write(line)
new_file.close()

maggradalpha   = np.loadtxt('./magGradAlpha')

# mean axial velocity
a_file = open(mostRecent_dirPath+'U', 'r')
lines = a_file.readlines()
a_file.close()
del lines[nx*ny*nz+skipFirstLines:]
del lines[0:skipFirstLines]
new_file = open('velocity', 'w+')
for line in lines:
    new_file.write(line)
new_file.close()
with open('./velocity', 'r') as file1 :
    filedata = file1.read()
    filedata = filedata.replace('(', ' ')
    filedata = filedata.replace(')', ' ')
with open('velocity', 'w') as file:
    file.write(filedata)

u = np.loadtxt('./velocity')[:,0]

# TKE
a_file = open(mostRecent_dirPath+'k', 'r')
lines = a_file.readlines()
a_file.close()
del lines[nx*ny*nz+skipFirstLines:]
del lines[0:skipFirstLines]
new_file = open('tke', 'w+')
for line in lines:
    new_file.write(line)
```

```
new_file.close()

tke    = np.loadtxt('./tke')

########################################
# Cration of dataframe for ML prediction
########################################
unitaire = [1]*nx*ny*nz

datalabels = {'eta1': u, 'eta2': alpha, 'eta3': maggradalpha, 'eta4': tke, 'unitaire': unitaire}
df = pd.DataFrame(data = datalabels)

df.to_csv("dataset_test.csv", index=False)
```

Listing A.16 `shell_script.py`: Shell script used to perform the adaptive correction method with M3 (coupling OpenFOAM/ML

```
import os
import time

SIMULATION_DURATION = 7200 # second is the unit
dirpath='./'

# Makes of ./0 the most recent dir
os.system(">> ./0/hop")
os.system("rm ./0/hop")

time.sleep(1)
time_0 = time.time() # reference start time

while (time.time()-time_0) < SIMULATION_DURATION:
    dirs = [s for s in os.listdir(dirpath) if os.path.isdir(os.path.join(dirpath, s))]
    dirs.sort(key=lambda s: os.path.getmtime(os.path.join(dirpath, s)), reverse=True)
# Initialisation loop
    if float(dirs[0]) == 0:
        print(">>>>>>>>>>> ONLY dir 0 DETECTED --> USING INIT FVOPTIONS... <<<<<<<<<<<")
        os.system("cp ./constant/fvOptions_init ./constant/fvOptions")
        os.system("postProcess -func writeCellCentres")
        os.system("setFields")
        os.system("topoSet")
        print(">>>>>>>>>>> CONVERTING NOTEBOOK <<<<<<<<<<<")
        os.system("jupyter nbconvert --to script prediction.ipynb")
        time.sleep(0.1)
        os.system("interFoam") # to initialise field for inputs ML (U, omega)
        time.sleep(0.1)
        os.system("ipython shell_input.py")
        os.system("ipython prediction.py")
```

```
        os.system("cp ./constant/fvOptions_run ./constant/
    fvOptions")
        time.sleep(0.1)
        os.system("interFoam")
        time.sleep(0.1)
     if float(dirs[0]) != 0:
        print("Sorting inputs...")
        os.system("ipython shell_input.py")
        time.sleep(0.1)
        print("Making predictions...")
        os.system("ipython prediction.py")
        time.sleep(0.1)
        os.system("interFoam")
        time.sleep(0.1)
```

Listing A.17 `clean.py`: script used to clean data after simulation performed with the adaptive method

```
import os

os.system("rm ./constant/fvOptions")
os.system("rm -r 0.* 1.* 2.* 3.* 4.* 5.* 6.* 7.* 8.* 9.*")
os.system("rm ./0/C*")
os.system("rm ./0/alpha.water")
os.system("rm prediction.py")
os.system("rm alpha")
os.system("rm dataset_*")
os.system("rm velocity")
#os.system("rm -r dynamicCode")
os.system("rm ./constant/budget*")
os.system("rm ./0/hop")
os.system("rm ./magGradAlpha")
os.system("rm ./Cy")
```

A.3 PyTorch model notebooks

Listing A.18 `training.ipynb`: jupyter notebook used to implement the neural network of M3 and perform the training of the model

```
import pandas as pd
import torch
import matplotlib.pyplot as plt
import numpy as np
import os

torch.__version__

# Source data path (.csv format)
dataset_path = 'dataset.csv'
# Number of outputs for the regression problem
n_outputs = 1
```

```
# Scaling technique
scaler = 'standard' # standard . maxmin . maxabs  . None
# Saving path for my best torch model state dictionary and
   params dictionary
save_path = './model_M3/'
# data shuffle
shuffle_data = True
# Number of neurons for the layers of the MLP
neurons = [512, 512]
# Activation function for hidden layers
activation = 'relu' # relu . tanh
dropout_rate = 0.2
learning_rate = 10**-4
batch_size = 64
# Training epochs
epochs = 1024
# Loss function
loss = 'mse'  # mse # mae
# Optimizer
optimizer_type = 'adam' # adam . sgd . rmsprop

if not os.path.exists(save_path):
  os.mkdir(save_path)

params_dict = {
    'dataset_path': dataset_path,
    'drop_columns': drop_columns,
    'n_outputs': n_outputs,
    'beta_threshold': beta_threshold,
    'scaler': scaler,
    'save_path': save_path,
    'shuffle_data': shuffle_data,
    'neurons': neurons,
    'dropout_rate': dropout_rate,
    'learning_rate': learning_rate,
    'batch_size': batch_size,
    'epochs': epochs,
    'loss': loss,
    'optimizer_type': optimizer_type,
}

dataset = pd.read_csv(dataset_path,
                      dtype=np.float32,
                      na_values=['?'])

print(f'Original Dataset shape: {dataset.shape}')
if not dataset.isnull().values.any():
  print('No NaN in dataset')
else:
  print('Removing NaN values...\n')
  dataset.dropna(inplace=True)
  print(f'New Dataset shape: {dataset.shape}')

```

```
print('Checking data distribution... \n')
plt.style.use('seaborn-darkgrid')
dataset.hist(figsize=(16, 10), bins=20)
plt.show()

from sklearn.preprocessing import MinMaxScaler, StandardScaler,
    MaxAbsScaler
from sklearn.model_selection import train_test_split

def scale(df: pd.DataFrame,
          scaler: str = 'standard') -> pd.DataFrame:
  if scaler == 'standard':
    scaler = StandardScaler()
  elif scaler == 'maxmin':
    scaler = MinMaxScaler()
  elif scaler == 'maxabs':
    scaler = MaxAbsScaler()
  else:
    return df

  scaler.fit(df)
  return pd.DataFrame(scaler.transform(df), columns=df.columns)
    , scaler

normed_dataset, my_scaler = scale(dataset, scaler)

print(f'Splitting data in 3 dataset for training - validation -
    test... (shuffle: {shuffle_data})\n')

if shuffle_data:
    train, validate, test = np.split(normed_dataset.sample(frac
    =1, random_state=22),
                                     [int(.78*len(normed_dataset))
    , int(.95*len(normed_dataset))])
else:
    train, validate, test = np.split(normed_dataset,
                                     [int(.78*len(normed_dataset))
    , int(.95*len(normed_dataset))])

import torch.nn as nn
import torch

# Implementation of the feed-forward neural network
class RegNet1D(nn.Module):
    def __init__(self,
                 num_feature: int = 4,
                 num_class: int = 1,
                 dropout_rate: float = 0.0,
                 neurons: list = neurons,
                 activation: str = 'relu') -> None:

        super(RegNet1D, self).__init__()

```

```
112         self.fc1 = nn.Linear(num_feature, neurons[0])
113         self.fc2 = nn.Linear(neurons[0], neurons[1])
114 
115         self.layer_out = nn.Linear(neurons[1], num_class)
116 
117         if activation == 'relu':
118           self.act = nn.ReLU()
119         elif activation == 'tanh':
120           self.act = nn.Tanh()
121         else:
122           self.act = nn.ReLU()
123 
124         self.dropout = nn.Dropout(p=dropout_rate)
125         self.bn1 = nn.BatchNorm1d(neurons[0])
126         self.bn2 = nn.BatchNorm1d(neurons[1])
127 
128     def forward(self, x) -> torch.Tensor:
129         x = self.fc1(x)
130         x = self.bn1(x)
131         x = self.act(x)
132         x = self.dropout(x)
133 
134         x = self.fc2(x)
135         x = self.bn2(x)
136         x = self.act(x)
137         x = self.dropout(x)
138 
139         x = self.layer_out(x)
140 
141         return x
142 
143 device = torch.device("cuda:0" if torch.cuda.is_available()
    else "cpu")
144 print(f'working on {device}')
145 
146 model = RegNet1D(
147     num_feature=train.shape[1] - n_outputs,
148     num_class=1,
149     neurons=neurons,
150     dropout_rate=dropout_rate,
151     activation=activation
152     ).to(device)
153 
154 #import torchsummary
155 #torchsummary.summary(model, (train.shape[1] - n_outputs,),
    batch_size=batch_size)
156 
157 print(f'Loss function: {loss}\n')
158 if loss == 'mse':
159   criterion = nn.MSELoss()
160 elif loss == 'mae':
161   criterion = nn.L1Loss()
162 else:
```

```
  criterion = nn.MSELoss()

print(f'optimizer: {optimizer_type}\n')

if optimizer_type == 'adam':
  optimizer = torch.optim.Adam(model.parameters(), lr=
   learning_rate)
elif optimizer_type == 'sgd':
  optimizer = torch.optim.SGD(model.parameters(), lr=
   learning_rate, momentum=0.9, nesterov=True)
elif optimizer_type == 'rmsprop':
  optimizer = torch.optim.RMSprop(model.parameters(), lr=
   learning_rate)
else:
  optimizer = torch.optim.Adam(model.parameters(), lr=
   learning_rate)

class CustomLogger(object):
    def __init__(self):
        self.accuracy = {
            'train': [],
            'val': []
        }
        self.loss = {
            'train': [],
            'val': []
        }

n_inputs = train.shape[1] - n_outputs
X_train = train.to_numpy()[:, :n_inputs]
X_val = validate.to_numpy()[:, :n_inputs]
X_test = test.to_numpy()[:, :n_inputs]
y_train = train.to_numpy()[:, n_inputs:n_inputs + n_outputs]
y_val = validate.to_numpy()[:, n_inputs:n_inputs + n_outputs]
y_test = test.to_numpy()[:, n_inputs:n_inputs + n_outputs]

from torch.utils.data import Dataset, DataLoader

class RegressionDataset(Dataset):

    def __init__(self, X_data, y_data):
        self.X_data = X_data
        self.y_data = y_data

    def __getitem__(self, index):
        return self.X_data[index], self.y_data[index]

    def __len__ (self):
        return len(self.X_data)

train_dataset = RegressionDataset(torch.from_numpy(X_train).
   float(), torch.from_numpy(y_train).float())
```

```
val_dataset = RegressionDataset(torch.from_numpy(X_val).float(), torch.from_numpy(y_val).float())

from sklearn.metrics import r2_score
import copy

# Training of the model
print("Starting Training...\n")
Logger = CustomLogger()
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=shuffle_data, drop_last=True)
val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=shuffle_data, drop_last=True)

model.train()
best_model = None
best_acc = -1000

for epoch in range(0, epochs):
    train_epoch_loss = 0
    train_epoch_acc = 0
    for X_train_batch, y_train_batch in train_loader:
        X_train_batch, y_train_batch = X_train_batch.to(device), y_train_batch.to(device).view(-1, 1)

        optimizer.zero_grad()

        y_train_pred = model(X_train_batch)
        train_loss = criterion(y_train_pred, y_train_batch)

        train_acc = r2_score(y_train_pred.cpu().detach().numpy(),
                             y_train_batch.cpu().detach().numpy())

        train_loss.backward()
        optimizer.step()

        train_epoch_loss += train_loss.item()
        train_epoch_acc += train_acc.item()

    with torch.no_grad():

        val_epoch_loss = 0
        val_epoch_acc = 0

        model.eval()
        for X_val_batch, y_val_batch in val_loader:
            X_val_batch, y_val_batch = X_val_batch.to(device), y_val_batch.to(device).view(-1, 1)

            y_val_pred = model(X_val_batch)

```

```
            val_loss = criterion(y_val_pred, y_val_batch)
            val_acc = r2_score(y_val_pred.cpu().detach().numpy(),
                               y_val_batch.cpu().detach().numpy())

            val_epoch_loss += val_loss.item()
            val_epoch_acc += val_acc.item()

    Logger.loss['train'].append(train_epoch_loss/len(train_loader))
    Logger.loss['val'].append(val_epoch_loss/len(val_loader))
    Logger.accuracy['train'].append(train_epoch_acc/len(train_loader))
    Logger.accuracy['val'].append(val_epoch_acc/len(val_loader))

    print(f'Epoch {epoch+1+0:04}: | Train Loss: {train_epoch_loss/len(train_loader):.5f} | Val Loss: {val_epoch_loss/len(val_loader):.5f} | Train Acc: {train_epoch_acc/len(train_loader):.3f} | Val Acc: {val_epoch_acc/len(val_loader):.3f} |')

    if val_epoch_acc > best_acc:
        print('Saving a new best model')
        best_model = copy.deepcopy(model)
        best_acc = val_epoch_acc


plt.style.use('seaborn-darkgrid')
fig, axs = plt.subplots(ncols=2, figsize=(10, 4))
axs[0].plot(Logger.loss['train'], label='Train loss')
axs[0].plot(Logger.loss['val'], label='Val loss')
axs[0].legend()
axs[1].plot(Logger.accuracy['train'], label='Train accuracy')
axs[1].plot(Logger.accuracy['val'], label='Val accuracy')
axs[1].legend()
plt.show()

# Saving the model
print('Saving best model to file... \n')
model = best_model
torch.save(model.state_dict(), os.path.join(save_path, 'model_line1.pth'))
print('Saving my simulation parameters...\n')
import json
with open(os.path.join(save_path, 'params.json'), 'w') as fp:
    json.dump(params_dict, fp)
model.eval()

# Testing phase
print('Making predictions...')
```

```
train_preds = model(torch.Tensor(X_train).to(device)).cpu().detach().numpy()
val_preds = model(torch.Tensor(X_val).to(device)).cpu().detach().numpy()
test_preds = model(torch.Tensor(X_test).to(device)).cpu().detach().numpy()

from sklearn.metrics import mean_absolute_error, mean_squared_error

def results_log(_t, _p, set:str = ''):
  print(f'Dataset: {set} - mae: {mean_absolute_error(_t, _p)} - mse: {mean_squared_error(_t, _p)} - r2: {r2_score(_t, _p)}\n')

results_log(train_preds, y_train, 'train')
results_log(val_preds, y_val, 'val')
results_log(test_preds, y_test, 'test')

print('Reloading parameters if needed...')
with open(os.path.join(save_path, 'params.json')) as f:
params = json.load(f)

# Save the scaler
from pickle import dump
dump(my_scaler, open(save_path+'/scaler_M3.pkl', 'wb'))
```

Listing A.19 `prediction.py`: Converted jupyter notebook used to perform the predictions of the ML model M3

```

#!/usr/bin/env python
# coding: utf-8

#import torch.nn as nn
#import torch
#import pandas as pd
#import numpy as np
#import os
#from pickle import load

save_path = './model_M3/'
saved_scalar_name = 'scaler_M3.pkl'
saved_model = 'model_M3.pth'
dataSet_name = './dataset.csv'
neurons = 512
num_inputs = 4
neurons = [neurons, neurons,]
activation = 'relu'
dropout_rate = 0.2

class RegNet1D(nn.Module):
    def __init__(self,
                 num_feature: int = 10,
```

```
                  num_class: int = 1,
                  dropout_rate: float = 0.0,
                  neurons = neurons, # Number of neurons for the
    layers of the MLP
                  activation: str = 'relu') -> None:

        super(RegNet1D, self).__init__()

        self.fc1 = nn.Linear(num_feature, neurons[0])
        self.fc2 = nn.Linear(neurons[0], neurons[1])

        self.layer_out = nn.Linear(neurons[1], num_class)

        if activation == 'relu':
          self.act = nn.ReLU()
        elif activation == 'tanh':
          self.act = nn.Tanh()
        else:
          self.act = nn.ReLU()

        self.dropout = nn.Dropout(p=dropout_rate)
        self.bn1 = nn.BatchNorm1d(neurons[0])
        self.bn2 = nn.BatchNorm1d(neurons[1])

    def forward(self, x) -> torch.Tensor:
        x = self.fc1(x)
        x = self.bn1(x)
        x = self.act(x)
        x = self.dropout(x)
        x = self.fc2(x)
        x = self.bn2(x)
        x = self.act(x)
        x = self.dropout(x)
        x = self.layer_out(x)

        return x

device = torch.device("cuda:0" if torch.cuda.is_available()
    else "cpu")
reloaded_model = RegNet1D(
                          num_feature= num_inputs,
                          num_class=1,
                          neurons=neurons,
                          dropout_rate=dropout_rate
                          ).to(device)
reloaded_model.load_state_dict(torch.load(os.path.join(
    save_path, saved_model)))
reloaded_model.eval()

dataset = pd.read_csv(dataSet_name, dtype=np.float32, na_values
    =['?'])

scaler = 'standard'
```

```
scaler_model = load(open(save_path+saved_scalar_name, 'rb'))
normed_dataset = scaler_model.transform(dataset)
X_data = normed_dataset[:, :-1]
pred_data = reloaded_model(torch.Tensor(X_data).to(device)).cpu
    ().detach().numpy()
#remove 'unitaire' that was used for scaling:
stacked_dataset = np.hstack((normed_dataset[:,:-1],pred_data))
inv_normed_dataset = scaler_model.inverse_transform(
    stacked_dataset)
beta_rescaled = inv_normed_dataset[:,-1]
beta_rescaled_nolog = [10**(beta_rescaled[i]) for i in range(
    len(beta_rescaled))]
beta_rescaled_ = ['(']+beta_rescaled_nolog+[')']
np.savetxt('./constant/budgetOmega', beta_rescaled_, fmt='%s')
```

$\star$

Appendix B

Additional figures

B.1 Additional figures from chapter 5

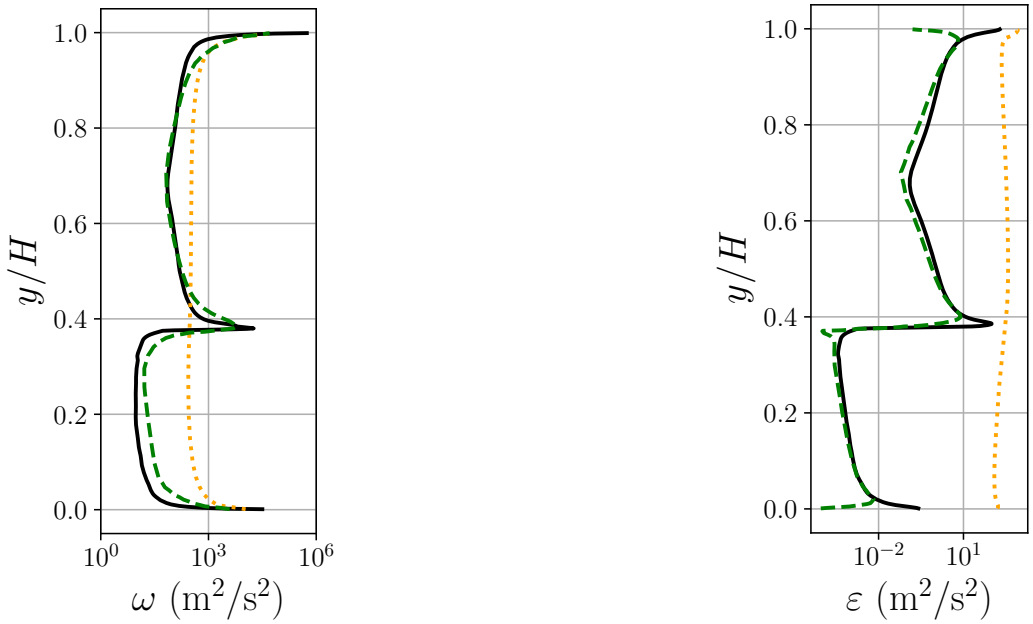


Figure B.1 Specific turbulence dissipation rate (left) and turbulence dissipation rate (right) profiles comparison between the qDNS, the standard $k-\omega$ model (stand.), and the corrected $k-\omega$ model (corr.) predictions in the closed channel configuration and wavy interface regime

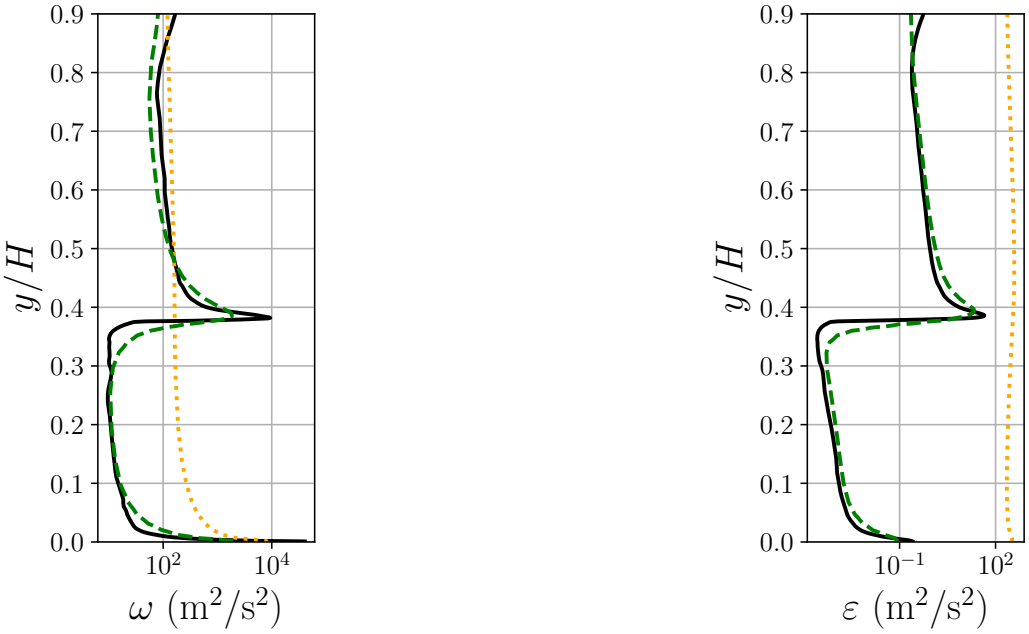


Figure B.2 profiles comparison between the qDNS, the standard $k-\omega$ model (stand.), and the corrected $k-\omega$ model (corr.) predictions in the open channel configuration and smooth interface regime

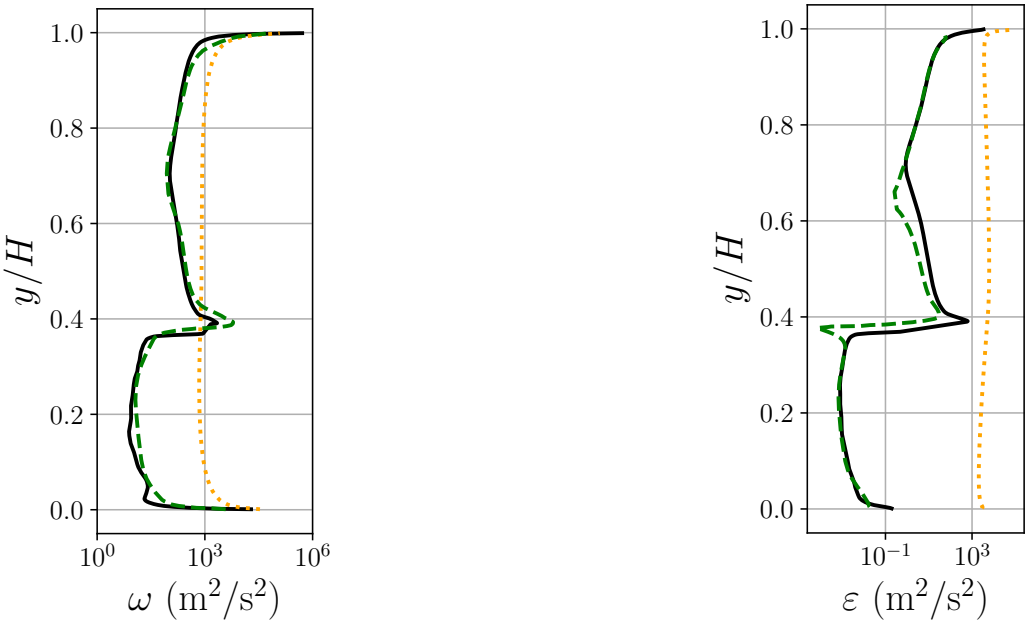


Figure B.3 Specific turbulence dissipation rate (left) and turbulence dissipation rate (right) profiles comparison between the qDNS, the standard $k-\omega$ model (stand.), and the corrected $k-\omega$ model (corr.) predictions in the closed channel configuration and smooth interface regime

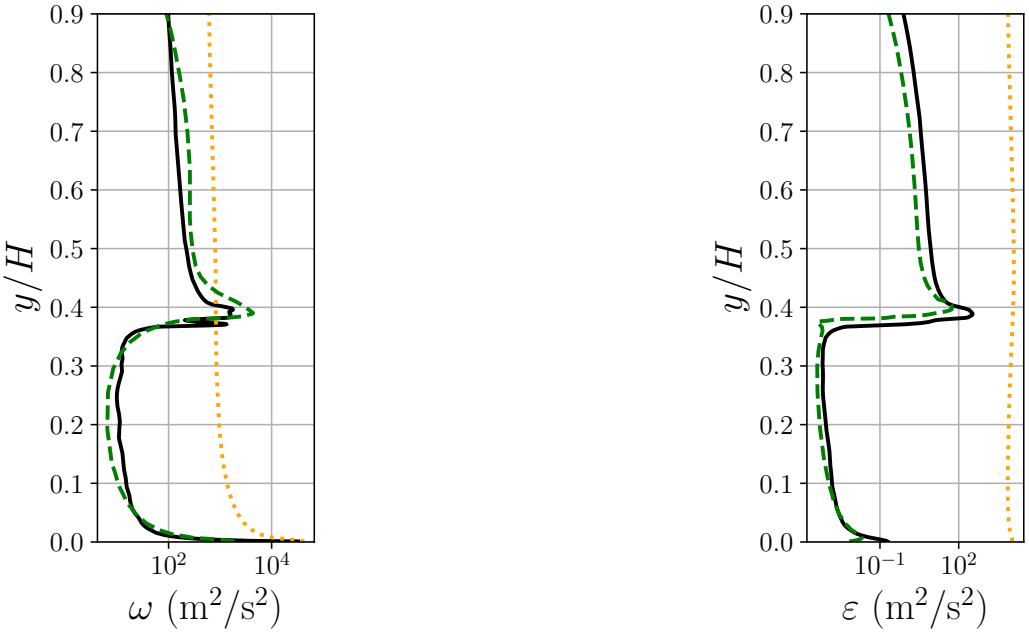


Figure B.4 profiles comparison between the qDNS, the standard $k-\omega$ model (stand.), and the corrected $k-\omega$ model (corr.) predictions in the open channel configuration and wavy interface regime

B.2 Additional figures from chapter 6

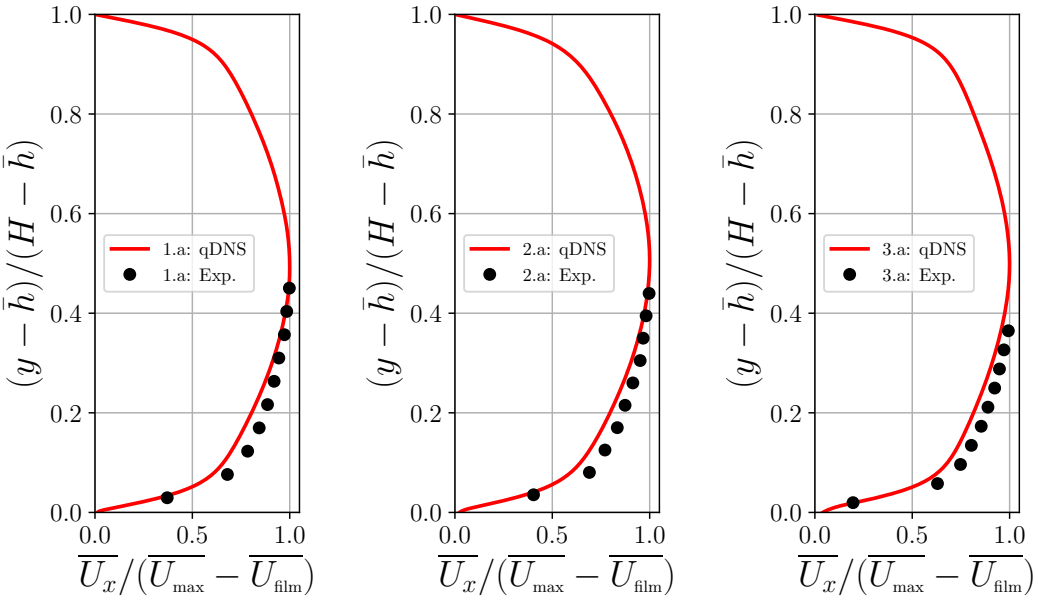


Figure B.5 Mean axial velocity profiles for $U_{b,g} = 3.1$ m/s and using $U_{b,l} = 0.008$ m/s (left), $U_{b,l} = 0.019$ m/s (centre), and $U_{b,l} = 0.031$ m/s (right)

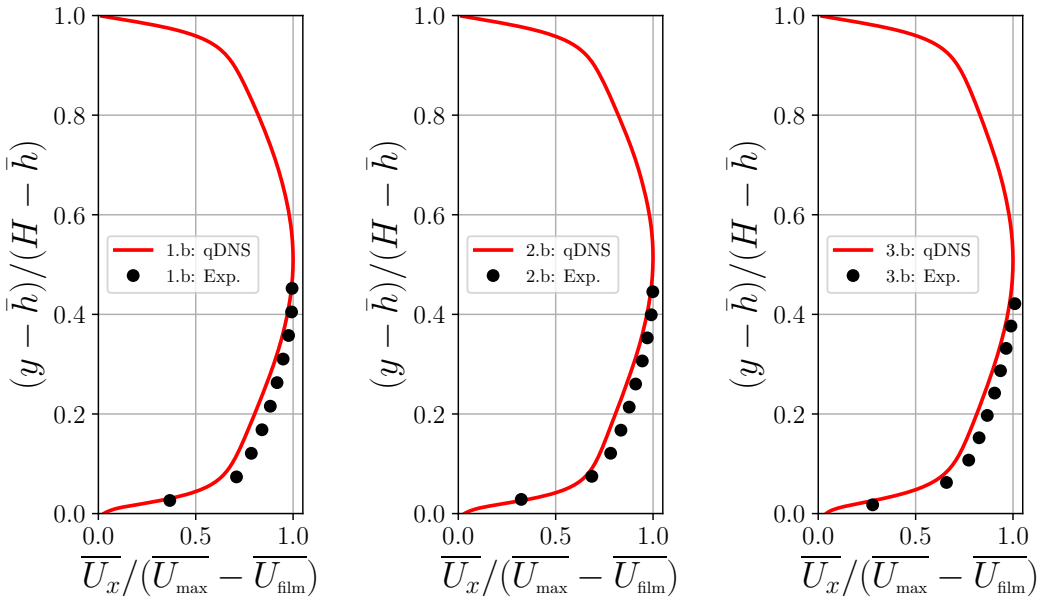


Figure B.6 Mean axial velocity profiles for $U_{b,g} = 3.6$ m/s and using $U_{b,l} = 0.008$ m/s (left), $U_{b,l} = 0.019$ m/s (centre), and $U_{b,l} = 0.031$ m/s (right)

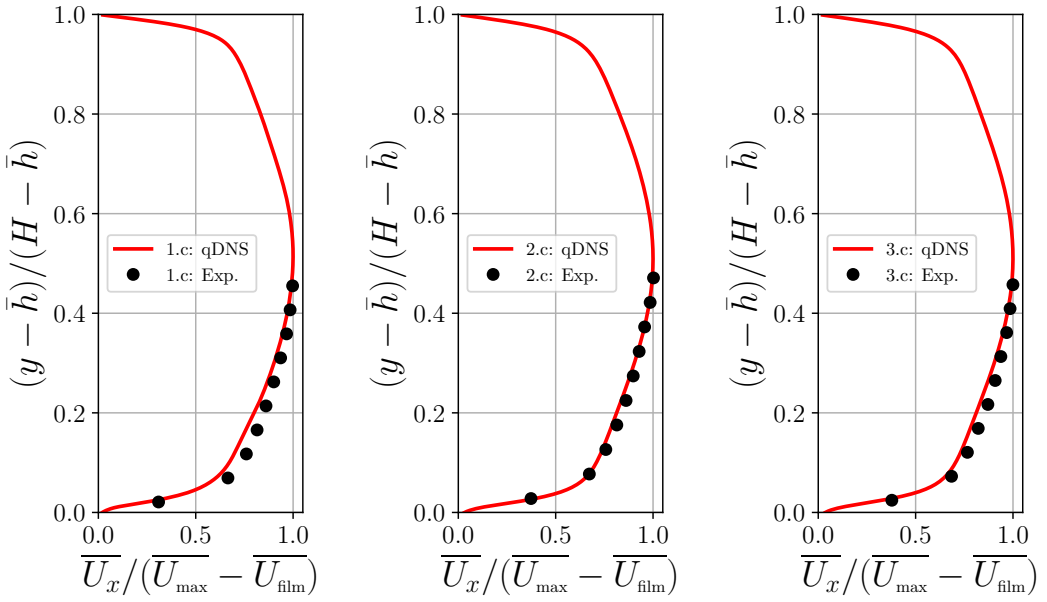


Figure B.7 Mean axial velocity profiles for $U_{b,g} = 4.2$ m/s and using $U_{b,l} = 0.008$ m/s (left), $U_{b,l} = 0.019$ m/s (centre), and $U_{b,l} = 0.031$ m/s (right)

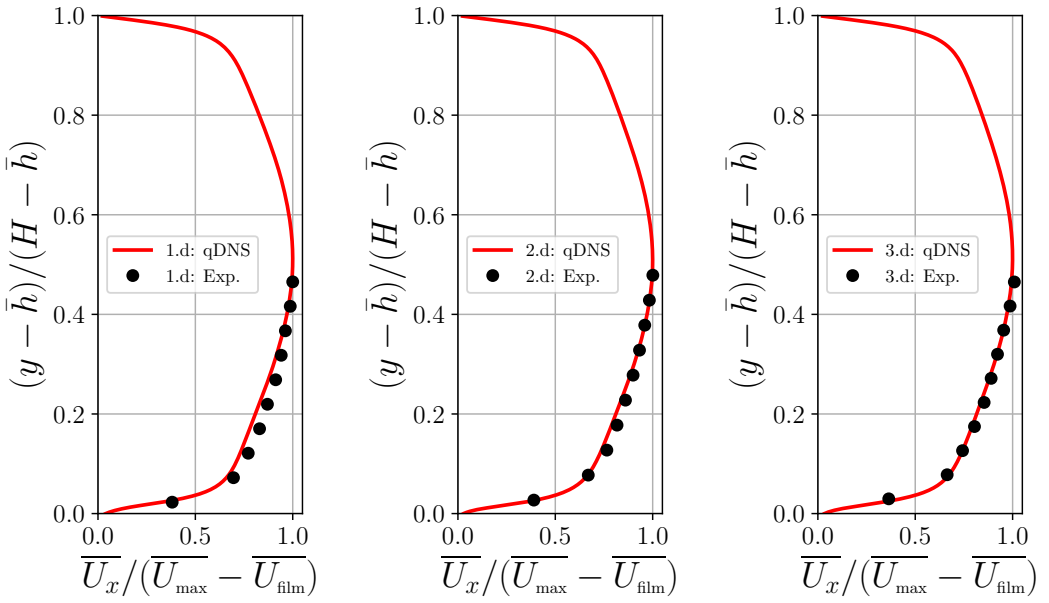


Figure B.8 Mean axial velocity profiles for $U_{b,g} = 4.7$ m/s and using $U_{b,l} = 0.008$ m/s (left), $U_{b,l} = 0.019$ m/s (centre), and $U_{b,l} = 0.031$ m/s (right)

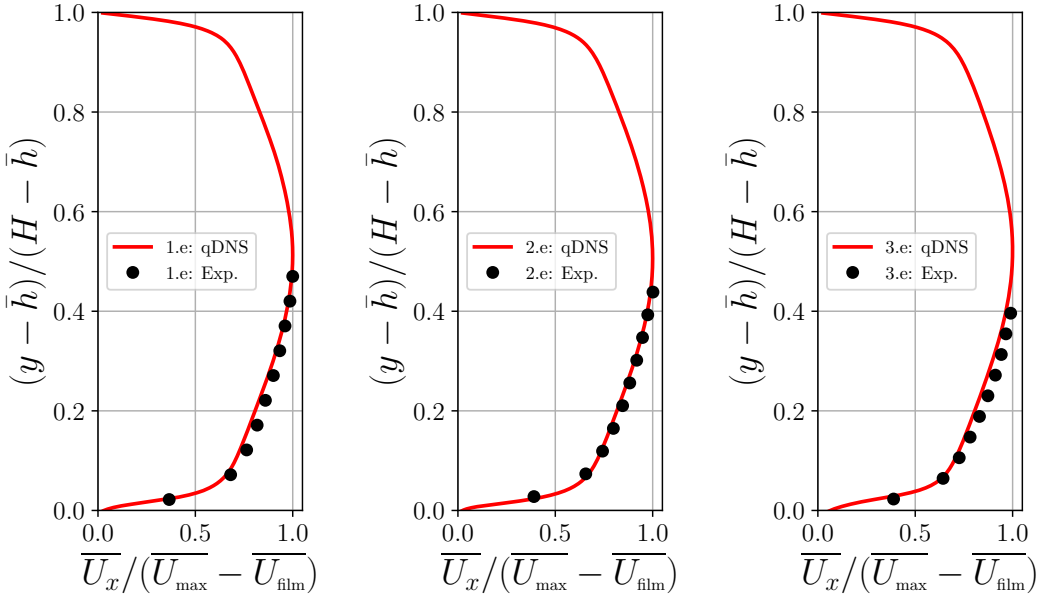


Figure B.9 Mean axial velocity profiles for $U_{b,g} = 5.2$ m/s and using $U_{b,l} = 0.008$ m/s (left), $U_{b,l} = 0.019$ m/s (centre), and $U_{b,l} = 0.031$ m/s (right)

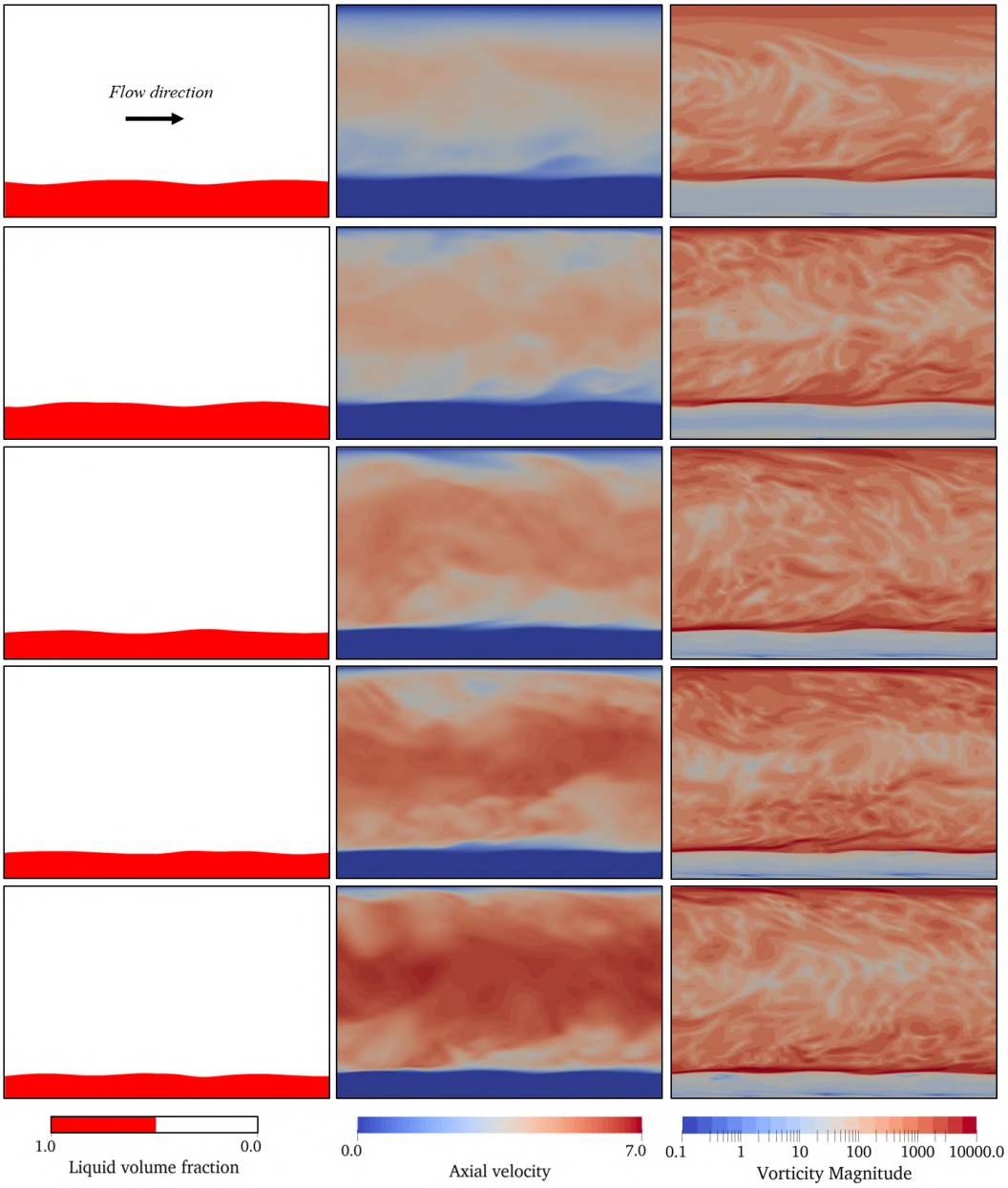


Figure B.10 qDNS simulations performed at the liquid film velocity 0.019 m/s

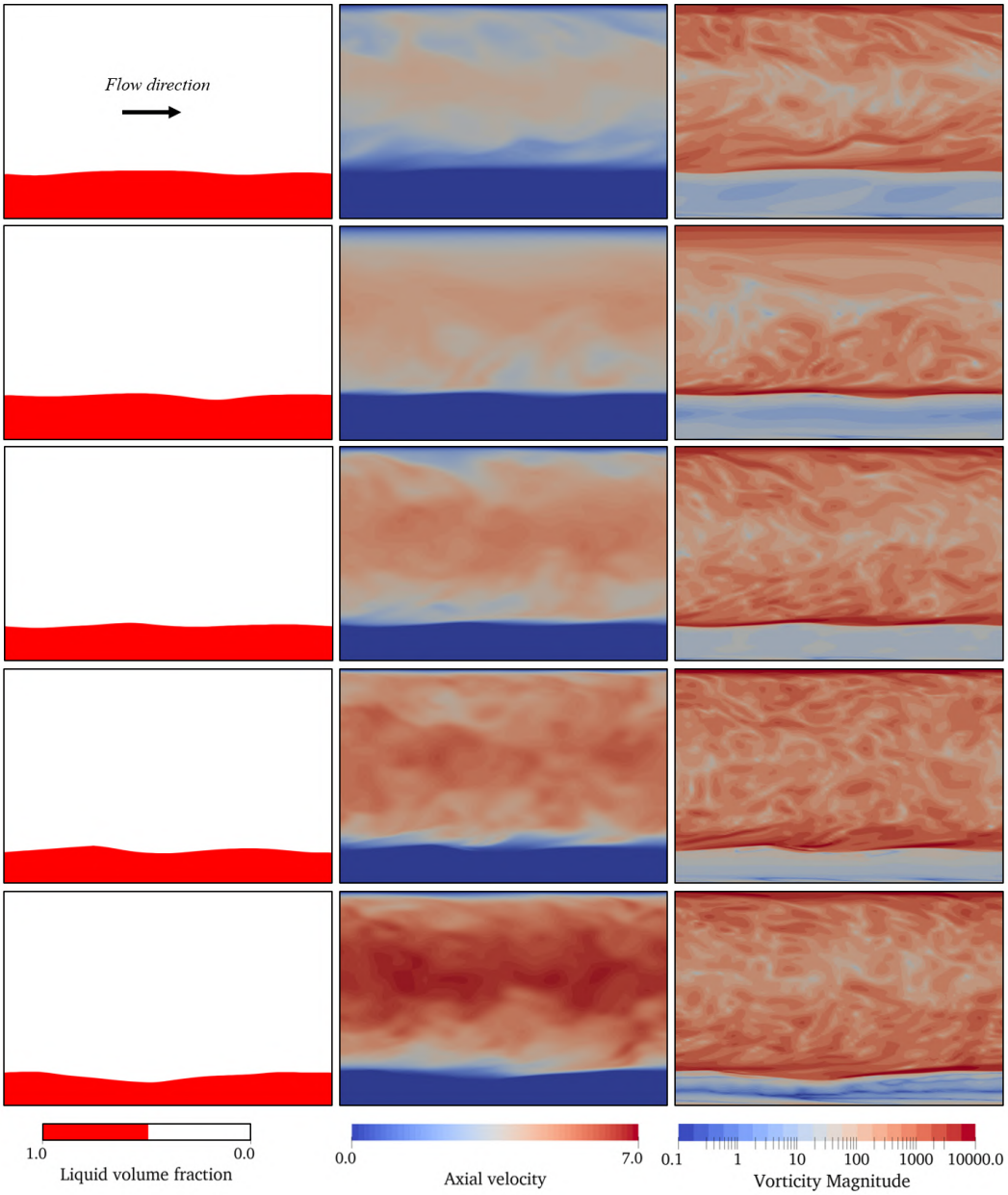


Figure B.11 qDNS simulations performed at the liquid film velocity 0.031 m/s

$\star$