POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Assessment and validation of an in-house finite-volume compressible CFD solver for Large Eddy Simulation

Laurea Magistrale in Aeronautical Engineering - Ingegneria Aeronautica

Author: Umberto Zucchelli

Advisor: Prof. Maurizio Quadrio

Co-advisor: Dr. PhD. Mauro Carnevale, Dr. Ing. Giove De Cosmo

Academic year: 2021-2022

1. Introduction

Turbulence is the natural state of fluid motion. Its study and understanding it's the key to improve our ability to predict the mean flow and consequently to enhance the performance of aerodynamic bodies. Of the three numerical methods for predicting turbulence, i.e. Direct Numerical Simulation (DNS), Large Eddy Simulation (LES) and Reynolds Averaged Navier Stokes (RANS), RANS approach has an inherent peculiarity: it only resolves the mean flow, averaging out the turbulent fluctuations. Therefore, in order to achieve a more comprehensive understanding of turbulent flows, additional information about their instantaneous nature are required. For this reason, high fidelity simulations, as LES and DNS, are necessary. In terms of computational and time requirements, LES occupies an intermediate position between DNS and RANS. Hence, in recent years, also due to the increase in available computational power, LES has been considered more and more attractive for the prediction of turbulent flows.
The goal of this work is to assess the ability of the research code Zephyrus, developed in-house at the University of Bath, to simulate a fundamental turbulent flow, such as the turbulent flow over a bump, through LES. The solver, originally designed for compressible RANS and URANS simulations of turbomachinery flows, has no Sub-Grid Stress (SGS) models and can run Implicit LES. In ILES the numerical schemes ensure that the inviscid energy cascade through the inertial range is accurately captured and the inherent numerical dissipation emulates the effects of the dynamics beyond the grid-scale filter cut-off.
To benchmark the results against another finite-volume code, the compressible OpenFOAM solver *rhoPimpleFOAM* is chosen. A SGS model is enabled in OpenFOAM, and its effect on the solution will be evaluated to verify that the numerical grid is fine enough to capture most of the phenomena related to turbulent kinetic energy in the domain.
The incompressible DNS dataset of the turbulent flow over a bump presented in [1] is taken as reference and reproduced. High-fidelity simulations of compressible turbulent channels and bumps are also common in the literature, but most involve transonic or supersonic flows, or high Reynolds numbers. High Reynolds numbers would have resulted in prohibitively computationally expensive grid resolutions, also due

to the lack of a SGS model. A supersonic test case would increase the complexity of the problem at this preliminary stage of code evaluation, so it was decided to consider incompressible data as a reference. Accordingly, Zephyrus and OpenFOAM compressible simulations setup is tuned to simulate a flow with Mach number low enough to be reasonably below the compressible threshold.

The HPC-Europe3 Transnational Access program provided the computational resources on the UK's HPC clusters Cirrus and Archer2.

2. CFD solvers

2.1. Governing equations

Both solvers, Zephyrus and OpenFOAM, solve the Favre-filtered Navier–Stokes equations for compressible flows. The working fluid is air and it is treated as calorically perfect gas while γ and the Prandtl number Pr are held constant at 1.4 and 0.72 respectively. μ is evaluated by Sutherland's law.

2.2. Numerical schemes

2.2.1. Zephyrus

The CFD solver Zephyrus, known in literature as AU3X [3], is a finite volume density-based CFD solver developed at the University of Bath. The inviscid fluxes are computed by the upwind scheme using the approximated Riemann solver of Roe. Second order spatial discretisation is obtained by extrapolating the values from the cell centre to the interface via the MUSCL scheme with the van Albada limiter. The viscous fluxes at the interface are computed by using the inverse of the distance weighting from the ones evaluated at the cell centres on both sides of the interface while the source terms are evaluated at the cell centres and are assumed to be piecewise constant in the cell. Cell-averaged flow gradient is computed at the cell centre using the weighted least square procedure.

The second order accurate backward Euler numerical scheme is applied in time discretization.

2.2.2. OpenFOAM

The numerical schemes employed in OpenFOAM, are second-order accurate in space and time: the *Gauss linear* divergence scheme has been used for all the terms in the equations except for pressure, discretized by Linear-Upwind Stabilised Transport ($LUST$). The *backward* scheme is used for the temporal term. Regarding the solution procedure, the $GAMG$ is employed for the pressure while *smoothSolver* for all the other quantities. The solver employed is *rhopimpleFoam*, a pressure-based solver which uses the PIMPLE algorithm.

3. Reference case

The reference case is the incompressible DNS reported in [1]. The computational domain, shown

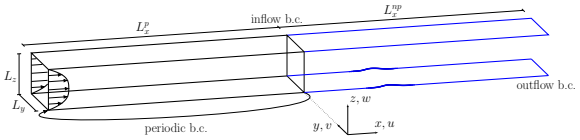


Figure 1: Sketch of the computational domain used in [1].

in Fig. 1, consists of two adjacent sections, aligned along the streamwise direction: an upstream channel with cyclic boundary conditions in the streamwise and spanwise directions, which feeds the downstream bump domain. A constant flow rate (CFR) is imposed in the channel. The bump domain has cyclic conditions in the spanwise direction. It takes as inflow the outflow from the upstream channel domain, while convective conditions are used at the outlet. The dimensions of the whole computational domain are $(L_x^p + L_x^{np}, L_y, L_z) = (4\pi\delta + 12\delta, \pi\delta, 2\delta)$ in the streamwise, spanwise and wall-normal directions respectively, with δ representing the channel half-height. The bump geometry which produces an attached flow is chosen. Simulations are performed at a bulk Reynolds number $Re_b = U_b\delta/\nu = 3173$ corresponding to a friction Reynolds number $Re_\tau = u_\tau\delta/\nu = 200$ for the channel, where the velocity scale is the bulk velocity U_b in the former case and the friction velocity $u_\tau = \sqrt{\tau_w/\rho}$.

Since the reference DNS data are incompressible, yet two compressible solvers are being utilized, the domain setup is adapted to simulate a flow with a Mach number $Ma = 0.14$, low enough to stay reasonably below the compressible threshold, with the compressibility effects being less than 1%.

4. Numerical simulations

4.1. Channel

In the reference case [1], the bump domain is directly fed by the outflow of a turbulent channel flow at $Re_b = 3173$ and $Re_\tau = 200$. In Zephyrus, since an algorithm to impose constant flow rate along with cyclic boundary conditions is not yet implemented, it was not possible to simulate the turbulent channel flow and couple the domains. Therefore, OpenFOAM is employed to perform the preliminary simulation of the turbulent channel flow, necessary to obtain the instantaneous fields needed in inflow to the bump domain. Through the *sampling* function, the instantaneous fields at the channel outlet are saved to file each timestep.

4.1.1. Grid generation

LES simulations compute the solution of the filtered equations, resolving large eddies, while eddies below the filter size are modeled. The separation wavenumber between computed and modelled eddies is set by the filter width, linked directly to the mesh resolution. Therefore, the grid plays a fundamental role in obtaining accurate results.
The channel domain $(L_x, L_y, L_z) = (12\delta, \pi\delta, 2\delta)$ is discretized with $(n_x, n_y, n_z) = (207, 110, 186)$ points in the streamwise, spanwise and wall-normal direction respectively. The computational grid is created with the open-source software GMSH [2].
The mesh spacing in spanwise and streamwise direction is uniform and corresponds to $\Delta x^+ = 12$ and $\Delta y^+ = 6$ respectively. In the wall-normal direction a non-uniform distribution according to the *bump* algorithm is implemented This a GMSH algorithm that locally increases the point density near the wall, keeping the discretization coarse close to the centerline. Thus it is possible to obtain a sufficiently fine grid close to the walls, where the main phenomena related to viscous dissipation take place, and slightly coarser near the centerline, saving computational resources. The cells at the wall correspond to $\Delta z^+ = 0.1$.

4.1.2. Boundary conditions

Cyclic boundary conditions are enforced in both the spanwise and streamwise directions. A Constant flow rate (CFR) is enforced through the OpenFOAM option *meanVelocityForce* applied to the entire domain, tuned with a bulk velocity matching a $Re_b = 3173$. On the lower and upper walls, no-slip and non penetration boundary conditions are imposed. A *zeroGradient* boundary condition is enforced at the walls for the temperature and pressure fields.

4.2. Bump

Simulations of turbulent flow over the bump are performed with both solvers on the same computational grid, with similar boundary conditions, and the same time step is employed.

4.2.1. Grid generation

The same spatial discretization implemented in the channel domain, is also employed for the bump: the domain $(L_x, L_y, L_z) = (12\delta, \pi\delta, 2\delta)$ is discretized with $(n_x, n_y, n_z) = (207, 110, 186)$ nodes. Consequently, the cell sizes in inner units are $\Delta x^+ = 12$, $\Delta y^+ = 6$ and $\Delta z^+ = 0.1$ at the inlet and $\Delta x^+ = 20$, $\Delta y^+ = 10$ and $\Delta z^+ = 0.2$ at the bump tip, according to the local u_τ.

4.2.2. Boundary conditions

The spanwise direction is made homogeneous by applying cyclic conditions, whereas no-slip and non-penetration conditions are applied to the lower and upper wall. The channel outlet instantaneous fields are set as input to the bump domain. This corresponds to a constant flow rate condition, being the mass flow rate fixed at the channel inlet. The fields are set using *timeVaryingMappedFixedValue* in OpenFOAM and *TuInl* (Turbulent Inlet) in Zephyrus. In a compressible simulation the static pressure is needed at the outlet. Therefore, the static pressure in outflow in OpenFOAM is set with the boundary condition *fixedValue*, for the other variables the condition *zeroGradient* is set. In Zephyrus, the boundary condition *FreeExit* is employed. It imposes the static pressure in outflow and extrapolates from the inside.

4.3. Simulations settings

The timestep is set to $\Delta t = 1.5 \times 10^{-7}$, resulting in an average $CFL = 0.1$ in the channel domain and to an average $CFL = 0.4$ in the bump domain.
In OpenFOAM the subgrid stress are modelled with the *kEquation* model.

5. Results

5.1. Channel

The turbulent channel flow simulation in OpenFOAM is let to develop for 120 $\bar{t}$ in order to obtain a statistically steady state. Here $\bar{t}$ is defined as:

$$\bar{t} = \frac{\delta}{U_b} \tag{1}$$

Subsequently, about 600 $\bar{t}$ have been used for accumulating the istantaneous fields (roughly 80000 timesteps).
The values of the skin friction coefficient C_f and the ratio of the mean centerline velocity U_c to the mean bulk velocity U_b, given in Table 1, are in good agreement with the values obtained by Banchetti *et al.* [1], Re_b and Re_τ correspond to the reference values.

	OpenFOAM	DNS
C_f	7.826×10^{-3}	7.833×10^{-3}
U_c/U_b	1.164	1.168

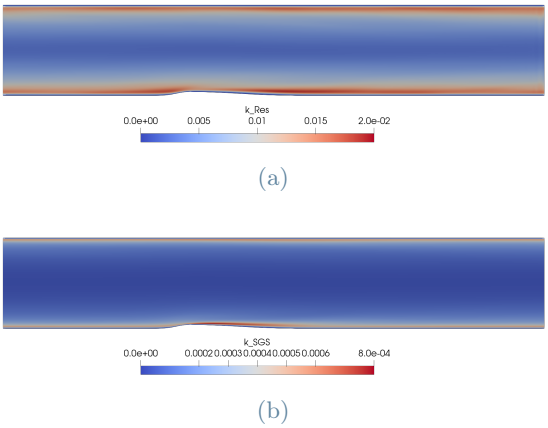
Table 1: Comparison of C_f and U_c/U_b values between OpenFOAM simulation and reference incompressible DNS data.

5.2. Bump

A major difference between the simulations performed with the two solvers is the presence of the SGS model in OpenFOAM simulations. For this reason, before starting to compare the results, the contribution of the *kEquation* SGS model employed in OpenFOAM is analyzed. This preliminary analysis allows us to verify that the results obtained in Zephyrus are not adversely affected by the absence of the model.

5.2.1. SGS model contribution

To study the effects of the sub-grid stress model in detail, some *function Objects* are created in OpenFOAM to compute the following fields: resolved turbulent kinetic energy, SGS dissipation, viscous dissipation, turbulent kinetic energy production.



(a)

(b)

Figure 2: Plot of the resolved turbulent kinetic energy (a) and modeled (b).

The resolved turbulent kinetic energy, in agreement with [1], shows in Figure 2 two high-value regions: one right upstream the bump and one, more intense, towards the end of the bump. The modeled turbulent kinetic energy has instead its maximum intensity just after the tip of the bump, however, with much lower magnitude than the resolved one. Regardless, the contribution of the turbulent kinetic energy provided by the model to the total kinetic energy is around 2% over the whole domin. Similar results are also obtained when analyzing the contribution of the dissipation due to the modeled turbulent viscosity ν_t to the total dissipation due to physical viscosity ν and turbulent viscosity ν_t: the modeled component has much lower magnitude than the resolved one, and it is mainly found in the area just after the bump tip. The results show that the numerical grid is fine enough to capture the main phenomena related to turbulent kinetic energy, and therefore it is legitimate to use the same grid on Zephyrus, which is not provided with SGS model.

5.2.2. Solvers comparison

The time-averaged results of the velocity profiles, Figure 3 and of the six components of the Reynolds stress tensor, Figure 4 highlight that the stress tensor component $\langle w'w' \rangle$ in Zephyrus is already significantly reduced at the first cell center downstream the (recall that w is the ve-

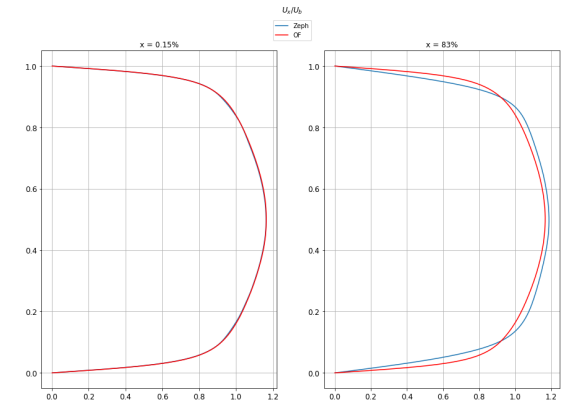


Figure 3: Plot of the bulk mean velocity profile at increasing x-coordinates.

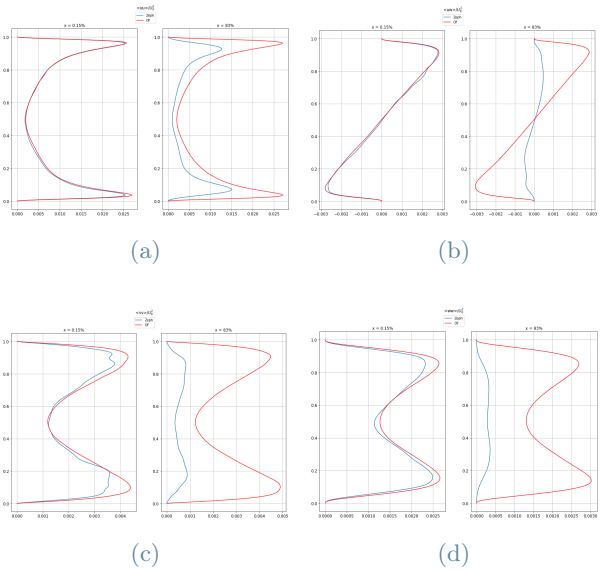


Figure 4: Plot of the $\langle u'u' \rangle$ (a), $\langle u'w' \rangle$ (b), $\langle v'v' \rangle$ (c), $\langle w'w' \rangle$ (d) component of the Reynolds stresses at increasing x-coordinate.

locity normal to the wall for how the axes of the reference system are oriented). This phenomenon, related to the lower intensity of the fluctuations, also occurs in all other components of the Reynolds stresses and becomes more pronounced along the streamwise direction of the bump domain. The decrease in intensity of the velocity fluctuations also affects the mean velocity profile. Indeed, at the outlet, the mean velocity profile obtained with Zephyrus deviates significantly from the one obtained with OpenFOAM, exhibiting a lower derivative at the wall and a higher mean velocity in the area of the domain near the centerline.

The averaged skin friction coefficient for the

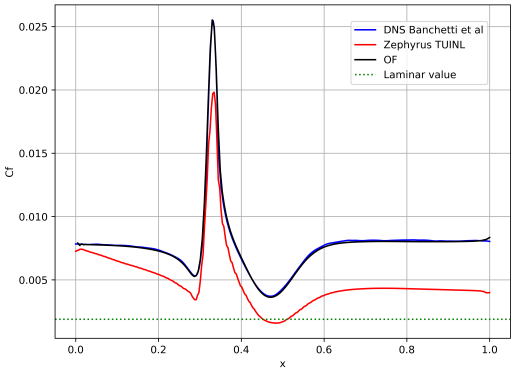


Figure 5: Comparison plot of the skin friction coefficient C_f.

reference case and the two solvers is compared Fig. 5. At the inlet, the C_f is similar between the two codes, with OpenFOAM exhibiting very good agreement with the incompressible DNS data. The C_f obtained in Zephyrus drops in the first few cells, consistently with the decrease in the intensity of the velocity fluctuations, to a value roughly half of the reference value. The Re_τ drops from the value of $Re_\tau = 200$ at the inlet to $Re_\tau = 150$ after the bump, which is retained until the domain outlet.

Fig. 6 shows the mean C_p averaged in time

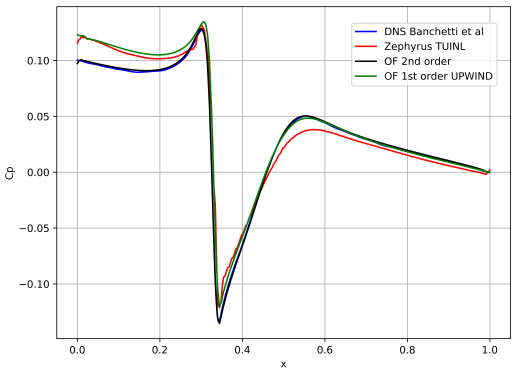


Figure 6: Comparison plot of the pressure coefficient C_p.

and in space over the lower wall. The calculated pressure coefficient C_p is set to 0 at the outlet of the domain. The pressure coefficient has a minimum exactly at the tip of the bump. Away from the bump, the pressure coefficient has a linear trend (i.e. uniform mean pressure

gradient), as expected for a plane channel flow. Again, OpenFOAM has an excellent agreement with DNS data. Zephyrus shows a worse behavior: at the outlet the pressure maintains the value enforced via the *Free Exit* boundary condition, while the inflow pressure read from file is adapted to a higher value, leading to a greater Δp across the domain.
A similar trend is observed in the results averaged for 120 $\bar{t}$ of an OpenFOAM LES simulation with the *kEquation* subgrid model and first-order accurate *UPWIND* numerical schemes in space, backward Euler second-order scheme in time. OpenFOAM results can be justified by the highly diffusive numerical method, which leads to the decrease in turbulent fluctuations and flow transition to a lower Re_τ.
As for Zephyrus, further investigation is needed. The fundamental turbulent flow analyzed is mainly characterized by being a wall-bounded flow and by the presence of a curvature (bump) on the lower wall. It might be useful to separate these two effects to assess the presence in the code of any issue regarding the inner face numerical flux (and thus related to the dissipation of the numerical method) or concerning the treatment of the wall face flux near the bump curvature.

6. Conclusions

In this thesis, the main features of Large Eddy Simulations with and without SGS model are reviewed. The scope is to assess the ability of the compressible CFD solver Zephyrus to simulate a turbulent flow over a bump using ILES and to develop the routines needed to get the task accomplished.
A turbulent channel flow is simulated in OpenFOAM to generate the instantaneous turbulent fields needed as inflow boundary condition for the bump domain. The effect of the SGS model set in OpenFOAM is carefully analyzed to verify that the grid resolution is sufficient to capture the main phenomena related to turbulent kinetic energy k.
The results show that in Zephyrus the turbulent fluctuations are greatly reduced after few cells from the inlet and that the flow tends to re-laminarize toward a turbulent condition corresponding to $Re_\tau = 150$. This diffusion phenomenon of turbulent fluctuations also affects the inlet-outlet pressure gradient. Counter-intuitively, a decrease in the friction coefficient corresponds to an increase of Δp. However, this phenomenon is also displayed in an OpenFOAM simulation with UPWIND first-order accurate numerical schemes in space.
In Zephyrus the cause of the decreased intensity of turbulent fluctuations has been attributed to the high dissipation of the implemented numerical methods, which cause the transition to a lower Re_τ.
Future developments involve investigating further and simulating in Zephyrus a turbulent channel flow with parallel walls (eliminating the effect of curvature) and performing a simulation similar to the one presented by Saad et al in [4]: a study of the decay of homogeneous isotropic turbulence in a 3D cube with cyclic conditions applied in all directions, allowing the dissipation of the numerical method to be evaluated in the absence of walls.

References

[1] J. Banchetti, P. Luchini, and M. Quadrio. Turbulent drag reduction over curved walls. *Journal of Fluid Mechanics*, 896:A10, 2020.

[2] C. Geuzaine and J.F. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities.

[3] I. Hadade, F. Wang, M. Carnevale, and L. di Mare. Some useful optimisations for unstructured computational fluid dynamics codes on multicore and manycore architectures. *Computer Physics Communications*, 235:305–323, 2019.

[4] T. Saad, D. Cline, R. Stoll, and J. C. Sutherland. Scalable Tools for Generating Synthetic Isotropic Turbulence with Arbitrary Spectra. *AIAA Journal*, 55(1):327–331, 2017.

POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Assessment and validation of an in-house finite volume compressible CFD solver for Large Eddy Simulation

Tesi di Laurea Magistrale in
Aeronautical Engineering - Ingegneria Aeronautica

Candidate: **Umberto Zucchelli**

Student ID: 952952
Advisor: Prof. Maurizio Quadrio
Co-advisors: Dr. PhD. Mauro Carnevale, Dr. Ing. Giove De Cosmo
Academic Year: 2021-22

Abstract

Turbulence is the natural state of fluid motion, is ubiquitous and influences many phenomena in engineering and the natural sciences. Historically, numerical simulations have proved very beneficial in the study of turbulent flows, as they allow the measurement of quantities that cannot be measured experimentally. In this context, the Reynolds-Averaged Navier Stokes (RANS) and Unsteady-RANS simulations approach is unable to provide high-fidelity results, as they are based on limited, albeit complex, models for resolving the mean component of the flow. Therefore, to gain a deeper understanding and gather information on the instantaneous fluctuations characteristic of turbulent flows, high-fidelity instationary simulations are needed, carried out with fine numerical grids, such as Large Eddy Simulation (LES) or Direct Numerical Simulation (DNS).

The finite-volume CFD code Zephyrus, developed at the Turbomachinery Research Center at the University of Bath, was originally designed to simulate compressible turbomachinery flows with RANS and URANS models. The purpose of this thesis is to investigate the ability of Zephyrus to perform compressible LES simulations without a subgrid model (Implicit LES, or ILES), comparing its accuracy with the open-source OpenFOAM code. Turbulent flow over a bump is a widely used case study, as it reproduces typical characteristics of realistic applications, such as favorable and adverse pressure gradients, non-constant friction along the flow direction, and possibly separations.

The results obtained in OpenFOAM have a very good match with DNS data, while Zephyrus shows a tendency to re-laminize. In fact, the turbulent Reynolds decreases from a value $Re_\tau = 200$ at the domain entrance to $Re_\tau = 150$ downstream of the bump. Contextually, the C_f, turbulent fluctuations and components of the Reynolds stress tensor are attenuated compared to the values obtained with OpenFOAM. This behavior was mainly attributed to the excessive dissipation of the numerical scheme used in Zephyrus.

Keywords: Turbulence, LES, ILES, OpenFoam, Turbulent channel, Bump

Abstract in lingua italiana

La turbolenza è lo stato naturale del moto dei fluidi, è onnipresente e influenza molti fenomeni dell'ingegneria e delle scienze naturali. Storicamente, le simulazioni numeriche si sono rivelate molto vantaggiose nello studio dei flussi turbolenti, poiché permettono la misurazione di quantità non rilevabili sperimentalmente. In questo contesto, l'approccio con simulazioni RANS (Reynolds-Averaged Navier Stokes) e Unsteady-RANS non è in grado di fornire risultati ad alta fedeltà, in quanto basate su modelli limitati, per quanto complessi, per la risoluzione della componente media del flusso. Pertanto, per ottenere una comprensione più profonda e raccogliere informazioni sulle fluttuazioni istantanee caratteristiche dei flussi turbolenti, sono necessarie simulazioni instazionarie ad alta fedeltà,,realizzate con griglie numeriche fini, come Large Eddy Simulation (LES) o Direct Numerical Simulation (DNS).

Il codice CFD ai volumi finiti Zephyrus, sviluppato presso il Turbomachinery Research Center dell'Università di Bath, è stato originariamente ideato per simulare flussi comprimibili in turbomacchine con modelli RANS e URANS. Lo scopo di questa tesi è di indagare la capacità di Zephyrus di eseguire simulazioni LES comprimibili senza modello di sottogriglia (Implicit LES, o ILES), confrontandone l'accuratezza con il codice oepn-source OpenFOAM. Il flusso turbolento sopra un bump è un caso studio ampiamente utilizzato, in quanto riproduce caratteristiche tipiche di applicazioni realistiche, come gradienti di pressione favorevoli ed avversi, un attrito non costante lungo la direzione del flusso ed eventualmente separazioni.

I risultati ottenuti in OpenFOAM hanno un'ottima corrispondenza con i dati DNS, mentre Zephyrus mostra una tendenza a ri-laminizzare. Infatti, il Reynolds turbolento si riduce da un valore $Re_\tau = 200$ all'ingresso del dominio a $Re_\tau = 150$ a valle del bump. Contestualmente, il C_f, le fluttuazioni turbolente e le componenti del tensore degli stress di Reynolds risultano attenuati rispetto ai valori ottenuti con OpenFOAM. Questo comportamento è stato prevalentemente attribuito alla eccessiva dissipazione dello schema numerico utilizzato in Zephyrus.

Parole chiave: Turbolenza, LES, ILES, OpenFoam, canale turbolento, Bump

Contents

Introduction

Turbulence is the natural state of fluid motion, it is ubiquitous, and affects many phenomena in engineering and the natural sciences. The aerodynamic or hydrodynamic performance of aircraft, cars and ships depends critically on the turbulent flow around the bodies. Hence the dynamics of velocity fluctuations in turbulent flows and their mechanisms of formation, dissipation and transport have aroused interest from the fluid dynamic community for years, as their study and understanding would improve our ability to predict the mean flow and consequently to possibly improve the performance of aerodynamic bodies.

In this context, Computational Fluid Dynamics (CFD) has proven useful to provide a better understanding of such turbulent flow, allowing for the measurement of key quantities not accessible with standard measurement techniques. Of the three numerical methods for predicting turbulence, i.e. Reynolds Averaged Navier Stokes (RANS), Large Eddy Simulation (LES) and Direct Numerical Simulation (DNS), RANS methods are most commonly used in the industrial field. By definition, the RANS approach only resolves the mean flow, averaging out the turbulent fluctuations. Due to this simplification, RANS and URANS computations are the lightest ones in terms of computational time. Nevertheless, in many cases, in order to achieve a more comprehensive understanding, additional information about the instantaneous nature of turbulent flows is required. This drawback is only partially overcome by URANS, whereas high fidelity simulations, as LES and DNS, are necessary. Large Eddy Simulation aims at resolving most of the turbulent spectrum, limiting modelling to the smallest scales. In Direct Numerical Simulation the whole spectrum is resolved.

In the past, the use of these techniques was limited by the available computational power. In fact, the required computation time heavily depends on the number of grid points selected to represent the computational domain under consideration, where LES and DNS require fine grids. The increasing computational power has made the use of these techniques more and more attractive and feasible. In terms of computational requirements and time, LES occupies an intermediate position between DNS and RANS. For this reason, in the last years, LES has been considered more and more interesting for the modelling of

turbulent flows. Meshing in LES is extremely important, as the grid size is linked to the filter that separates the resolved scales from the modelled ones. Various meshing guidelines are available in literature, based on both experience and theory, see for example the estimation proposed by H. Choi *et al* [1]. Such guidelines are aimed at reducing the computational effort, while keeping an acceptable level of accuracy of involved phenomena.
The turbulent flow over a bump is a widely considered case of study, as it reproduces features typical of real-world applications, as well as favorable and adverse pressure gradients, a non-constant friction along the streamwise direction and, eventually, separation of the flow. These are, indeed, characteristics of many aerodynamic flows such as flow past airfoils, sails and gas turbine blades. The turbulent flow over a bump has been extensively studied experimentally and numerically. Over the years, it has been used as a test case for the development, validation and accuracy estimation of both RANS models and LES subgrid stress models; see for example [2–4]. The domain with the bump described by Banchetti *et al.* in [5] is considered as the reference case.
The CFD solvers selected for the study are OpenFOAM and Zephyrus, known in literature as AU3X [6]. Zephyrus is a CFD solver developed in-house at the Turbomachinery Research Center (University of Bath). The solver was originally developed for compressible RANS and URANS simulations of turbomachinery flows. Recently, interest has arisen in high-fidelity simulations, as they allow more accurate prediction of flow features such as heat transfer or flow separation. Therefore, the code is tested in LES modelling with the goal of developing the necessary routines and performing an assessment of the accuracy in the prediction of turbulent flows.
The HPC-Europe3 Transnational Access program provided the computational resources on the UK's HPC clusters Cirrus and Archer2.

Project rationale

RANS simulations are unable to capture the instantaneous properties of a turbulent flow, furthermore, turbulent models can lead to incorrect results when used in flows with separations, as is often the case in turbomachinery. For this reason, the state of the art in turbomachinery research is moving toward LES simulations, which can provide more accurate results.
The scope of this work is to assess the ability of the research code developed in-house at the University of Bath, Zephyrus, originally designed to solve compressible RANS and URANS simulations of turbomachinery flows, to simulate a fundamental turbulent flow, such as the turbulent flow over a bump, through LES. The solver has no SGS models, implying that the simulations are Implicit LES. This approach was introduced in [7] (see

also [8]), under the name Monotone Integrated Large Eddy Simulation (MILES), but has recently come to encompass a wider range of schemes under the name ILES. These are numerical methods that capture the inertial and energy-containing ranges of turbulent flows, relying on their inherent dissipation to act as a subgrid model.
To compare the results with another finite-volume code, the OpenFOAM solver *rhoPimpleFOAM* is chosen. A SGS model is considered in OpenFOAM, and its contribution to the solution is evaluated to verify that the numerical grid is fine enough to capture the main phenomena related to turbulent kinetic energy even in the case of Zephyrus, which does not have a SGS model.
Incompressible DNS data presented in [5] are taken as reference. This choice might seem not entirely appropriate for the assessment of a compressible solver. However, high-fidelity simulations of compressible turbulent channels and bumps in the literature often involve transonic or supersonic flows, or flows at high Reynolds number, as in [9, 10]. High Reynolds numbers would have resulted in prohibitively computationally expensive grid resolutions, also due to the lack of a SGS model, while it is intended to avoid simulating a supersonic test case so as to not increase the complexity of the problem. Therefore, the domain setup is adapted to simulate a flow with Mach number low enough to stay reasonably below the compressible threshold.

1 | Large Eddy Simulation

1.1. Formulation of LES

Large Eddy Simulation (LES) stands in an intermediate position between DNS and RANS. In turbulence the small eddies can be considered nearly isotropic while the large eddies are more anisotropic and influenced by the geometry of the considered domain as well as by the boundary conditions. The LES approach is to compute exactly the large energy-carrying eddies, while the effect of the small scales of turbulence is modelled. The separation between exactly-computed and modelled eddies is done by defining a spatial filtering function. The filter width is then of particular interest in this kind of simulations: as the width decreases, LES becomes closer to DNS as it improves its accuracy as well as its computational cost; the opposite occurs with a "larger" filter width. In most finite volume implementations the filter width is identical to the grid width (implicit filter width), then a correct meshing for LES is essential to obtain high fidelity results in describing turbulence with an acceptable computational cost [11].
The general filtered function, integrated in space and not in time, can be written as follows:

$$\bar{\phi}(\mathbf{x}, t) = \iiint_{-\infty}^{\infty} G(\mathbf{x}, \mathbf{x}', \Delta)\, \phi(\mathbf{x}, t)\, dx_1' dx_2' dx_3' \tag{1.1}$$

where:

- $G(\mathbf{x}, \mathbf{x}', \Delta)$ is the filter function;
- $\phi(\mathbf{x}, t)$ is the original function;
- $\bar{\phi}(\mathbf{x}, t)$ is the filtered function;
- Δ is the filter width i.e. the smallest wavelength retained by the filter [12].

In the finite volume implementations of LES the filter function is usually the so called "box filter"; this is defined as:

$$G(\mathbf{x}, \mathbf{x}', \Delta) = \frac{1}{\Delta^3} \tag{1.2}$$

for $x - \frac{\Delta}{2} \leq x' \leq x + \frac{\Delta}{2}$.
The filtered equations, describing the motion of large eddies, are computed directly, while the motion of smaller eddies is modelled through a subgrid-scale stress term (SGS) or residual stress tensor τ_{ij}^r equal to:

$$\tau_{ij}^r = \overline{u_i u_j} - \overline{u}_j \overline{u}_i \tag{1.3}$$

The LES modelling approach is to take information from the smallest resolved scales and then to use them as boundary conditions for the modelled scales. The different LES methods are characterized by the modelling procedures of the residual stress tensor, that links exactly-computed with modelled eddies. This relation can be written as follows:

$$\tau_{ij}^r - \frac{\delta_{ij}}{3}\tau_{kk}^r = -2\nu\overline{S}_{ij} \tag{1.4}$$

where $\overline{S}_{ij}$ is the filtered (or large-scale) rate of strain tensor defined as:

$$\overline{S}_{ij} = 2\left(\overline{S}_{ij}\overline{S}_{ij}\right) \tag{1.5}$$

1.1.1. Subgrid-Scale Modelling

LES models are usually called subgrid-scale (SGS) models to emphasize the key role of the residual stress tensor in this kind of modelled turbulence. In this section some SGS models, i.e. LES no-model, Smagorinsky model (1963) and the one-equation eddy-viscosity SGS model by A. Yoshizawa and K. Horiuti (1985) [13], will be illustrated. In this thesis only the LES no-model and the one-equation eddy-viscosity model have been applied (the latter also known as the *k-equation* model).

LES Smagorinsky Model

The Smagorinsky model (1963) assumes that, because the smallest eddies are almost isotropic, the Boussinesq hypothesis is able to provide a good description of the effects of unresolved eddies on the resolved flow. This consideration allows to state that the local SGS stresses τ_{ij}^r are proportional to the local rate of strain of the resolved eddies $\overline{S}_{ij}$.
Thus, the relation between filtered and modelled quantities can be written as follows:

$$\tau_{ij}^r = -2\nu_{SGS}\overline{S}_{ij} + \frac{1}{3}\tau_{ij}^r\delta_{ij} = -\nu_{SGS}\left(\frac{\partial\overline{u}_i}{\partial x_j} + \frac{\partial\overline{u}_j}{\partial x_i}\right) + \frac{1}{3}\tau_{ij}^r\delta_{ij} \tag{1.6}$$

where ν_{SGS} is the subgrid kinematic viscosity. For this model, the length scale is chosen as a grid-characteristic length scale or filter-width length scale, $\Delta = (h_x h_y h_z)^{1/3}$. The time scale is defined from the norm of the strain rate tensor $|\overline{S}|$. This finally gives:

$$\nu_{SGS} = (C_S\Delta)^2 |\overline{S}| = (C_S\Delta)^2 \sqrt{\overline{S}_{ij}\overline{S}_{ij}} \tag{1.7}$$

where C_S is the Smagorinsky constant. As C_S is real and positive, this model is purely dissipative and does not account for backscatter (inverse energy cascade). In several studies (e.g. Rogallo and Moin (1984) [14]) C_S was shown not to be a constant, depending on the Reynolds number and the case geometry. For strongly not isotropic geometries (such as channel flows), the assumption of proportionality decays and the constant has to be adapted.

k Equation Model

As in the case of the Smagorinsky model, the one equation eddy viscosity SGS model uses the eddy viscosity approximation, so the subgrid scale stress tensor τ_{ij}^r is approximated as before. The subgrid scale eddy viscosity ν_{SGS} is computed using k_{SGS}:

$$\nu_{SGS} = C_k \sqrt{k_{SGS}} \Delta \tag{1.8}$$

where $k_{SGS} = 1/2\tau_{kk}^r$ and C_k is a model constant whose default value is 0.094.
The procedure so far is the same as in the Smagorinsky model but these models are different in terms of how they compute k_{SGS}. The Smagorinsky model assumes the local equilibrium to compute k_{SGS} but the one equation eddy viscosity model solves a transport equation for k_{SGS} [13]. The main reason to develop the one-equation SGS models is to overcome the deficiency of local balance assumption between the SGS energy production and dissipation adopted in algebraic eddy viscosity models. A transportation equation is derived to account for the historic effect of k_{SGS} due to production, dissipation and diffusion:

$$\frac{\partial \rho k_{SGS}}{\partial t} + \frac{\partial(\rho \overline{u}_j k_{SGS})}{\partial x_j} - \frac{\partial}{\partial x_j}\left[\rho\left(\nu + \nu_{SGS}\frac{\partial k_{SGS}}{\partial x_j}\right)\right] = -\rho\tau_{ij}^r : \overline{S}_{ij} - C_\epsilon \frac{\rho k_{SGS}^{3/2}}{\Delta} \tag{1.9}$$

where C_ϵ is another model constant. The terms in 1.9 are from left to right, the time derivative term, convective term, diffusion term, production term and dissipation term. In the case of the Smagorinsky SGS model, only the production and dissipation terms are taken into account with the assumption of local equilibrium. The production term in 1.9

can be rearranged to yield the expression used in the source code as follows:

$$\frac{\partial \rho k_{SGS}}{\partial t} + \frac{\partial (\rho \overline{u}_j k_{SGS})}{\partial x_j} - \frac{\partial}{\partial x_j}\left[\rho\left(\nu + \nu_{SGS}\frac{\partial k_{SGS}}{\partial x_j}\right)\right] = \rho G - \frac{2}{3}\rho k_{SGS}\frac{\partial \overline{u}_k}{\partial x_k} - C_\epsilon \frac{\rho k_{SGS}^{3/2}}{\Delta} + S_k \tag{1.10}$$

Implicit LES

In ILES (Implicit LES) or LES no-model the modelling of the small scales is avoided: no subgrid-scale stress tensor is used. The main difference between LES no-model and DNS lies in the numerical method: the numerical method for ILES is usually a Finite Difference Method (FDM) or Finite Volume Method (FVM) second order accurate in space and time, while in DNS spectral methods and fourth order or higher discretization methods are usually implemented [15].

The main advantage in using LES no-model lies in its applicability to more complicate geometries and higher Reynolds number turbulent flows. On the other hand, one must be aware that LES no-model is not a DNS: some information, especially regarding the small scales of turbulence, is definitely lost.

In ILES the numerical schemes are used such that the inviscid energy cascade through the inertial range is accurately captured and the inherent numerical dissipation emulates the effects of the dynamics beyond the grid-scale cut-off. For this reason the numerical grid acquires a key role in modelling the main processes responsible for the production and dissipation of turbulent kinetic energy

2 | CFD solvers

This chapter presents the main features of Zephyrus and OpenFoam [16] CFD solvers in the context of the compressible LES simulations that are object of this thesis.

2.1. Zephyrus

2.1.1. Governing Equation

In compressible flows, it is convenient to use Favre-filtering to avoid the introduction of subfilter-scale term in the equation of conservation of mass. A Favre-filtered variable is defined as:

$$\widetilde{f} = \frac{\overline{\rho f}}{\overline{\rho}} \tag{2.1}$$

The Favre-filtered Navier–Stokes equations for compressible flows in the differential form read:

$$\begin{cases} \dfrac{\partial \overline{\rho}}{\partial t} + \dfrac{\partial \left(\overline{\rho}\widetilde{u}_i\right)}{\partial x_j} = 0, & \text{(2.2a)} \\ \dfrac{\partial \left(\overline{\rho}\widetilde{u}_i\right)}{\partial t} + \dfrac{\partial \left(\overline{\rho}\widetilde{u}_i\widetilde{u}_j\right)}{\partial x_j} = -\dfrac{\partial \overline{p}}{\partial x_i} + \dfrac{\partial}{\partial x_j}\left(\widetilde{\tau}_{ij} + \tau^r_{ij}\right), & \text{(2.2b)} \\ \dfrac{\partial \left(\overline{\rho}\widetilde{e}\right)}{\partial t} + \dfrac{\partial \left(\overline{\rho}\widetilde{u}_j\widetilde{h}\right)}{\partial x_j} = -\dfrac{\partial}{\partial x_j}\left(\kappa\dfrac{\partial \widetilde{T}}{\partial x_j} + \widetilde{u}_i\left(\widetilde{\tau}_{ij} + \tau^r_{ij}\right)\right), & \text{(2.2c)} \end{cases}$$

where:

$$\widetilde{\tau}_{ij} = 2\mu\left(\widetilde{S}_{ij} - \frac{1}{3}\frac{\partial \widetilde{u}_k}{\partial x_k}\delta_{ij}\right) \tag{2.3}$$

$$\overline{p} = (\gamma - 1)\,\overline{\rho}\left(\widetilde{e} - \frac{1}{2}\widetilde{u}_j\widetilde{u}_j\right) \tag{2.4}$$

$$\kappa\frac{\partial T}{\partial x_j} = \frac{\gamma}{\gamma - 1}\frac{\mu}{Pr}\frac{\partial}{\partial x_j}\left(\frac{\overline{p}}{\overline{rho}}\right) \tag{2.5}$$

τ^r_{ij} is equal to 0 since there is no SGS model implemented yet. The working fluid is air and it is treated as calorically perfect gas while γ and the Prandtl number Pr are held constant at 1.4 and 0.72 respectively, alongside with μ which is evaluated by Sutherland's

law and is based on a reference viscosity of $1.7894 \times 10^{-5} \frac{kg}{ms}$ together with a reference temperature of $288.15K$ and Sutherland's constant at $110K$.

2.1.2. Numerical schemes

In Zephyrus the flow variables are stored at the cell centres and the boundary conditions are applied at the ghost cells, the positions of which are generated by mirroring the positions of the cells immediately adjacent to the boundary. The inviscid fluxes are computed by the upwind scheme using the approximated Riemann solver of Roe [17]. Second order spatial discretisation is obtained by extrapolating the values from the cell centre to the interface via the MUSCL scheme [18] with the van Albada limiter [19]. The viscous fluxes at the interface are computed by using the inverse of the distance weighting from the ones evaluated at the cell centres on both sides of the interface while the source terms are evaluated at the cell centres and are assumed to be piecewise constant in the cell. Cell-averaged flow gradient is computed at the cell centre using the weighted least square procedure [20]. The matrix of the weighted least square gradient is evaluated once at pre-processing for static grids.

2.1.3. Time integration

Steady solution

After inviscid, viscous fluxes and source terms are computed for each cell, the coupled system in 2.2 can be described as the following:

$$\Omega_i \frac{dU_i}{dt} = -\sum_{j=1}^{N} R_i \left(U_j\right) \tag{2.6}$$

where U_i are the conservative variables of cell i, namely $(\overline{\rho}, \overline{\rho}\widetilde{u_i}, \overline{\rho}\widetilde{e})^T$, Ω_i the cell volume, U_j the conservative variables in the neighbouring cells of U_i, N the number of neighbouring cells and R_i the residual of cell i, which is the net balance of fluxes evaluated at each cell. Here we assume no mesh motion and Ω_i remains a constant for each cell in the computation.

The system in 2.2 is solved implicitly by first applying the backward Euler scheme:

$$\Omega_i \frac{\Delta U_i}{\Delta t} = -\sum_{j=1}^{N} R_i\left(U_j^n\right) - \sum_{j=1}^{N} \frac{\partial\left(R_i\left(U_j^n\right)\right)}{\partial U_j} \Delta U_j^n \tag{2.7}$$

where the last term is the flux Jacobian. Equation 2.7 can be rearranged and the flux Jacobian is approximated by its spectral radius. The resulting linear system reads:

$$\left[J_i^n \left(\frac{\Omega_i}{J_i^n \Delta t} + 1\right)\right] \Delta U_i^n = -\sum_{j=1}^{N} R_i\left(U_j^n\right) \tag{2.8}$$

Equation 2.8 is the resulting linear system to march the solution from time level n to $n+1$, where J_i^n is the spectral radius of the flux Jacobian matrix which is accumulated across the cell interfaces. Linearised fluxes $\frac{\partial\left(R_i\left(U_j^n\right)\right)}{\partial U_j} \Delta U_j^n$ are required to update the solutions at each Newton–Jacobi iteration and they are evaluated exactly for the inviscid and viscous fluxes. The Newton–Jacobi method is executed for user-specified iterations to march the solution from n to $n+1$, the right and left hand sides are then updated, and the Newton–Jacobi is invoked again. This process proceeds until a user-specified convergence criterion is met.

Unsteady solution

Zephyrus, uses an implicit, dual-time stepping approach for time-dependent problems. Equation (2.6) also applies here, consequently, the treatment of an unsteady problem is analogous to the treatment of steady problems. Again, Equation (2.8) is solved until the desired convergence is reached, as the problem was steady. Then the solution is advanced in physical time by updating the time derivative approximation, according to the backward Euler scheme.

2.2. OpenFoam

This chapter presents an overview of the CFD package used, i.e. OpenFoam v2006, as well as the numerical schemes used to carry out LES and how the system of the governing equations is solved. OpenFOAM is a free, open source CFD software developed primarily by OpenCFD Ltd since 2004, distributed by OpenCFD Ltd and the OpenFOAM Foundation. It has a large user base across most areas of engineering and science, from both commercial and academic organisations.

2.2.1. Governing equation

OpenFoam compressible LES simulations also solve the Favre-filtered compressible Navier–Stokes equations (2.2). In this case the term τ_{ij}^r is calculated according to the one eddy *kEquation* subgrid model (1.10). The settings setted in *turbulence properties* for the simulations presented in this thesis are shown in Figure 2.1.

```
simulationType LES;
LES
{
    LESModel        kEqn;
    kEqnCoeffs
    {
        Ce              1.048;
        Ck              0.02654; // set to approximate a Cs of 0.065
    }
    turbulence      on;
    printCoeffs     on;
    delta           cubeRootVol;
    delta           vanDriest;
    vanDriestCoeffs
    {
        delta           cubeRootVol;
        cubeRootVolCoeffs
        {
            deltaCoeff      1;
        }
        smoothCoeffs
        {
            delta           cubeRootVol;
            cubeRootVolCoeffs
            {
                deltaCoeff      1;
            }
            maxDeltaRatio   1.1;
        }
        Aplus           26;
        Cdelta          0.158;
    }
    smoothCoeffs
    {
        delta           cubeRootVol;
        cubeRootVolCoeffs
        {
            deltaCoeff      1;
        }
        maxDeltaRatio   1.1;
    }
}
```

Figure 2.1: OpenFoam settings in *turbulenceProperties*

The two possible OpenFoam solvers suitable for unsteady compressible LES application are *rhoCentralFoam* and *rhoPimpleFoam*. The former is a density-based solver based on a central-upwind scheme. The latter is a pressure-based solver which uses the PIMPLE algorithm (see below). Unfortunately, while *rhoCentralFoam* is able to solve discontinuity problems, it is not able to model turbulences in a physically correct way due to the dissipative nature of the underlying algorithm [21]. This leaves *rhoPimpleFoam* as the

preferred solver. *rhoPimpleFoam* is solver for unsteady, compressible, non isothermal single phase fluid flows. It uses the PIMPLE algorithms to solve the flow equations.

PIMPLE algorithm

The PIMPLE algorithm is a combination of the SIMPLE (acronym for *Semi-Implicit Method for Pressure-Linked Equations*) developed by Patankar and Spalding and published in 1972 [22] and the PISO algorithm (acronym for *Pressure-Implicit with Splitting of Operators*) published by Issa in 1986 [23].
For each time step, a steady state solution that converges is sought after, using a specified number of correction loops. After that, all other transport equations are solved. This would be equal to the PISO algorithms with the specified number of correction loops. But following this, the algorithm loops back over the entire time step and solves the PISO algorithm again with a new initial guess. A simplified flow diagram of the algorithm is shown in Figure 2.2

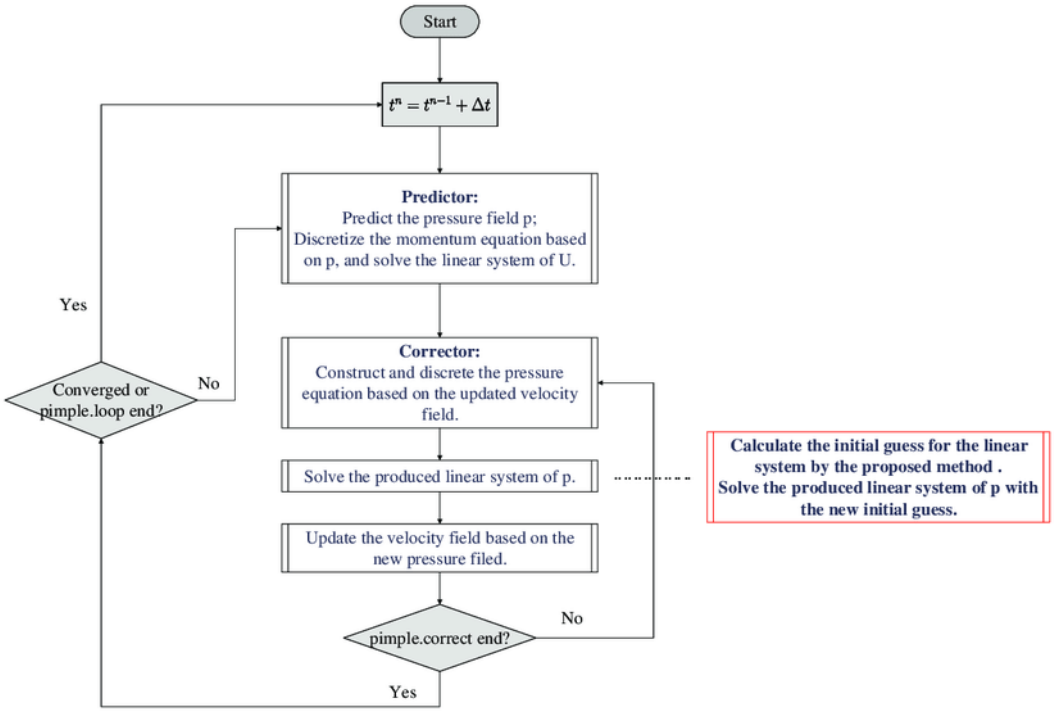


Figure 2.2: Simplified flow chart of the PIMPLE algorithm.

2.2.2. Spatial and temporal discretisation

For a detailed description of the finite-volume discretisation as applied in OpenFOAM refer to de Villiers [24] and Jasak [25]. In the following, the main lines are summarized. The general form of the transport equation for a scalar ϕ in OpenFOAM is:

$$\frac{\partial(\rho\phi)}{\partial t} + \vec{\nabla}\cdot(\rho\vec{u}\phi) - \vec{\nabla}\cdot\left(\rho\Gamma_\phi\vec{\nabla}\phi\right) = S_\phi(\phi) \tag{2.9}$$

where Γ is the diffusivity coefficient. Its integral form for the arbitrary control volume V_P reads:

$$\begin{aligned}\int_t^{t+\Delta t}\left[\int_{V_P}\frac{\partial(\rho\phi)}{\partial t}dV + \int_{V_P}\vec{\nabla}\cdot(\rho\vec{u}\phi)\,dV - \int_{V_P}\vec{\nabla}\cdot\left(\rho\Gamma_\phi\vec{\nabla}\phi\right)dV\right]dt \\ = \int_t^{t+\Delta t}\left[\int_{V_P}S_\phi(\phi)\,dV\right]dt\end{aligned} \tag{2.10}$$

The numerical solution of Eq. 2.10 basically consists of integration and interpolation. In relation to this, the goal is to have a second-order accurate solution in space and time. In order to obtain second-order accuracy, the variation of $\phi = \phi(x,t)$ has to be linear in both space and time, that is, it is assumed that

$$\phi(\vec{x}) = \phi_P + \left(\vec{X} - \vec{x_P}\right)\cdot\left(\vec{\nabla}\phi\right)_P, \tag{2.11}$$

$$\phi(t+\Delta t) = \phi^t + \Delta t\left(\frac{\partial\phi}{\partial t}\right)^t \tag{2.12}$$

where $\phi_P = \phi(\vec{x}_p)$ and $\phi^t = \phi(t)$. Application of Gauss' theorem together with Eq. 2.11 in the spatial integration of Eq. 2.10 yields the following semi-discretised form

$$\begin{aligned}\int_t^{t+\Delta t}\left[\left(\frac{\partial(\rho\phi)}{\partial t}\right)V_P + \sum_f\phi_f F_f - \sum_f(\rho\Gamma_\phi)_f\left(\vec{\nabla}\phi\right)_f\cdot\vec{n}_f S_f\right]dt \\ = \int_t^{t+\Delta t}\left[\left(S_C + S_P\phi_P\right)V_p\right]dt\end{aligned} \tag{2.13}$$

where the face flux F_f is introduced as $F_f = (\rho\vec{u})\cdot\vec{n}_f S_f$ and where the source term S_ϕ has been linearized as $S_\phi = S_C + S_P\phi_P$. Furthermore, the unit normal vector of S_f is denoted by n_f. In this thesis the temporal term is discretised using a second-order implicit backward differencing scheme based on three time levels (as in Zephyrus), named *backward* in OpenFOAM,

$$\frac{\partial\phi}{\partial t} = \frac{\frac{3}{2}\phi^n - 2\phi^{n-1} + \frac{1}{2}\phi^{n-2}}{\Delta t} \tag{2.14}$$

where $\phi^n = \phi(t + \Delta t)$. By assuming that the density, the convective fluxes, and the diffusive fluxes do not change in time within each time step, the final form of the discretised transport equation becomes

$$\frac{\frac{3}{2}\phi^n - 2\phi^{n-1} + \frac{1}{2}\phi^{n-2}}{\Delta t}\rho_P V_P + \sum_f \phi_f^n F_f - \sum_f (\rho\Gamma_\phi)_f \left(\vec{\nabla}\phi\right)_f^n \cdot \vec{n}_f S_f = (S_C + S_P\phi_P^n) V_p \tag{2.15}$$

For the convection term in Eq. 2.15, the values of ϕ need to be interpolated on the cell faces using the so-called Convection Differencing Scheme (CDS) [26].
In OpenFOAM, the central differencing scheme is indicated as the *linear* differencing scheme, since it corresponds to linear interpolation. *Gauss linear* scheme have been used for all the terms in the equations. Furthermore both the laplacian scheme and surface normal gradient scheme are *uncorrected* since the mesh is structured, therefore, its non-orhogonality is very low.
The discretisation procedure produces a system of linear algebraic equations that must be solved for ϕ_P^n. The Geometric agglomerated Algebraic Multi-Grid solver (*GAMG*, also named generalized Geometric-Algebraic Multi-Grid solver in the OpenFOAM manual) has been applied for pressure, while *smoothSolver* GaussSeidel for all the other quantities (Fig. 2.3).

```
solvers
{
    "(p|rho)"
    {
        solver          GAMG;
        tolerance       1e-6;
        relTol          0.001;
        smoother        GaussSeidel;
    }

    "(p|rho)Final"
    {
        $p;
        tolerance       1e-06;
        relTol          0;
    }

    "(U|e|k|nuTilda)"
    {
        solver          smoothSolver;
        smoother        symGaussSeidel;
        tolerance       1e-05;
        relTol          0.01;
    }

    "(U|e|k|nuTilda)Final"
    {
        $U;
        tolerance       1e-05;
        relTol          0;
    }
}
```

Figure 2.3: OpenFoam settings in *fvSolution*

2.3. Summary of the major differences between the two solvers

This section reports the main differences between the Zephyrus and OpenFOAM solvers presented in this chapter. The main difference between the two solvers is the way the Favre filtered NS equations are solved.

RhoPimpleFoam is a compressible pressure-based solver. This means that a pressure equation is solved and the density is related to the pressure via an equation of state. The solver follows a segregated solution strategy. This means that the equations for each variable that characterizes the system are solved sequentially, and the solution of the previous equations is included in the next equation. The nonlinearity that appears in the momentum equation is solved by calculating it from the velocity and pressure values of the previous iteration. In addition, the discretization scheme for gradients is based on the standard finite-volume discretization of Gaussian integration.

Zephyrus, on the other hand, is a density-based CFD code, which solves the equations via Roe's Riemann solver implicitly. The gradient discretization scheme is still second-order, but based on the least-squares method.

3 | Reference case

To quantitatively compare the results, the incompressible DNS dataset reported in [5] is chosen as a reference. The reference data compare two different bump geometries described analytically as the sum of two overlapping Gaussian curves:

$$G_1(x) = a \cdot exp\left[-\left(\frac{x-b}{c}\right)^2\right] + a' \cdot exp\left[-\left(\frac{x-b'}{c'}\right)^2\right] \tag{3.1}$$

The values of the parameters are $a = 0.0505, b = 4, c = 0.2922$ and $a' = 0.060425, b' = 4.36, c' = 0.3847$, resulting in a bump with maximum height $h_b = 0.0837\delta$, with δ representing the channel half-height. The bump is similar to the one considered by Marquillie *et al.* [27], but with significantly smaller size, to reduce blockage. The two resulting bumps, G_1 and G_2, have the same height and produce an attached and a separated flow, respectively. The G_2 geometry is identical to G_1 in the front part, from the domain inlet to the bump tip, but shows a streamwise expansion factor 2.5 applied to the rear part of the bump. In this thesis, only the G_2 geometry is considered, as it produces an attached flow.

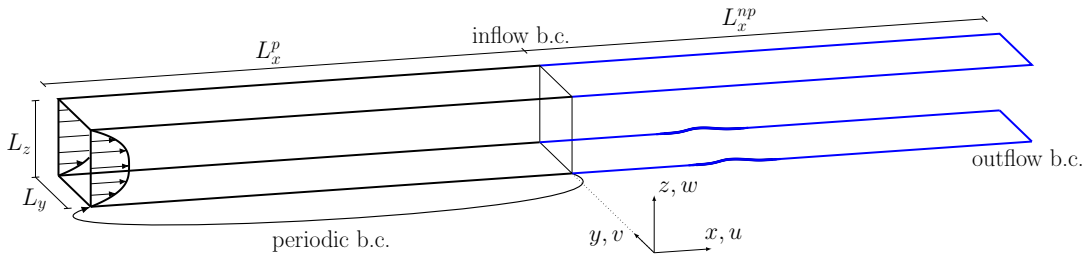


Figure 3.1: Sketch of the computational domain used in [5].

The computational domain adopted in the DNS study, shown in Fig. 3.1, consists of two domains adjacent along the streamwise direction: a channel with parallel walls and periodic conditions in both streamwise and spanwise directions, which feeds the downstream domain with the bump. The downstream domain has cyclic conditions in the spanwise

direction. It takes as inflow the outflow from the upstream domain, whereas convective conditions are used at the outlet [28]. No-slip and non-penetration conditions are applied on the walls of both domains.

The size of the whole computational domain is $(L_x^p + L_x^{np}, L_y, L_z) = (4\pi\delta + 12\delta, \pi\delta, 2\delta)$ in the streamwise, spanwise and wall-normal directions respectively. The upstream portion of the domain simulates a standard DNS turbulent channel flow of length $L_x^p = 4\pi\delta$ with constant flow rate. The channel domain has a spatial resolution of $(n_x, n_y, n_z) = (360, 312, 241)$. The discretization points are uniformly spaced in streamwise and spanwise directions, with $\Delta x = 0.04$ that corresponds to $\Delta x^+ = 8$ and $\Delta y = 0.01$ equivalent to $\Delta y^+ = 2$ based on the inlet u_τ. In the wall-normal direction the discretization points are neither uniformly nor symmetrically distributed with respect to the centerline. A constant $\Delta z = 0.001$, which corresponds to $\Delta z^+ = 0.2$ at the inlet, is adopted from the lower wall to the height corresponding to the tip of the downstream bump, $z = h_b$. Then Δz gradually increases until $\Delta z = 0.02$ at the centerline, and then decreases again to $\Delta z = 0.004$ at the upper wall.

The downstream domain with the bump is discretized with $(n_x, n_y, n_z) = (800, 312, 241)$ discretization points. In the wall-normal and spanwise directions the same discretization is maintained to avoid interpolation at the interface. While in the streamwise direction the grid spacing is not uniform, rather it is finer near the bump, reaching a minimum up to $\Delta x^+ = 2$.

The simulations are carried out at a bulk Reynolds number $Re_b = U_b\delta/\nu = 3173$ which in the reference case corresponds to a friction Reynolds number $Re_\tau = u_\tau\delta/\nu = 200$ in the plane channel, where the velocity scale is the bulk velocity U_b in the former case and the friction velocity $u_\tau = \sqrt{\tau_w/\rho}$ in the latter. The time step is set in order to obtain an average Courant–Friedrichs–Lewy (CFL) number of approximately 0.5.

This choice of an incompressible dataset as a reference might seem not entirely appropriate for the assessment of a compressible solver. However, high-fidelity simulations of compressible turbulent channels and bumps in the literature often involve transonic or supersonic flows, or flows at high Reynolds number. High Reynolds numbers would have resulted in prohibitively computationally expensive grid resolutions, also due to the lack of a SGS model, while it is intended to avoid simulating a supersonic test case so as to not increase the complexity of the problem. Therefore, the domain setup is adapted to simulate a flow with a Mach number $Ma = 0.14$, low enough to stay reasonably below the compressible threshold, with the compressibility effects being less than 1%.

4 | Simulation Setup

The scope of this work is to assess the capability of the compressible CFD solver Zephyrus, to simulate the turbulent flow over a bump, through Implicit Large Eddy Simulation. To achieve the goal, an LES of a turbulent channel is carried out with OpenFOAM in order to generate the instantaneous fields to be given in inflow to the simulation with the bump. In addition, the domain with the bump is simulated by both Zephyrus and OpenFOAM, so that a comparison between two compressible LES simulations can be performed.
In this chapter, the grid generation, the applied boundary conditions and the setup of the simulations performed is therefore reported.

4.1. Channel

In the reference case chapter 3, the computational domain with the bump is directly fed by the outflow of a turbulent channel flow at $Re_b = 3173$ and $Re_\tau = 200$. In Zephyrus, since an algorithm to impose constant flow rate along with cyclic boundary conditions is not yet implemented, it was not possible to simulate the turbulent channel flow and couple the domains. Therefore, it was not possible to replicate exactly what is done in the DNS reference case. To perform the preliminary simulation of the turbulent channel flow, necessary to obtain the instantaneous fields needed in inflow to the domain with the bump, OpenFOAM is employed, which has already been validated for this type of simulation [29]. Through the *sampling* function, the instantaneous fields of velocity, pressure, temperature, subgrid viscosity and k at the channel outlet are saved to file at each timestep.
Multiple ways exist to simulate a fully turbulent channel flow; a possibility is to enforce at the inlet of the channel the synthetic turbulent fluctuations through the function *turbulentDFSEMInlet* [30]. To enable reproducibility, a more traditional approach is considered. The flow is first initialized with a steady RANS simulation with the $k-\omega$ turbulence model, and then it is used as the starting solution for the LES simulation.

4.1.1. Grid generation

In LES simulations the numerical grid plays a fundamental role in obtaining accurate results. This kind of simulations computes the solution of the filtered equations, which resolve large eddies, while eddies below the filter size are modeled. The separation between computed and modelled eddies is set by the filter width, linked directly to the mesh resolution. A systematic grid-convergence study would offer no greater benefit, as a grid-independent LES is a DNS and therefore loses its economical advantage on DNS, on account of resolving only the most energetic eddies [31].
There are several databases and DNS studies of turbulent channel flow [15, 32] that is possible to refer to in order to have an estimate a priori of the Kolmogorov length scale η and of the number of discretization points necessary to have a sufficient resolution of the grid. Based on these considerations and considering the amount of computational resources needed to complete the simulations, the channel domain $(L_x, L_y, L_z) = (12\delta, \pi\delta, 2\delta)$ is discretized with $(n_x, n_y, n_z) = (207, 110, 186)$ points in the streamwise, spanwise and wall-normal direction respectively. The computational grid is created with the open-source software GMSH [33].
The grid spacing in spanwise and streamwise direction is uniform and corresponds to $\Delta x^+ = 12$ and $\Delta y^+ = 6$ respectively, values similar to those from Kim *et al.* [34]. In the wall-normal direction a non-uniform distribution according to the *bump* algorithm is implemented. This is a GMSH algorithm that locally increases the point density near the wall, keeping the discretization coarse close to the centerline. In this way it is possible to obtain a sufficiently fine grid close to the walls, where the main phenomena of viscous dissipation take place, and slightly coarser near the centerline, saving computational resources. The wall cell height in inner units corresponds to $z^+ = 0.1$.

4.1.2. Boundary conditions

The boundary conditions are cyclic in both the spanwise and streamwise direction, a constant flow rate (CFR) is imposed through the OpenFOAM option *meanVelocityForce* applied to the entire domain, tuned with a bulk velocity that leads to $Re_b = 3173$. On the lower and upper walls, no-slip and no-penetration boundary conditions are imposed. *zeroGradient* boundary condition is enforced at the walls for the temperature and pressure fields. The boundary conditions for the different variables are summarized in Table 4.1.

Patch	U	p	T	ν_t	k
Inlet Outlet Sides (y-dir)	cyclic	cyclic	cyclic	cyclic	cyclic
Walls (z-dir)	*noSlip*	*zeroGradient*	*zeroGradient*	*fixedValue: 0*	*fixedValue: 0*

Table 4.1: OpenFOAM channel boundary conditions.

4.1.3. LES simulation settings

The timestep is set to $\Delta t = 1.5 \times 10^{-7}$, resulting in a $CFL_{mean} = 0.1$ and $CFL_{max} = 0.3$. The numerical schemes employed in OpenFOAM, described in Section 2.2.2, are second-order accurate in space and time: the *Gauss linear* divergence scheme has been used for all the terms in the equations except for pressure, for which Linear-Upwind Stabilised Transport ($LUST$) is employed. Furthermore both the laplacian scheme and surface normal gradient scheme are *uncorrected*. The *backward* scheme is used for the temporal term. Regarding the solvers settings, the *GAMG* is employed for the pressure while *smoothSolver* for all the other quantities. Among the subgrid stress models available in OpenFOAM, the *kEquation* model [35] is selected.

4.2. Bump

The bump domain is simulated by both Zephyrus and OpenFOAM, with the same computational grid, so that a comparison can be made between two compressible LES simulations.

4.2.1. Grid generation

The same spatial discretization implemented in the channel domain and described in Section 4.1.1, is also used for the bump: the domain $(L_x, L_y, L_z) = (12\delta, \pi\delta, 2\delta)$ is discretized with $(n_x, n_y, n_z) = (207, 110, 186)$ discretization points. Consequently, the cell sizes in inner units are $\Delta x^+ = 12$, $\Delta y^+ = 6$ and $\Delta z^+ = 0.1$ at the inlet and $\Delta x^+ = 20$, $\Delta y^+ = 10$ and $\Delta z^+ = 0.2$ at the bump tip, according to the local u_τ.

4.2.2. Boundary Conditions

The spanwise direction is made homogeneous by applying cyclic conditions, whereas no-slip and non-penetration conditions are applied to the lower and upper wall. At the bump inflow constant flow rate is enforced, through the imposition of the instantaneous fields obtained with the CFR cannel. In OpenFOAM the function *timeVaryingMappedFixedValue* is employed to enforce the instantaneous fields at the bump domain inflow. In Zephyrus, a boundary condition named *TuInl* (Turbulent Inlet) has been implemented: the variables (face center coordinates, velocity, temperature and pressure) are read from file and set in the ghost-cell nearest to the given position.
To obtain a well-posed and stable compressible simulation, several possible combinations of boundary conditions exist. The one used in our case is: constant mass flow rate in inflow, and fixed static pressure in outflow. In OpenFOAM the outlet static pressure is set with the boundary condition *fixedValue*, whereas for other variables the condition *zeroGradient* is set. In Zephyrus, the boundary condition *FreeExit* is employed, imposing the static pressure at the outlet and extrapolating the remaining variables from the inner domain.
For a faster comparison, the boundary conditions implemented in OpenFOAM and Zephyrus are shown in the following Tables 4.2 to 4.5.

Patch	Condition OpenFOAM	Condition Zephyrus
Inlet	*timeVaryingMappedFixedValue*	*TuInl*
Outlet	*zeroGradient*	*FreeExit*
Sides (y-dir)	*cyclic*	*cyclic*
Walls (z-dir)	*no-slip, non penetration*	*no-slip, non penetration*

Table 4.2: Bump boundary conditions, Velocity field U.

Patch	Condition OpenFOAM	Condition Zephyrus
Inlet	*timeVaryingMappedFixedValue*	*TuInl*
Outlet	*fixedValue*	*FreeExit*
Sides (y-dir)	*cyclic*	*cyclic*
Walls (z-dir)	*zeroGradient*	*zeroGradient*

Table 4.3: Bump boundary conditions, Pressure field p.

Patch	Condition OpenFOAM	Condition Zephyrus
Inlet	*timeVaryingMappedFixedValue*	*TuInl*
Outlet	*zeroGradient*	*FreeExit*
Sides (y-dir)	*cyclic*	*cyclic*
Walls (z-dir)	*zeroGradient*	*zeroGradient*

Table 4.4: Bump boundary conditions, Temperature field T.

Patch	ν_t	k
Inlet	*timeVaryingMappedFixedValue*	*timeVaryingMappedFixedValue*
Outlet	*zeroGradient*	*zeroGradient*
Sides (y-dir)	*cyclic*	*cyclic*
Walls (z-dir)	*fixedValue* 0	*fixedValue* 0

Table 4.5: OpenFOAM Bump boundary conditions, ν_t and k fields.

4.2.3. LES Simulation Settings

The time step remains unchanged, $\Delta_t = 1.5 \times 10^{-7}$, corresponding to an average $CFL = 0.4$ and to a maximum $CFL = 0.6$. The numerical methods and the subgrid model used in OpenFOAM are as given in 4.1.3. In Zephyrus the numerical methods are second order accurate: MUSCL with Van Albada limiter in space and backward Euler in time.

5 | Results

5.1. Turbulent channel flow

The results presented in this paragraph are related to the fully turbulent channel flow obtained in OpenFOAM. This is a preliminary simulation necessary to generate the turbulent fields for the inlet of the bump domain.
The channel is initialized with a RANS simulation with the $k-\omega$ turbulence model to obtain a preliminary solution with a developed turbulent boundary layer and with an initial field for the turbulent kinetic energy k, so as to help convergence for the first time-steps of the LES simulation and to have the field k initialized for the SGS $kEquation$ model.
Before starting to save variables on the outlet plane, 120 $\bar{t}$ were needed for the flow to develop into a statistically stable state. Here $\bar{t}$ is defined as:

$$\bar{t} = \frac{\delta}{U_b} \tag{5.1}$$

Subsequently, about 600 $\bar{t}$ have been used for accumulating the istantaneous fields (roughly 80000 timesteps).

5.1.1. Istantaneous and Mean flow properties

To begin with a qualitative picture of the flow, Fig. 5.1 represents the instantaneous fluctuations of the three velocity components in the channel adimensionalized by the mean bulk velocity U_b. The statistics are averaged for 600 $\bar{t}$ with the use of the OpenFOAM function *fieldAverage*. The Reynolds number based on the bulk mean velocity and the half-height of the channel (2δ) is $Re_m = 3143$. The skin friction coefficient, $C_f = \tau_w / \frac{1}{2}\rho U_b^2 = 7.826 \times 10^{-3}$, which is in good agreement with Dean's correlation of $C_f = 0.073 Re_m^{-0.25} = 8.1799 \times 10^{-3}$ and in excellent agreement with Banchetti *et al.* [5], $C_f = 7.833 \times 10^{-3}$. The ratio between the mean velocity at the centerline and the bulk mean velocity, $U_c/U_b = 1.164$, again in excellent agreement with Dean's correlation, $U_c/U_b = 1.28 Re_m^{-0.0116} = 1.156$ and the averaged value by Banchetti *et al*, $U_c/U_b = 1.168$.

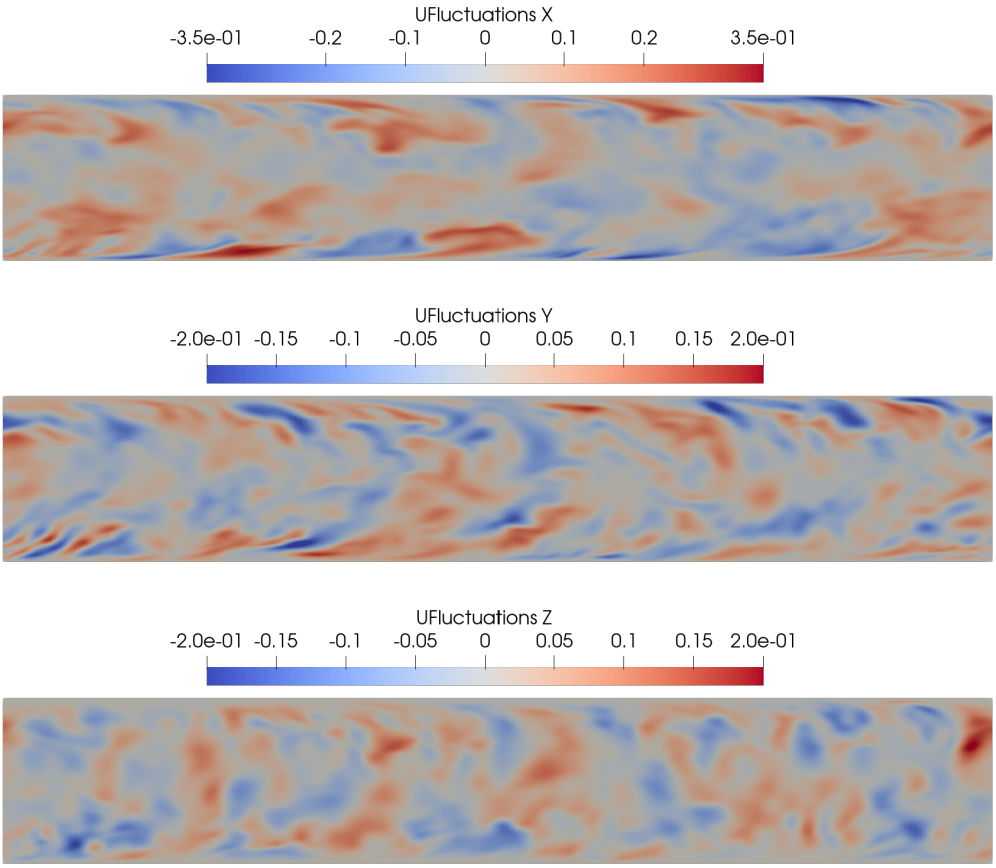


Figure 5.1: Istantaneous velocity fluctuations field $u' = \frac{U_{mean}-U}{U_b}$.

5.1.2. SGS contribution

Since it was possible to set the SGS model only in OpenFOAM simulations, it is interesting to investigate the contribution of the modelled turbulent viscosity with respect to the physical viscosity of the fluid.

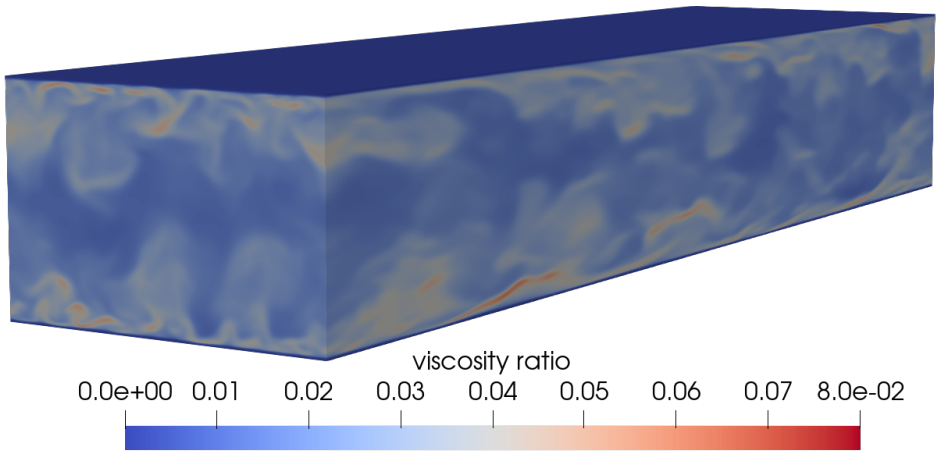


Figure 5.2: Representation of an instantaneous field ν_t/ν.

It is observed that in an instantaneous field, Fig. 5.2, the maximum value of the modelled turbulent viscosity ν_{SGS} is less than 10% of the value of the physical viscosity ν. Over the whole domain, the model's contribution to the effective viscosity, $\nu_{eff} = \nu + \nu_{SGS}$, is about 2%.
Alternatively, the ratio of the resolved turbulent kinetic energy to the total turbulent kinetic energy (namely the LES index) is analysed, leading to similar results:

$$LESindex = \frac{k_{res}}{k_{res} + k_{SGS}} \times 100 = 97.8\% \tag{5.2}$$

5.2. Bump

Once the outflow statistics of the channel flow are saved for a total of 600 $\bar{t}$, it is possible to begin simulating the domain with the bump with the two codes.
Some analysis about the contribution of the SGS model are carried out. This preliminary analysis allows to verify that Zephyrus results are not adversely affected by the absence of the model.

5.2.1. OpenFOAM bump analysis

To study the effects of the sub-grid model in detail, some *function Objects* are created in OpenFOAM to compute the fields: resolved turbulent kinetic energy k_{res}, SGS dissipation

ϵ_{SGS}, viscous dissipation ϵ_{visc}, turbulent kinetic energy production P. Their implementation is reported in Appendix A.

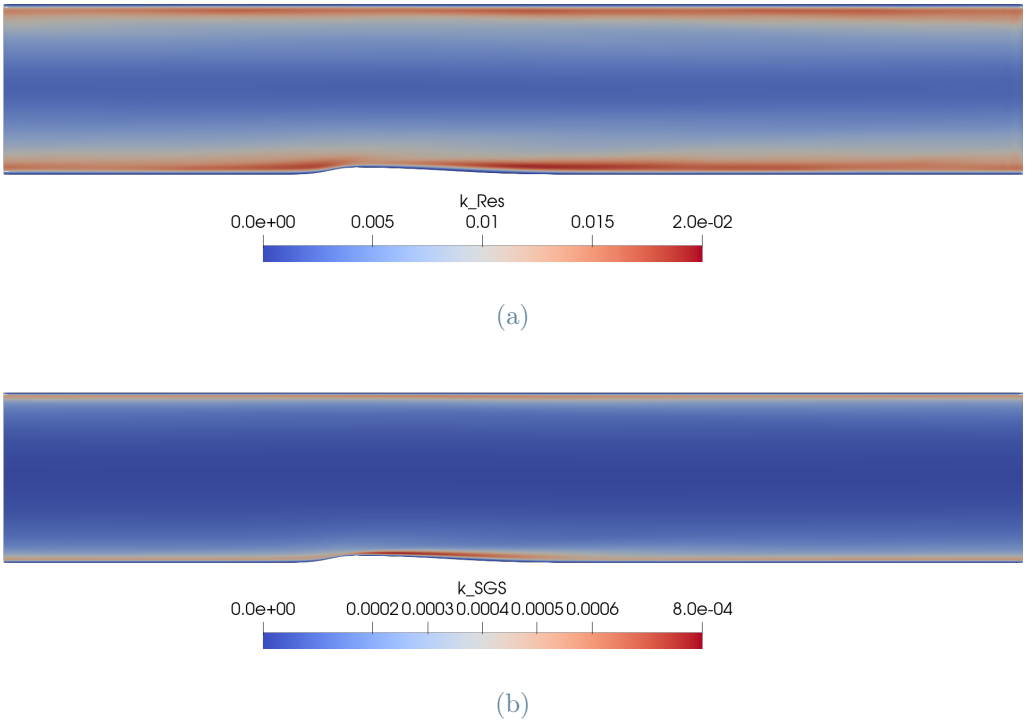
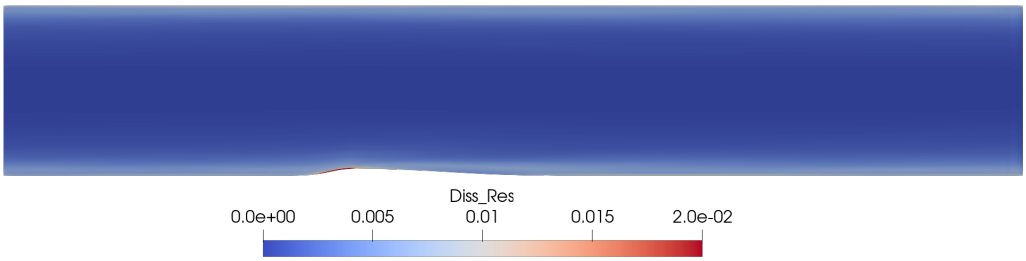


Figure 5.3: Plot of the turbulent kinetic energy resolved (a) and modeled (b).

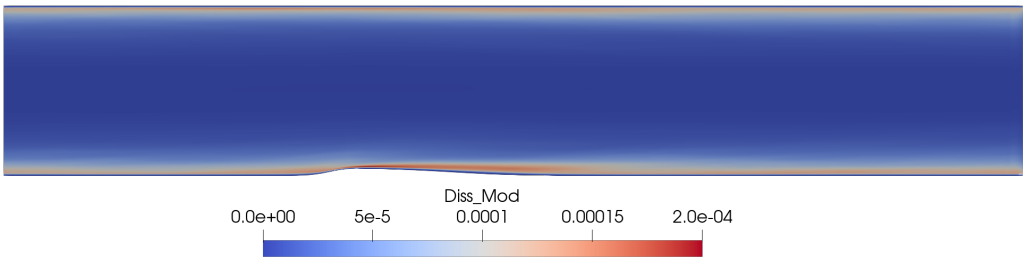
Fig. 5.3 shows the modeled k_{SGS} and resolved turbulent kinetic energy:

$$k_{res} = \frac{1}{2} \langle u_i' u_i' \rangle \tag{5.3}$$

The resolved turbulent kinetic energy (TKE), in agreement with the literature, shows two main areas of high k: one just before the bump and one, more intense, toward the end of the bump. The modeled turbulent kinetic energy has its maximum intensity just after the tip of the bump, with a value equal to 3.4% of the maximum intensity of the resolved TKE. In general, it can be observed that the model does not operate at the cells at the wall, but at a distance from the wall of a few cells; this is due to the discretization methodology. A smoother rate of progression, or a larger number of discretization points in the wall-normal direction, would allow for smaller cells in that area, resulting in a smaller fraction of modeled TKE.



(a)



(b)

Figure 5.4: Plot of the viscous dissipation (a) and SGS dissipation (b) of turbulent kinetic energy.

Fig. 5.4 shows the turbulent kinetic energy dissipation due to physical viscosity and due to SGS viscosity.

$$\epsilon_{visc} = 2\nu \left(S_{ij} S_{ij}\right) \tag{5.4}$$

$$\epsilon_{SGS} = 2\nu_{SGS} \left(S_{ij} S_{ij}\right) \tag{5.5}$$

Maximum resolved dissipation is concentrated at the front of the bump , with intermediate values behind the tip and near wall. Again, as in the channel, the turbulent viscosity ν_t is much lower than the physical viscosity ν, consequently the SGS dissipation is also lower than the resolved dissipation. Regarding its distribution in the domain, the same considerations made for the k_{SGS} apply.

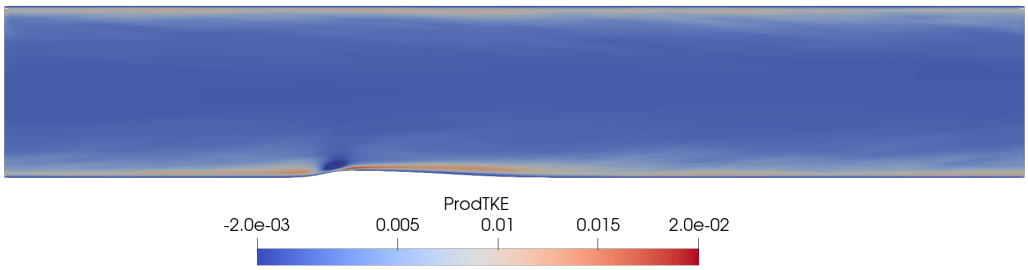


Figure 5.5: Plot of the production P of turbulent kinetic energy.

Turbulent kinetic energy production:

$$P = -\left\langle u_i' u_j' \right\rangle \frac{\partial \left\langle u_i' \right\rangle}{\partial x_j} \tag{5.6}$$

is plotted in Fig.5.5 to show that its spatial distribution is consistent with k and that the values are comparable with those found by Banchetti *et al.* [5].
It can be concluded that the grid is fine enough to be able to capture the main mechanisms of turbulent kinetic energy production and dissipation. Furthermore, a test simulation in OpenFOAM has been carried out without a subgrid stress model and the results have been averaged for 120 $\bar{t}$, showing no significant difference between the results of the two simulations.

5.2.2. Solvers comparison

Since a function to average run-time the variable fields is not available in Zephyrus, a database of the results obtained throughout 600 $\bar{t}$ is created. In order to obtain the averaged statistics in Zephyrus the solution is saved every 5 $\bar{t}$ ($\approx$ 700 timesteps), hence obtaining a database of 160 statistically independent timesteps and saving space on the HPC clusters.
The velocity profiles and the six components of the Reynolds stress tensor for seven increasing x-coordinates of the domain are shown in Figures 5.6 to 5.12.

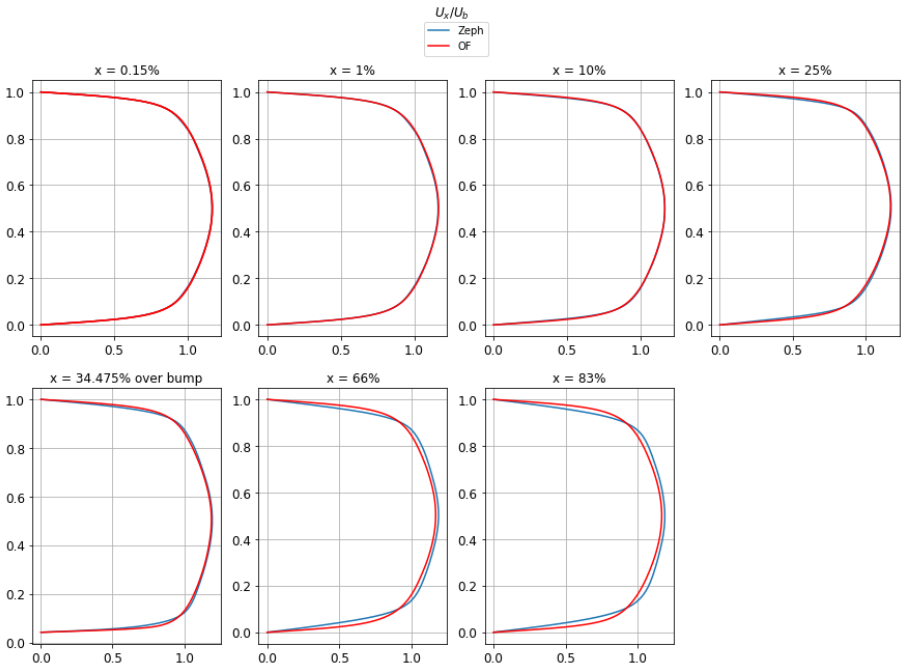


Figure 5.6: Plot of the bulk mean velocity profile at increasing x-coordinates.

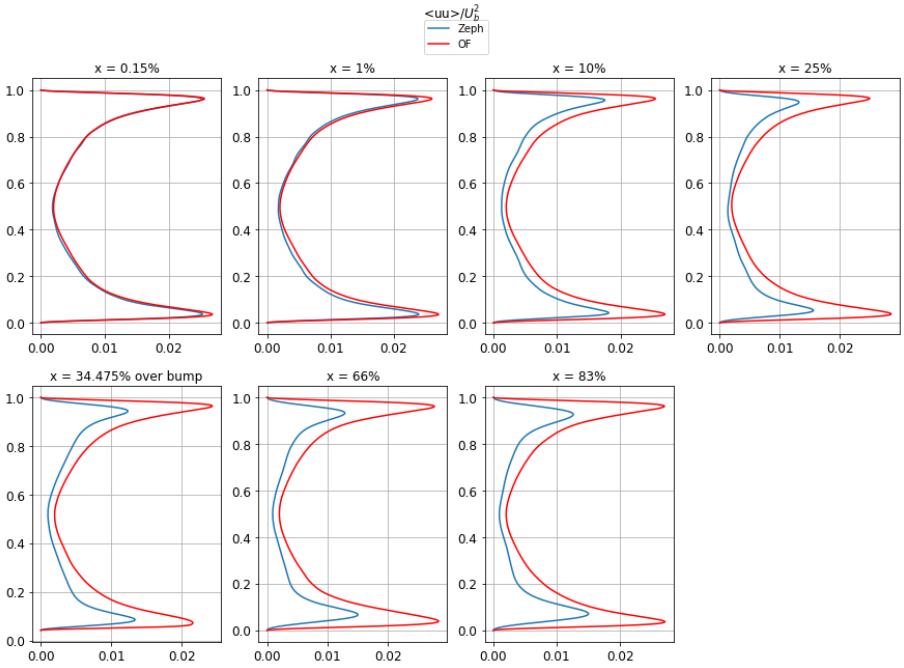


Figure 5.7: Plot of the first component of the Reynolds stresses $\langle u'u' \rangle$ at increasing x-coordinates.

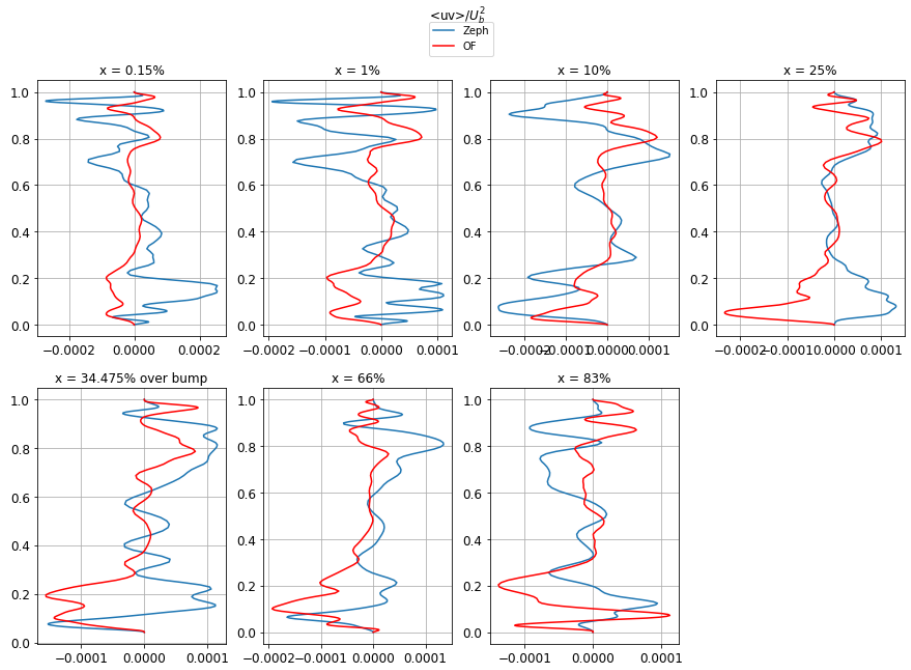


Figure 5.8: Plot of the second component of the Reynolds stresses $\langle u'v' \rangle$ at increasing x-coordinates.

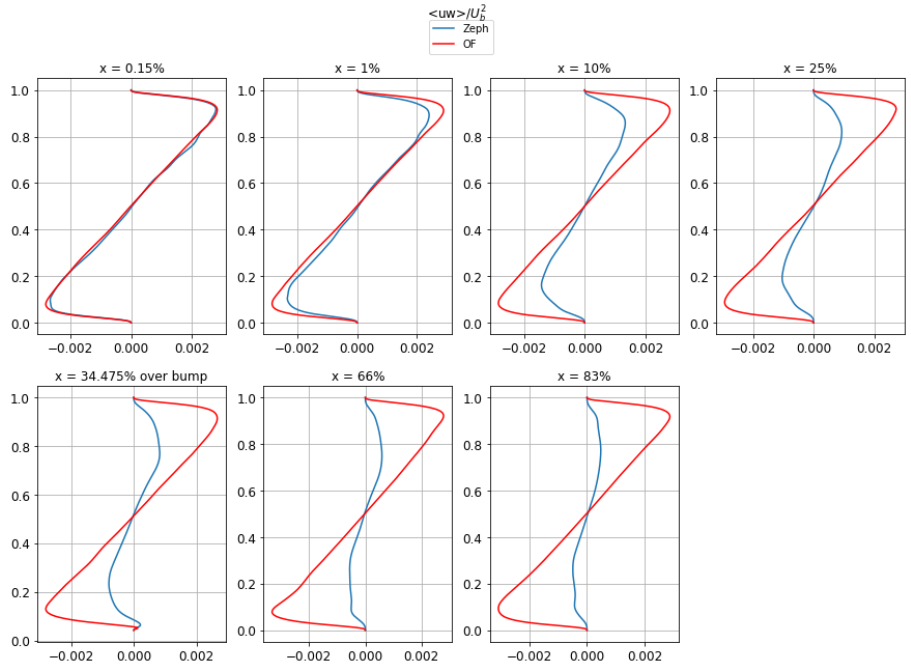


Figure 5.9: Plot of the third component of the Reynolds stresses $\langle u'w' \rangle$ at increasing x-coordinates.

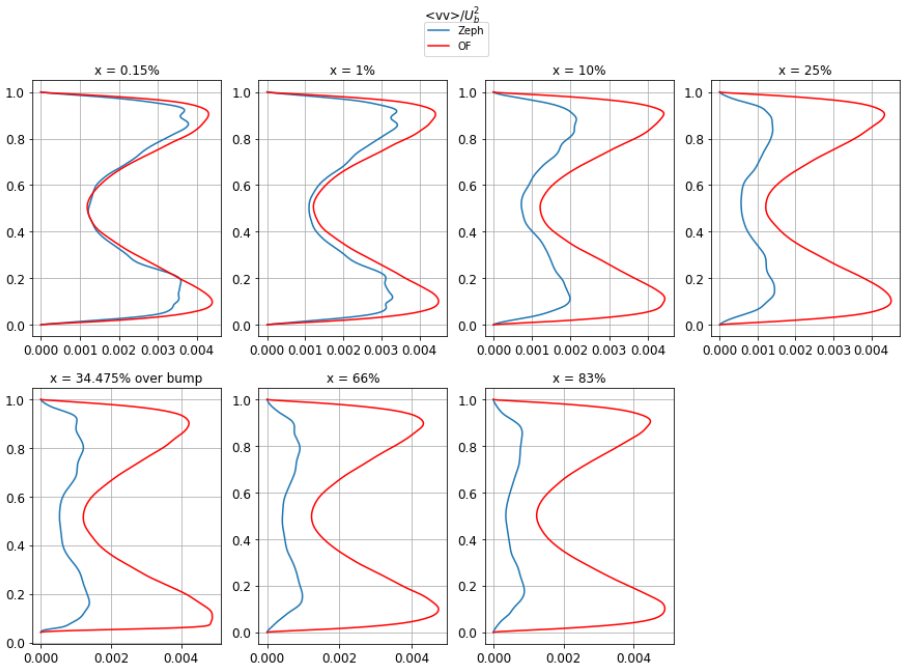


Figure 5.10: Plot of the fourth component of the Reynolds stresses $\langle v'v' \rangle$ at increasing x-coordinates.

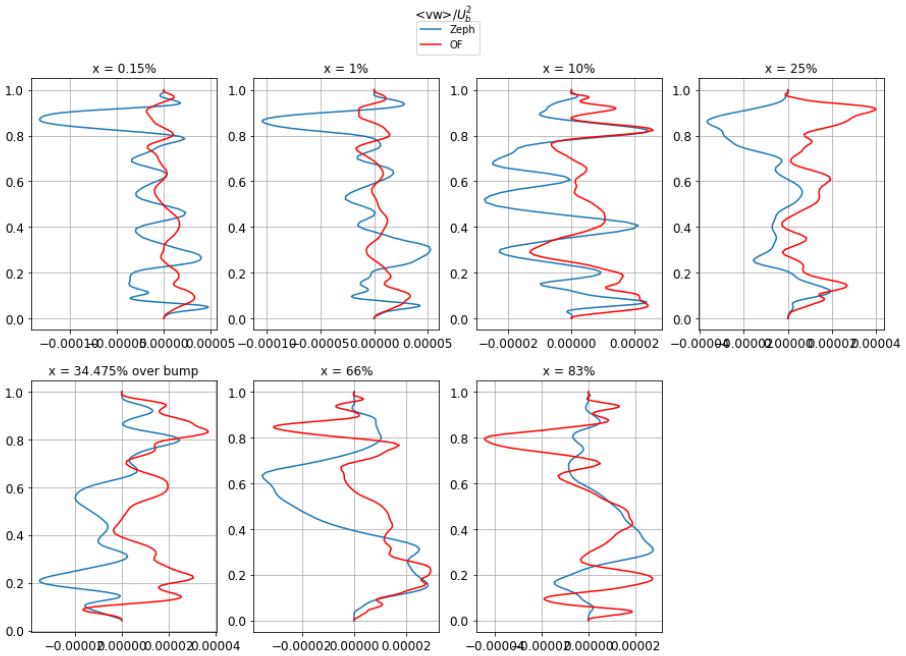


Figure 5.11: Plot of the fifth component of the Reynolds stresses $\langle v'w' \rangle$ at increasing x-coordinates.

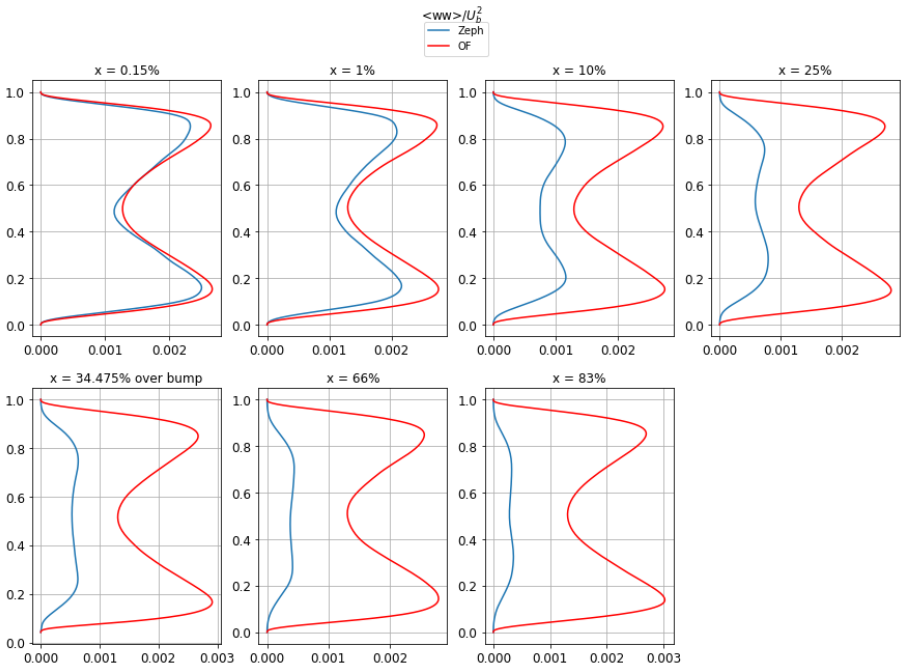


Figure 5.12: Plot of the sixth component of the Reynolds stresses $\langle w'w' \rangle$ at increasing x-coordinates.

At the inlet, first plot at $x = 0.15\%L_x$ in Figures 5.6 to 5.12, corresponding to the solution in the first cell center streamwise, the velocity profile and Reynolds stress components have no significant differences between the two solvers, apart from the stress tensor component $\langle w'w' \rangle$ (recall that w is the wall-normal velocity for how the axes of the reference system are oriented). Indeed, it is evident how the wall-normal fluctuations are attenuated after the first cell already. This phenomenon, related to the lower intensity of the fluctuations, also occurs in the other components of the Reynolds stresses and become more pronounced as the x-coordinate increases along the bump domain. At $x = 83\%L_x$ the $\langle u'u' \rangle$ component is found to be significantly lower compared to the value obtained with OF, whereas for the $\langle u'w' \rangle$,$\langle v'v' \rangle$ and $\langle w'w' \rangle$ components the reduction is more pronounced. The decrease in the intensity of the velocity fluctuations also affects the average velocity profile. At the outlet, the mean velocity profile obtained with Zephyrus deviates significantly from the mean turbulent profile obtained with OpenFOAM, showing a lower derivative at the wall and a higher mean velocity in the area of the domain near the centerline, which is consistent with less turbulent flow.

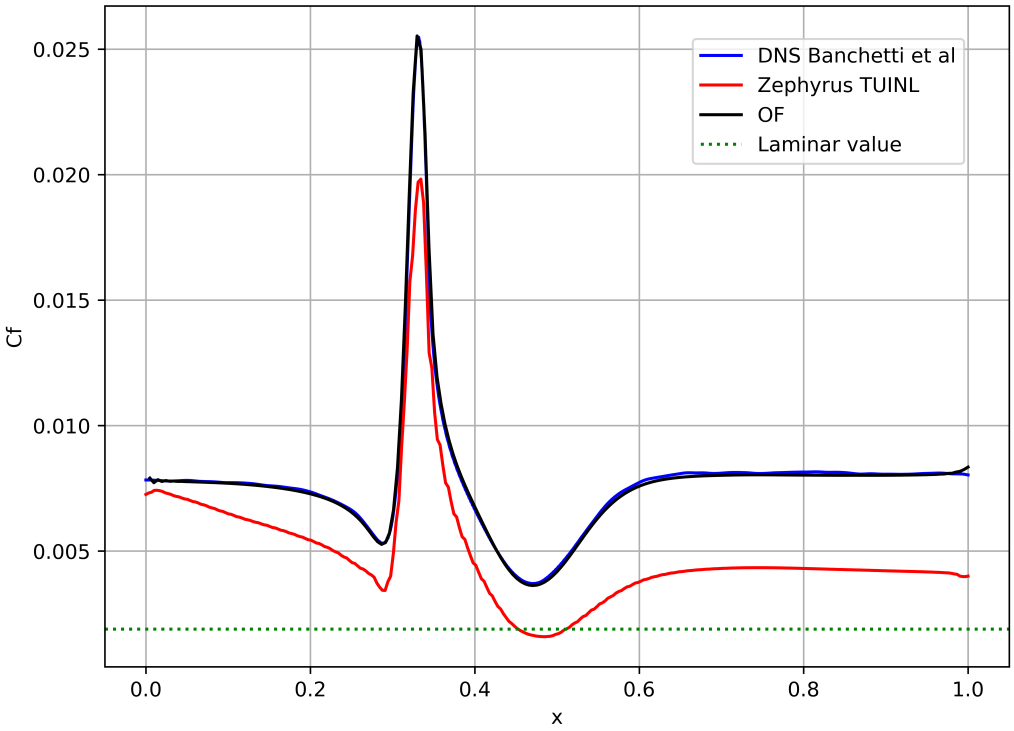


Figure 5.13: Comparison plot of the skin friction coefficient C_f.

The skin friction coefficient:

$$C_f(x) = \frac{2\langle \tau_w(x) \rangle}{\rho U_b^2} \tag{5.7}$$

is compared in Fig. 5.13. At the inlet, the C_f value is similar between the two codes, with OpenFOAM exhibiting very good agreement with the incompressible DNS data. The C_f obtained in Zephyrus drops dramatically in the first few cells, consistent with the decrease in the intensity of the velocity fluctuations, to a value roughly half of the reference value. The Re_τ drops from the value of $Re_\tau = 200$ at the inlet to $Re_\tau = 150$ after the bump, which is retained until the domain outlet. The turbulent solution thus tends to re-laminarize.

The calculated pressure coefficient is:

$$C_p(x) = \frac{2\langle p \rangle(x)}{\rho U_b^2} \tag{5.8}$$

Where the mean pressure $\langle p \rangle$ is set to 0 at the outlet of the domain.

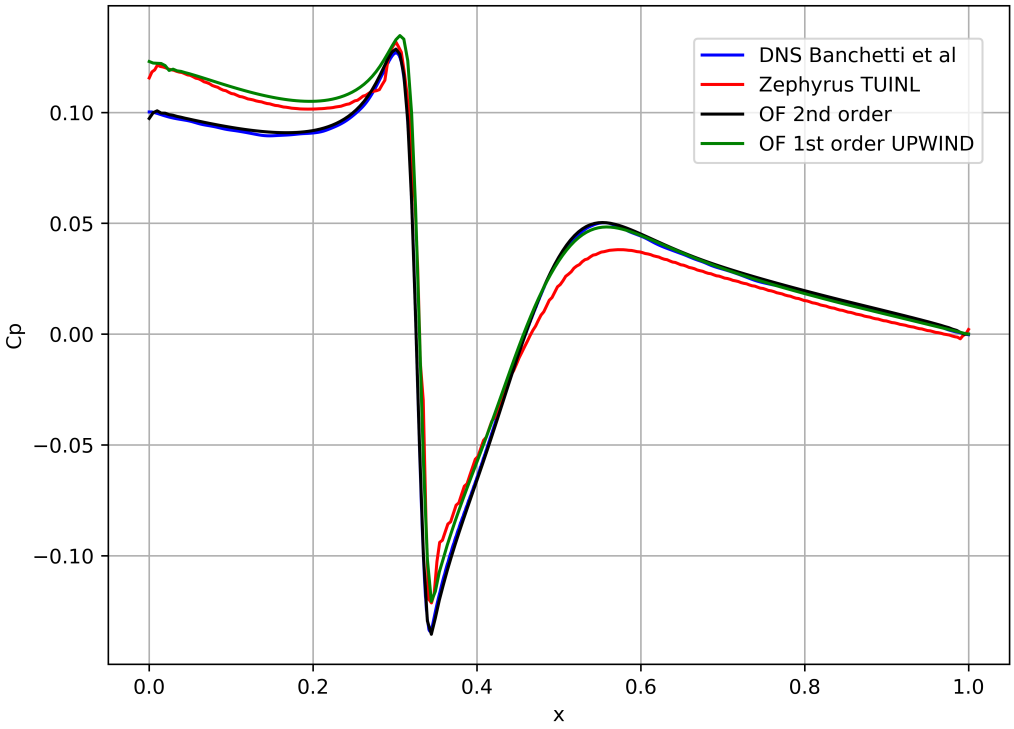


Figure 5.14: Comparison plot of the pressure coefficient C_p.

Fig. 5.14 shows the mean C_p averaged in time and in space over the lower wall. The pressure coefficient has a minimum exactly at the tip of the bump. Away from the bump, the pressure coefficient has a linear trend (i.e. uniform mean pressure gradient), as expected for a plane channel flow. Similarly to C_f, OpenFOAM has an excellent agreement with DNS data. Zephyrus has a different behavior: at the outlet the pressure maintains the value enforced via the *Free Exit* boundary condition, while it adapts the inflow pressure read from file to a higher value, leading to a Δp across the domain greater than expected. A trend similar to the one just reported is observed in the results averaged for 120 $\bar{t}$ of an OpenFOAM LES simulation with the *kEquation* subgrid model and first-order accurate *UPWIND* numerical schemes in space, backward Euler second-order scheme in time. OpenFOAM results with first-order accurate numerical schemes in space can be justified by the highly diffusive numerical method, which leads to the decrease in turbulent fluctuations and flow transition to a lower Re_τ.

As for Zephyrus, in which the numerical schemes are second-order accurate both in space and time, further investigation is needed. The fundamental turbulent flow analyzed is mainly characterized by being a wall-bounded flow and by the presence of a curvature

(bump) on the lower wall. It might be useful to separate these two effects to assess the presence in the code of any problem regarding the inner face numerical flux (and thus related to the dissipation of the numerical method) or related to the treatment of the wall face flux near the bump curvature.

6 | Conclusions and future developements

In this thesis, the main features of Large Eddy Simulations with and without SGS model are reviewed. The scope is to assess the ability of the compressible CFD solver Zephyrus to simulate a turbulent flow over a bump using ILES and to develop the routines needed to get the task accomplished.
A turbulent channel flow is simulated in OpenFOAM to generate the instantaneous turbulent fields needed as inflow boundary condition for the doamin with the bump. The effect of the SGS model set in OpenFOAM is carefully analyzed to verify that the grid resolution is sufficient to capture the main phenomena related to turbulent kinetic energy k. To enable the solver Zephyrus to switch to solving ILES, modifications to the code were necessary in terms of pre-processing, boundary conditions, and post-processing functions. The results show that in Zephyrus the turbulent fluctuations are greatly reduced after few cells from the inlet and that the flow tends to re-laminarize toward an intermediate turbulent condition corresponding to $Re_\tau = 150$. This diffusion phenomenon of turbulent fluctuations also affects the inlet-outlet pressure gradient. Counter-intuitively, a decrease in the friction coefficient corresponds to an increase of Δp. However, this phenomenon is also displayed in an OpenFOAM simulation with UPWIND first-order accurate numerical schemes in space.
In Zephyrus the cause of the decreased intensity of turbulent fluctuations has been attributed to the high difssipation of the implemented numerical methods, which cause the transition to a lower Re_τ. As for Zephyrus, further investigations are needed. The fundamental turbulent flow analyzed is mainly characterized by being a wall-bounded flow and by the presence of a curvature (bump) on the lower wall.
Future developments involve investigating further and simulating in Zephyrus a turbulent channel flow with parallel walls (eliminating the domain curvature) and performing a study similar to the one presented by Saad et al in [36]: a study of the decay of homogeneous isotropic turbulence in a 3D cube with cyclic conditions applied in all directions, allowing the dissipation of the numerical method to be evaluated in the absence of walls.

Bibliography

[1] H. Choi and P.Moin. Grid-point requirements for large eddy simulation: Chapman's estimates revisited. *Physics of Fluids*, 24(1):011702, 2012.

[2] M. Breuer, N. Peller, Ch. Rapp, and M. Manhart. Flow over periodic hills – Numerical and experimental study in a wide range of Reynolds numbers. *Computers & Fluids*, 38(2):433–457, 2009.

[3] L. Temmerman and M. A. Leschziner. Large eddy simulation of separated flow in a streamwise periodic channel constriction. In *Proceeding of Second Symposium on Turbulence and Shear Flow Phenomena*, pages 399–404, KTH, Stockholm, Sweden, 2001. Begellhouse.

[4] X. Wu and K. D. Squires. Numerical investigation of the turbulent boundary layer over a bump. *Journal of Fluid Mechanics*, 362:229–271, 1998.

[5] J. Banchetti, P. Luchini, and M. Quadrio. Turbulent drag reduction over curved walls. *Journal of Fluid Mechanics*, 896:A10, 2020.

[6] I. Hadade, F. Wang, M. Carnevale, and L. di Mare. Some useful optimisations for unstructured computational fluid dynamics codes on multicore and manycore architectures. *Computer Physics Communications*, 235:305–323, 2019.

[7] J. P. Boris. On large eddy simulation using subgrid turbulence models. In J. L. Lumley, editor, *Whither Turbulence? Turbulence at the Crossroads*, volume 357, pages 344–353. Springer Berlin Heidelberg, Berlin, Heidelberg, 1989. Series Title: Lecture Notes in Physics.

[8] J. P. Boris, F. F. Grinstein, E. S. Oran, and R. L. Kolbe. New insights into large eddy simulation. *Fluid Dynamics Research*, 10(4-6):199–228, 1992.

[9] P. G. Huang, G. N. Coleman, and P. Bradshaw. Compressible turbulent channel flows: DNS results and modelling. *Journal of Fluid Mechanics*, 305:185–218, December 1995. Publisher: Cambridge University Press.

[10] N. D. Sandham, Y. F. Yao, and A. A. Lawal. Large-eddy simulation of transonic

turbulent flow over a bump. *International Journal of Heat and Fluid Flow*, 24(4):584–595, August 2003.

[11] S. B. Pope. *Turbulent flows*. Cambridge University Press, Cambridge ; New York, 2000.

[12] U. Piomelli. Large-eddy simulation: achievements and challenges. *Progress in Aerospace Sciences*, 35(4):335–362, 1999.

[13] S. Huang and Q. S. Li. A new dynamic one-equation subgrid-scale model for large eddy simulations. *International Journal for Numerical Methods in Engineering*, pages 835–865, 2009.

[14] R. S. Rogallo and P. Moin. Numerical simulation of turbulent flows. *Annual Review of Fluid Mechanics*, 16:99–137, 1984. ADS Bibcode: 1984AnRFM..16...99R.

[15] DNS Database of Wall Turbulence and Heat Transfer at Tokyo University of Science. `http://www.rs.tus.ac.jp/t2lab/db/`.

[16] H.G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object orientated techniques. *Computers in Physics*, 12:620–631, 1998.

[17] P. L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357–372, 1981.

[18] B. van Leer. Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method. *Journal of Computational Physics*, 32(1):101–136, 1979.

[19] C. Hirsch. *Numerical computation of internal and external flows: fundamentals of computational fluid dynamics*. Elsevier/Butterworth-Heinemann, Oxford ; Burlington, MA, 2nd ed edition, 2007.

[20] D. Mavriplis. Revisiting the Least-Squares Procedure for Gradient Reconstruction on Unstructured Meshes. In *16th AIAA Computational Fluid Dynamics Conference*, Orlando, Florida, 2003. American Institute of Aeronautics and Astronautics.

[21] A. E. Bondarev and A. E. Kuvshinnikov. Analysis of the Accuracy of OpenFOAM Solvers for the Problem of Supersonic Flow Around a Cone. In Y. Shi, H. Fu, Y. Tian, V. V. Krzhizhanovskaya, M. H. Lees, J. Dongarra, and P. M. A. Sloot, editors, *Computational Science – ICCS 2018*, volume 10862, pages 221–230. Springer International Publishing, Cham, 2018. Series Title: Lecture Notes in Computer Science.

[22] S. V. Patankar and D. B. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat and Mass Transfer*, 15(10):1787–1806, 1972.

[23] R. I. Issa. Solution of the implicitly discretised fluid flow equations by operator-splitting. *Journal of Computational Physics*, 62(1):40–65, 1986.

[24] E. de Villiers. The Potential of Large Eddy Simulation for the Modelling of Wall Bounded Flows. page 377.

[25] H. Jasak. Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows. page 396.

[26] H. Jasak. Error analysis and estimation for the finite volume method with applications to fluid flows. 1996. Accepted: 2011-10-10T13:39:40Z Publisher: Imperial College London (University of London).

[27] M. Marquillie, J.P. Laval, and R. Dolganov. Direct numerical simulation of a separated channel flow with a smooth profile. *Journal of Turbulence*, 9:1–23, 2008.

[28] M. Quadrio and P. Luchini. Integral space–time scales in turbulent wall flows. *Physics of Fluids*, 15(8):2219–2227, 2003.

[29] E. Komen, A. Shams, L. Camilo, and B. Koren. Quasi-DNS capabilities of OpenFOAM for different mesh types. *Computers & Fluids*, 96:87–104, 2014.

[30] R. Poletto, T. Craft, and A. Revell. A New Divergence Free Synthetic Eddy Method for the Reproduction of Inlet Flow Conditions for LES. *Flow, Turbulence and Combustion*, 91:1–21, 2013.

[31] I. Celik. RANS/LES/DES/DNS: The Future Prospects of Turbulence Modeling. *Journal of Fluids Engineering*, 127(5):829–830, 2005.

[32] turbulence fileserver. `http://turbulence.oden.utexas.edu`.

[33] C. Geuzaine and J.F. Remacle. Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities.

[34] J. Kim, P. Moin, and R. Moser. Turbulence statistics in fully developed channel flow at low Reynolds number. *Journal of Fluid Mechanics*, 177:133–166, April 1987.

[35] A. Yoshizawa and K. Horiuti. A Statistically-Derived Subgrid-Scale Kinetic Energy Model for the Large-Eddy Simulation of Turbulent Flows. *Journal of The Physical Society of Japan - J PHYS SOC JPN*, 54:2834–2839, 1985.

[36] T. Saad, D. Cline, R. Stoll, and J. C. Sutherland. Scalable Tools for Generating Synthetic Isotropic Turbulence with Arbitrary Spectra. *AIAA Journal*, 55(1):327–331, 2017.

A | Appendix A

This appendix reports the implementation in the OpenFOAM source code of the *functionObjects* developed to compute run-time, and eventually in post-processing, the following turbulent fields: turbulent kinetic energy production P, resolved turbulent kinetic energy k_{res}, SGS dissipation ϵ_{SGS}, viscous dissipation ϵ_{visc}.

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

bool Foam::functionObjects::tkeProd::calc()
{
    if (foundObject<volVectorField>(fieldName_))
    {
     if (foundObject<volVectorField>("UMean", false))
     {
      const volVectorField& UMean =  lookupObject<volVectorField>("UMean");
  const volVectorField& U = lookupObject<volVectorField>(fieldName_);
  const volVectorField UPrime = (U - UMean);
   return store
            (
             resultName_,
  -(UPrime)*(UPrime) && fvc::grad(U)
            );
    }
    else
    {
     const volVectorField& U = lookupObject<volVectorField>(fieldName_);
  volVectorField UMean
     (
      IOobject
      (
       "UMean",
       obr_.time().timeName(),
       obr_,
       IOobject::MUST_READ,
       IOobject::AUTO_WRITE
      ),
      U.mesh()
     );
  const volVectorField UPrime = (U - UMean);
   return store
            (
             resultName_,
  -(UPrime)*(UPrime) && fvc::grad(U)
            );
    }
   }

    return false;
}
```

Figure A.1: Implementation of *functionObject* P, which calculates the instantaneous field of the turbulent kinetic energy production P

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

bool Foam::functionObjects::kRes::calc()
{
 //In questo caso fieldName_ corrisponde a "U"
    if (foundObject<volVectorField>(fieldName_))
    {

     if (foundObject<volVectorField>("UMean", false))
     {
      const volVectorField& UMean =  lookupObject<volVectorField>("UMean");
  const volVectorField& U = lookupObject<volVectorField>(fieldName_);
  const volVectorField UPrime = (U - UMean);

   return store
            (
             resultName_,
  0.5*(UPrime & UPrime)
            );
    }
    else
    {
     const volVectorField& U = lookupObject<volVectorField>(fieldName_);

  volVectorField UMean
      (
       IOobject
       (
        "UMean",
        obr_.time().timeName(),
        obr_,
        IOobject::MUST_READ,
        IOobject::AUTO_WRITE
       ),
       U.mesh()
      );

  const volVectorField UPrime = (U - UMean);

   return store
            (
             resultName_,
  0.5*(UPrime & UPrime)
            );
    }
   }
   return false;
}


// * * * * * * * * * * * * * * * * * * Constructors  * * * * * * * * * * * * * * * //
```

Figure A.2: Implementation of *functionObject kRes*, which calculates the instantaneous field of the resolved turbulent kinetic energy k_{res}

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

bool Foam::functionObjects::SGSDiss::calc()
{
 //In questo caso fieldName_ corrisponde a "U"
    if (foundObject<volVectorField>(fieldName_))
    {
     const word turbModelName = Foam::turbulenceModel::propertiesName;

     if (foundObject<volVectorField>("UMean", false))
     {
      const volVectorField& UMean =  lookupObject<volVectorField>("UMean");
      const volVectorField& U = lookupObject<volVectorField>(fieldName_);
  const volVectorField UPrime = (U - UMean);
  const volScalarField& rho = lookupObject<volScalarField>("rho");
  const volScalarField nuSGS = (lookupObject<compressible::turbulenceModel>(turbModelName).mut()) / (rho);
  const volSymmTensorField B = -2.0*nuSGS*symm(fvc::grad(U));
  const volSymmTensorField SGSstrainTensor = symm(fvc::grad(UPrime));

  return store
            (
             resultName_,
   B && SGSstrainTensor
            );
    }
    else
    {
     const volVectorField& U = lookupObject<volVectorField>(fieldName_);

     volVectorField UMean
     (
       IOobject
      (
       "UMean",
       obr_.time().timeName(),
       obr_,
       IOobject::MUST_READ,
       IOobject::AUTO_WRITE
      ),
      U.mesh()
     );

  const volVectorField UPrime = (U - UMean);
  const volScalarField& rho = lookupObject<volScalarField>("rho");
  const volScalarField nuSGS = (lookupObject<compressible::turbulenceModel>(turbModelName).mut()) / (rho);
  const volSymmTensorField B = -2.0*nuSGS*symm(fvc::grad(U));
  const volSymmTensorField SGSstrainTensor = symm(fvc::grad(UPrime));

  return store
              (
             resultName_,
   B && SGSstrainTensor
            );
    }
   }

    return false;
}
```

Figure A.3: Implementation of *functionObject SGSDiss*, which calculates the instantaneous field of the SGS dissipation ϵ_{SGS}

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

bool Foam::functionObjects::viscDiss::calc()
{
    if (foundObject<volVectorField>(fieldName_))
    {
     const word turbModelName = Foam::turbulenceModel::propertiesName;

     if (foundObject<volVectorField>("UMean", false))
     {
      const volVectorField& UMean =  lookupObject<volVectorField>("UMean");
      const volVectorField& U = lookupObject<volVectorField>(fieldName_);
  const volVectorField UPrime = (U - UMean);
  const volScalarField& rho = lookupObject<volScalarField>("rho");
  const volScalarField nuLam   = 1.85*1e-5 / (rho);
  const volSymmTensorField SGSstrainTensor = symm(fvc::grad(UPrime));

  return store
            (
             resultName_,
   -2.0*nuLam*(SGSstrainTensor && SGSstrainTensor)
            );
    }
    else
    {
     const volVectorField& U = lookupObject<volVectorField>(fieldName_);

     volVectorField UMean
     (
       IOobject
      (
       "UMean",
       obr_.time().timeName(),
       obr_,
       IOobject::MUST_READ,
       IOobject::AUTO_WRITE
      ),
      U.mesh()
     );

  const volVectorField UPrime = (U - UMean);
  const volScalarField& rho = lookupObject<volScalarField>("rho");
  const volScalarField nuLam   = 1.85*1e-5 / (rho);
  const volSymmTensorField SGSstrainTensor = symm(fvc::grad(UPrime));

  return store
              (
             resultName_,
   -2.0*nuLam*(SGSstrainTensor && SGSstrainTensor)
            );
    }
   }
   return false;
}
```

Figure A.4: Implementation of *functionObject viscDiss*, which calculates the instantaneous field of the viscous dissipation ϵ_{visc}

List of Figures

List of Tables

Listings

Acknowledgements

I offer my most heartfelt and sincere gratitude to my family, Ugo, Luisa and Carlo, who have always been there for me and supported me in every possible way, helping me in difficult times and celebrating with me in the most joyful ones.
I thank my fiancée, Arianna, for putting up with me but also supporting me always and in any case, despite the distance and tough times.

I thank my friends Ing. Edoardo Venti and Ing. Matteo Zemello for accompanying me through these years of study, struggle and projects that seemed endless. Directly and indirectly they have spurred me to give my best and aim higher and higher.

I would also like to thank Professor Maurizio Quadrio and Dr. PhD. Mauro Carnevale for giving me the opportunity to participate in this thesis and live all the experiences that have shaped me during this past year.
A special thanks goes to Giove De Cosmo, a PhD student at the University of Bath, who is ultra-prepared and without whom, it must be said, I probably would never have graduated.

There are so many people I have not named but who have been close to me in this long journey, a big thank you goes to all of them.

The work has been performed under the Project HPC-EUROPA3 (INFRAIA-2016-1-730897), with the support of the EC Research Innovation Action under the H2020 Programme; in particular, the author gratefully acknowledges the support of Dr Mauro Carnevale of the Department of Mechanical Engineering, University of Bath, the Phd student Giove De Cosmo and the computer resources and technical support provided by EPCC.