Graz University of Technology
Institute of Hydraulic Engineering and Water Resources Management

Florian LORA, BSc

Implementation of fractional step algorithm with Runge-Kutta method in OpenFOAM for Large Eddy Simulations in hydraulic engineering

Master's Thesis

to achieve the university degree of
Diplom-Ingenieur
Master's programme Civil Engineering – Geotechnical and Hydraulic Engineering

submitted to
Graz University of Technology

Supervisor:
Josef SCHNEIDER, Assoc.Prof. Dipl.-Ing. Dr.nat.techn.

Assisting adviser:
Shervin SHAHRIARI, M.Sc.

Graz, March 2019

Affidavit

I declare that I have authored this thesis independently, that I have not used anything other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or contextually from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Date	Signature

Acknowledgement

Special gratitude belongs to my friend and former colleague Shervin Shahriari for piquing my interest especially in numerical methods and providing me with this interesting topic. He also provided very valuable and excellent support for the work on this topic. Furthermore I want to express my gratitude to all people I shared my time at the institute and I was able to have very interesting discussions, especially my supervisor Prof. Josef Schneider. Additionally I want to thank everybody who accompanied me throughout my studies.

I am very grateful to my family for their support and patience during my studies.

Abstract

Over the past few decades, numerical methods have become one of the main tools for the study and design of hydraulic structures. In recent years, due to the increase of computational power on the one hand and demand for accurate and better understanding of complex engineering flows on the other hand, Large Eddy Simulation (LES) has become a valuable tool for the study of complex hydraulic engineering flow situations. However, this method requires significantly large computational power in comparison with more common techniques like Reynolds-averaged Navier-Stokes (RANS) methods. Furthermore, the numerical diffusion must be minimised in LES. Higher-order explicit time integration like Runge-Kutta (RK) methods along with the fractional step methods are shown to be low-dissipative and, dependent on the method, computationally less demanding than the more common implicit second order time integration and the standard pressure correction approach like PISO (Pressure Implicit with Splitting of Operators). In this study, two solvers based on explicit third and fourth order RK methods with fractional step (projection) methods for incompressible flows are described and implemented in OpenFOAM. The new solvers are validated by the classical turbulent channel flow case and then for turbulent flow over a backward-facing step. In conclusion, the computational demand of the new solver with fourth order RK method measures only 77 % in comparison with PISO algorithms and the demand of the third order algorithm is even lower at only 60 % of the demand of PISO. Extending OpenFOAM with new custom code is easy and straightforward due to its high level syntax.

Kurzfassung

Numerische Methoden entwickelten sich im Laufe der letzten Jahrzehnte zu einem der wichtigsten Hilfsmittel zum Studium und Entwurf von wasserbaulichen Anlagen. In den vergangenen Jahren wurde Large Eddy Simulation, einerseits durch die Zunahme der verfügbaren Rechenleistung und andererseits durch den Bedarf an einem genaueren und besseren Verständnis von komplexen Strömungen, zu einem wertvollen Werkzeug zur Untersuchung komplexer hydraulischer Strömungssituationen. Diese Methode erfordert jedoch beträchtlich höheren Rechenaufwand im Vergleich zu gebräuchlichen Methoden wie RANS. Außerdem erfordert LES eine Minimierung der Numerischen Diffusion. Explizite Zeitintegration höherer Ordnung, wie Runge-Kutta-Methoden zusammen mit der Fractional-Step-Methode, zeigt sich als gering dissipativ und, abhängig von der Methode, als weniger Rechenleistung fordernd als die verbreitete implizite Zeitintegration zweiter Ordnung in Verbindung mit dem Standardansatz der Druckkorrektur wie zum Beispiel PISO oder SIMPLE. In dieser Arbeit werden zwei Lösungsverfahren basierend auf expliziter Runge-Kutta-Methode dritter und vierter Ordnung mit Fractional-Step (Projektions)-Methode für inkompressible Strömungen beschrieben und in OpenFOAM implementiert. Die neuen Algorithmen werden anhand der klassischen turbulenten Gerinneströmung und anhand des bekannten Falles turbulenter Strömung über eine plötzliche Querschnittsweitung validiert. Zusammenfassend beträgt der Rechenaufwand des neuen Verfahrens vierter Ordnung nur 77 % im Vergleich zu PISO und der Algorithmus dritter Ordnung erfordert nur 60 % der Rechenleistung für PISO. Die Erweiterung von OpenFOAM mit neuem benutzerspezifischen Programmcode ist aufgrund der hochentwickelten Syntax leicht und unkompliziert.

Contents

List of Figures

List of Tables

1 Introduction

Water is the most important resource on earth. Life on earth had its origin in the bodies of water millions of years ago and the existence of liquid water is known as the fundamental condition for the development of life. Current estimations Shiklomanov, 1998 quantify the total amount of water that is stored in Earth's hydrosphere with 1,386 million cubic kilometres. This is all the free water existing in the atmosphere, on Earth's surface and in the upper 2,000 metres of Earth's crust. Another large amount of water is bond in the crystalline structure of minerals. The majority of the free water fills oceans and only 2.5 % are fresh water. A portion of 68.7 % of the fresh water is frozen in the polar ice caps and glaciers and 30 % of the fresh water exists as ground water. The importance of water is given across all disciplines around the world:

- Water is vital to all known forms of life on Earth. A human needs up to seven litres of water a day in order not to suffer from dehydration.
- Water plays a major role in our climate (ice caps, ocean currents, rain seasons, ...) and weather (evaporation, precipitation, run off, formation of clouds, ...).
- Geology is also strongly influenced by the presence of water. A large amount of water is stored in fractures in rock and within pores in soil or even bond in the mineral structure. Many geologic processes are dependent on water, so does water lower the melting point of rock in subduction zones. Water is also one major actor in the process of erosion and sediment transport.
- Water has a major role in today's world economy. It is irreplaceable for the production of food (agriculture, fishing, food processing, ...) and is an excellent solvent for many chemicals used in pharmaceutical and chemical engineering. Industry and households use water for their heating and cooling systems. The world wide transportation systems strongly depends on water as shipping is by far the most used transportation system for the transportation of goods between continents. Last, water is very central to many leisure activities, such a swimming, sailing and many others and it is the reason for spending holidays by the sea.

The importance of water was already recognised by ancient civilisations. With the emergence of early agricultural techniques about 7,500 BC in the Near East, soon the first irrigation structures had been build. About 4,000 BC the first large scale irrigation systems have been developed. As settlements started to grow in areas of arid to semi-arid climate, ancient civilisations in the region of today's Iraq, Iran and Saudi-Arabia started building more sophisticated irrigations systems, the so-called Qanats. Qanats are gently sloped tunnels for the transportation of water from a spring or a well in mountainous region to fertile areas and settlements in valleys without any large river and the aquifer to deep for the construction of wells. Usually Qanats are less than five kilometres in length but some are recorded to measure up to 70 km in length. Typically Qanats are constructed about 20 to 200 metres deep in the ground. The inclination of such a tunnel is usually 1 : 1,000 and even lower with increasing length. The precision and sophisticated design indicate a very high level of engineering work. Also the ancient Greeks used this techniques to transport water from the spring across a mountain into the settlement.

Furthermore the flow of water was used to measure time. The earliest documented so-called water clocks had been used about 1,500 BC in Egypt and the ancient Mesopotamia.
The science of construction and design of structures for the transportation of water, such as Qanats, is called hydraulic engineering in modern times. Hydraulic engineering is the discipline of civil engineering that studies the flow of water utilising the principles of fluid mechanics. Generally three different techniques are available for the study of water in motion: in-situ measurements, model tests and numerical models. In-situ measurements are the measuring and recording of the flow situation in nature at the existing structure or river. This is often used to document the current state of the situation and prove afterwards the success of the implemented measures. Because situations in nature can not be controlled or reproduced identically and it is impossible or at least very expensive to investigate effects of future implementations the problem is often scaled down and the investigated in laboratories a smaller copy – a model. In laboratories the conditions can be controlled and repeated multiple times with and this allows the investigation of different situations just one after another. In numerical models the situation is described mathematically and the problem is simulated in any scale. In model tests and numerical models different alternative solutions for a problem can be tested or the influence of changing parameters can be determined.
The fields of application of this three methods are not only the study of the flow and the design of structures, but they are also used for the investigation of river systems and sediment transport as well as the study and design of sewer and water supply systems. The increase in computational power and the advances in modelling in recent years led to an increase in popularity of numerical models.
However, in hydraulic engineering the flow is of turbulent nature most of the time. Considering turbulence in numerical models is a challenge next to the problem of solving the general governing equations of fluid flow. As turbulence is essential to the flow in engineering a lot of effort is spend on how to consider turbulence. Over the past decades three general techniques have been established which deal with the phenomena of turbulence:

- Reynolds Averaged Navier Stokes (RANS) methods
- Large Eddy Simulations (LES)
- Direct Numerical Simulation (DNS)

RANS methods do not compute turbulent motion at all but include the effect of turbulence with the help of a model for large and small eddies. In contrast, DNS solves the governing equations without introducing models and simulates all scales of turbulent motion. LES is located in between RANS methods and DNS. In LES large turbulent motion is simulated and small eddies are modelled. All of this techniques have their advantages and disadvantages. Among the advantages of LES are the following:

- Large Eddies, which contain most of the energy, are simulated. They are problem dependent and control the dynamics of the flow. Modelling this large eddies causes most difficulties in RANS methods and makes a general RANS turbulence model impossible so far.

- Small turbulent scales, which only contain little energy, are modelled in LES. This scales are generally more universal and homogeneous than large eddies and therefore small eddies are much easier to model.
- Large Eddy Simulations are able to resolve transient turbulent effects which are of importance as these effects may induce vibrations and harmonic effects in structures. A tragic but well-known example is the Takoma bridge accident in 19xx.
- LES results in a better picture of the turbulence than RANS, especially when the problem includes recirculation zones, vortex structures and mixture of different phases.

However, there are some significant disadvantages of Large Eddy simulations:

- In comparison to DNS, LES require some modelling which can be difficult in vicinity to boundaries.
- In comparison to RANS methods, LES requires three-dimensional and transient simulations. Also a higher spatial and temporal resolution is required to capture all the energy-containing eddies. This points increase the numerical difficulty of the problem significantly and more accurate numerical methods are required to manage the computational effort, although the computational effort of LES will still be demanding. Also practical problems evolve in the storage and analysis of the large data sets retrieved from LES.

Exponentially increasing computational power in the past decade from one side and the success of LES for studying hydro-environmental flows from the other side lead to an increasing usage of this technique in hydraulic engineering problems. Recently, two papers have been published to discuss the current status as well as the challenges and opportunities of the LES method in hydraulic engineering. Stoesser (2014) summarised recent applications of LES in hydraulic and hydro-environmental research. According to Stoesser (2014), in a period from 1st of January 2009 to 31st of December 2013, a total number of 50 LES papers had been published in hydraulic journals. This demonstrates that not only LES is a well-established method for carrying out hydraulic related studies, also, it is expected that the number of papers increases in future. In another paper by Sotiropoulos (2015), the author highlighted the potential of simulation-based engineering science and discussed major computational challenges that lie ahead.

Aim of the study

Large Eddy Simulations usually demand large computational power and therefore LES studies are very costly. Simulations of high Reynolds number flows in complex geometries require a large number of CPU cores and it is not unusual that this computations take days and months. Therefore, if there is any method able to reduce the required computational time, it is desirable as it can lower the cost of LES significantly. Commonly used algorithms are based on SIMPLE or PISO. They are good, stable and robust solvers and well-established for many applications. Whereas steady flow problems can be solved with these implicit solvers at CFL numbers greater than one, unsteady flow situations like in LES require the CFL number to be smaller than one and the major advantage of the implicit solvers vanishes. This Courant-Friedrichs-Lewy condition states that the distance any flow information travels during a time step must be smaller than

the cell size. So does a flow particle only move from one cell to another and not pass through multiple cells at on time step. Any increase of the time steps will decrease the accuracy of the solution. Existing PISO and SIMPLE algorithms are using first and second order implicit time integration. Such low order methods are shown to have significant dissipative properties. However, LES require higher order methods to resolve unsteady vortices properly and to minimise the influence of numerical diffusion and dispersion. Additionally the computational cost to achieve the same level of accuracy is lower for higher order methods than for low order methods. Vuorinen et al. (2014) introduced Runge-Kutta projection methods for time-dependent flows. They claim a lack of low-dissipative methods for the use in CFD and describe therefore a new low-dissipative method and provide a guide for practical implementation. Vuorinen et al. (2014) summarise the advantages of their methods over PISO as following: (1) a computational speed-up up to 60 % can be achieved, (2) the method shows a low level of numerical dissipation and (3) the implementation is easy and straightforward due to the absence of iteration loops.
The objectives of the thesis are to (1) present the theory of LES and fractional step algorithms with Runge-Kutta method following the approach by Vuorinen et al. (2014), (2) to illustrate the implementation of two new algorithms with third and fourth order RK-method in OpenFOAM for LES in hydraulic engineering and to (3) validate the solvers for the classical turbulent channel flow case as well as the turbulent flow over a backward-facing step case.

Methodology

This thesis starts with the basic theory, continues with more detailed theory preparing for the implementation and validation of new solvers and finally summarises the achievements. At first research is done on OpenFOAM itself and general information is summarised in Chapter 2. This Chapter also contains a brief introduction to the functionality and structure of OpenFOAM and a short explanation how to set up any simulation is given. In Chapter 3 the theory of fluid mechanics is presented. Beginning with the governing equations and the finite volume discretisation it is then continued with methods for solving the transport equations of fluid flow. The method of PISO is described shortly and followed by an comprehensive explanation on the newly implemented fractional step algorithms with Runge-Kutta method. The theory of turbulence modelling and Large Eddy Simulation is described before the explanation of the WALE SGS-model closes Chapter 3. In Chapter 4 it is dealt with the implementation of the algorithms presented in Chapter 3 and provides a guide for the implementation of custom solvers to OpenFOAM. For Chapter 5, numerous Large Eddy Simulations are carried out to test the new solvers. First the solvers are tested on the classical turbulent channel flow case and last for turbulent flow over a backward-facing step. For each case a detailed description of the situation is given and the simulation set-up is presented. This is followed by the presentation, comparison and discussion of the results. The case definitions to run the simulations in OpenFOAM can be found in Appendix A. In Chapter 6 the most important achievements of this study are summarised and an outlook on future research topics is given.

2 Introduction to OpenFOAM

OpenFOAM is the leading free open source software for computational fluid dynamics (CFD). It is owned by the OpenFOAM Foundation and is distributed under General Public Licence (GPL).
The software Open Source Field Operation and Manipulation (introduced as OpenFOAM) is a collection of C++ libraries providing a framework for the development of application executables. OpenFOAM is delivered with approximate 250 pre-built applications which can be divided into two categories: solvers, that are designed for a specific problem and cover a wide range in fluid dynamics; and utilities, that perform data manipulation tasks. OpenFOAM provides a variety of pre-built pre- and post-processing environments as meshing and sampling tools.
Users can extend the collection of libraries and applications to fit their needs. New applications can be easily developed by using the packaged functionality within the C++ libraries.

2.1 History

- created by Henry Weller as 'FOAM' in 1989
- released open source as 'OpenFOAM' through OpenCFD Ltd., founded by Henry Weller, Chris Greenshields and Mattijs Janssens in December 2004
- In 2011 the software was transferred to the OpenFOAM Foundation and the trademark OpenFOAM® was licensed to the Foundation. Later OpenCFD was acquired by SGI Group.
- SGI Group sold OpenCFD to ESI Group in 2012
- since 2014 the development line (OpenFOAM-dev) is available for the public on GitHub (`github.com/OpenFOAM`)
- CFD Direct Ltd. was founded in March 2015 by Weller, Greenshields and Jenya Collings to maintain and develop OpenFOAM on behalf of the OpenFOAM Foundation. In June the Contribution Agreement was introduced to formalise the contributions by companies or individuals. CFD Direct hosts and maintains the official user guide for OpenFOAM.
- since 2016 OpenCFD Ltd., owned by ESI Group, directly releases twice a year an own version of OpenFOAM as OpenFOAM plus.
- OpenFOAM 5.0, the release by the OpenFOAM Foundation which is used for the implementation was published in July 2017. The latest version OpenFOAM 6.0 was released in July 2018.

2.2 Structure

Figure 2.1 below shows the overall structure of OpenFOAM. The interface to pre- and post-processing are themselves OpenFOAM utilities that ensures consistent data handling across different environments.

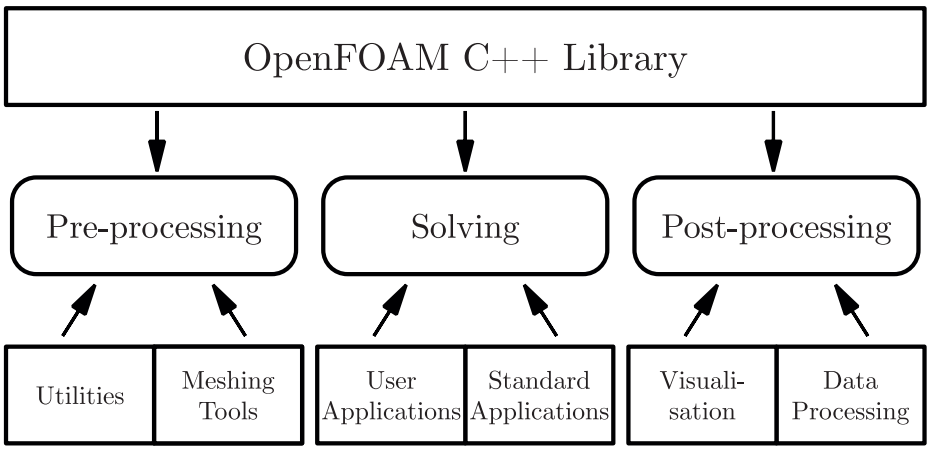

Figure 2.1: Overview of OpenFOAM structure (adapted from cfd.direct/openfoam/user-guide-v5/)

Each OpenFOAM case has its own directory in which all files and subdirectories are stored. These files and subdirectories are organised in a prescribed manner. This basis directory structure, containing the minimum set of files required to run an application, is shown in Figure 2.2. The user would assign a representative name to the case.

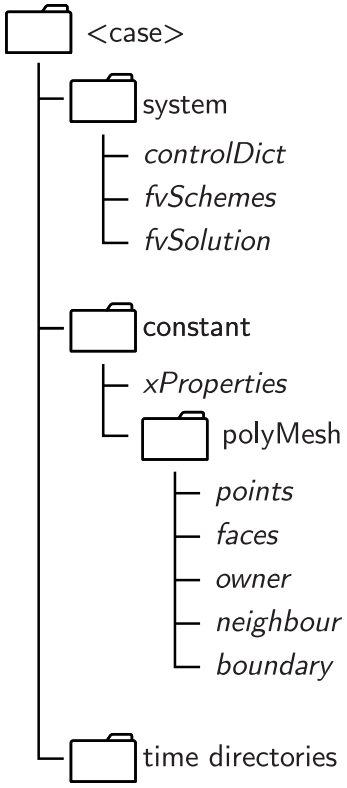

Figure 2.2: Case directory structure (adapted from cfd.direct/openfoam/user-guide-v5/)

The system directory contains parameters associated with the solution procedure itself. It contains a least the files controlDict, fvSchemes and fvSolution. The controlDict file sets run control parameters as start/end time, time step and parameters for data output. It can be modified while the application is running. fvSchemes specifies the used discretisation schemes and fvSolution defines equation solvers, tolerances and other algorithm controls.
The constant directory holds the mesh in the subdirectory polyMesh and files specifying physical properties for the problem e.g. transportProperties.
The time directories contain files of data for particular fields (e.g. pressure and velocity) at the corresponding time. The data can be: either initial values or boundary conditions that have to be specified; or results written by OpenFOAM. As the OpenFOAM fields must be initialised at least the time directory and files associated with the start time specified in the controlDict must exist.

3 Theoretical background

3.1 The transport equations

The governing equations of fluid flow are the starting point of all simulation methods for turbulent flow. For incompressible isothermal flow of Newtonian fluids these equations representing the conservation laws of mass and momentum in tensor notation are:

- conservation of mass (continuity equation):

$$\frac{\partial \mathrm{u_i}}{\partial \mathrm{x_i}} = 0 \tag{3.1}$$

- conservation of momentum (Navier-Stokes equation):

$$\underbrace{\frac{\partial \mathrm{u_i}}{\partial \mathrm{t}}}_{\text{temporal derivative}} + \underbrace{\frac{\partial \mathrm{u_i u_j}}{\partial \mathrm{x_j}}}_{\text{convection term}} = \underbrace{-\frac{1}{\rho}\frac{\partial \mathrm{p}}{\partial \mathrm{x_i}}}_{\text{pressure gradient}} + \underbrace{\nu \frac{\partial^2 \mathrm{u_i}}{\partial \mathrm{x_j}\,\partial \mathrm{x_j}}}_{\text{diffusion term}} \tag{3.2}$$

where $\mathrm{u_i}$ is the velocity component in $\mathrm{x_i}$ direction, p is the static pressure and ν represents the kinematic viscosity of the fluid. The equations 3.1 and 3.2 form a closed set and are describing all details of turbulent motion. In CFD generally three general approaches are used for turbulence modelling. First, DNS where the equations are solved without any turbulence model. Thus increasing the computational effort into excessive extends and making DNS not feasible for practical applications. Second, RANS methods introduce a model for all scales of turbulent motion which results in time-averaged fields. Last, in LES only small-scale turbulence is captured by models and larger motion is simulated. More on turbulence modelling can be found in Section 3.5.
The Navier-Stokes equation can be extended to account for additional momentum sources. This could be small but negligible contributions from the viscous stress term, external body forces, custom momentum sources or gravitational acceleration. In practice these additional sources can be neglected for many flow situations.

3.2 Finite volume discretisation

In order to reduce the required effort to solve flow problems it is aimed only to retrieve the solution on prescribed locations within the domain instead of all points in the continuous field. A discretisation is performed to transform partial differential equations (Equation 3.1 and 3.2) into a linear system of equations. The solution of the system of equations corresponds to the original solution in pre-determined locations in space and time. Any discretisation can be divided into two parts: discretisation of the domain and discretisation of the equations.
The discretisation of the domain can be understood as numerical description of the domain prescribing the locations in which the solution is obtained and defining the boundaries. The domain is divided into a finite number of discrete parts, named cells or control volumes. The time interval for time-dependent problems is split into a finite number of time steps. There are three approaches commonly used for the discretisation of equations:

- finite element method (FEM),

- finite difference method (FDM) and
- finite volume method (FVM).

The finite volume method, which is the basic principle of OpenFOAM, is based on the integral form of the governing equations, which implies the conservation of mass and momentum at the discrete level. In contrast to the finite element method the equations are solved in a fixed Cartesian coordinate system and the cells can be of any polyhedral shape. Systems of partial differential equations are solved separately with explicit coupling of the equations.
A more detailed explanation of the finite volume discretisation has been written by Jasak (1996, p. 73ff).

3.3 Solving the equations of fluid transport

The Equations 3.1 and 3.2 form a coupled system of equations which is almost impossible to solve analytically and still hard to solve numerically. These partial differential equations couple the main variables such as velocity and pressure to each other requiring the equations to be solved simultaneously. Additionally the non-linear convection term increases the difficulty in solving. Hence, a strong dependence on boundary conditions creates a different problem for each flow situation. Therefore some approximations and simplifications such as the linear viscosity law, which is already included in Equation 3.2, must be applied to make the problem solvable. This gives then four equations for four unknowns in the case of incompressible Newtonian fluids.
For solving these coupled system of non-linear partial differential equations different methods have been proposed in the past. A lot of frequently used methods belong to the group of projections methods which attempt to decouple the system of equations and solving the equations sequentially.

3.4 Projection methods

Projection methods are a class of procedures to solve the Navier-Stokes equation. They were originally introduced in the context of numerical simulations of fluids by Chorin (1967). The key advantage of projection methods is the decoupling of velocity and pressure which increases the efficiency of the computations. The algorithm of projection methods is based on the decomposition of the vector into a divergence-free and a curl-free vector field (Helmholtz decomposition). The method can be divided into two basic algorithm steps: First,in the predictor step, an intermediate velocity field is computed based on the momentum equation ignoring the pressure gradient. Second, in the projection step, the intermediate velocity is projected onto divergence-free space by using the pressure. Today solving algorithms based on the projection method can be split into two groups. First the group of SIMPLE-related solvers to which SIMPLE and PISO belong to and second the group of fractional step methods based on Chorin (1967).
In the following section the PISO method is introduced and afterwards a detailed explanation of fractional step methods is given.

3.4.1 The PISO method

The PISO (Pressure-Implicit with Splitting of Operators) method is one approach of a projection method. It was developed and published by Issa (1986). It is an extension of the SIMPLE algorithm which can deal with larger time steps and works more efficiently. The method can be divided into three steps: one predictor and two corrector steps. The PISO algorithm is organised in three steps:

- **Momentum predictor.** The momentum equation is solved with the pressure field from the previous time step. This gives an approximation of the new velocity field $\mathrm{u_i^*}$. In general $\mathrm{u_i^*}$ will not satisfy the zero-divergence condition.
- **First corrector step.** The predicted velocity $\mathrm{u_i^*}$ is used to obtain the pressure field $\mathrm{p^*}$. The updated pressure field is used to solve for the $\mathrm{u_i^{**}}$ field, which satisfies the zero-divergence condition.
- **Second corrector step.** A new velocity field $\mathrm{u_i^{***}}$ together with its corresponding pressure field $\mathrm{p^{**}}$ are computed similar to the first corrector step. Obviously additional corrector steps can be introduced but the accuracy with which $\mathrm{u_i^{***}}$ and $\mathrm{p^{**}}$ approximate the exact solution is sufficient for the most practical applications.

3.4.2 Fractional step methods with Runge-Kutta time scheme

Fractional step methods are based on the solving procedure proposed by Chorin (1967). Because of the absence of iterating steps such a solving algorithm is much easier to implement compared to SIMPLE methods. The solving procedure can be roughly organised in the following steps:

- **Predictor step.** The velocity field is advanced to a mid time step position by solving the momentum equation while neglecting the pressure term. This is followed by an intermediate projection step which enforces the divergence-free property of the velocity field.
- **Corrector step.** The velocity field advances to the full time step position using the mid time step estimates. The final projection ensures the divergence-free velocity field and thus giving us the pressure field. Finally all fields advanced to the new time.

The cornerstone of projection methods is the Helmholtz decomposition. The decomposition is the mathematical basis for decoupling velocity and pressure in incompressible flow problems. It states that any vector field can be expressed as the sum of a divergence-free and a curl-free part. The decomposition requires a pressure p that fulfils

$$\mathrm{u_i} = \mathrm{u_i^*} - \frac{\Delta t}{\rho}\frac{\partial \mathrm{p}}{\partial \mathrm{x_i}} \tag{3.3}$$

where $\dfrac{\partial \mathrm{u_i^*}}{\partial \mathrm{x_i}} \neq 0$, $\dfrac{\partial \mathrm{u_i}}{\partial \mathrm{x_i}} = 0$ (Equation 3.1) and $\epsilon_{\mathrm{ijk}}\dfrac{\partial^2}{\partial \mathrm{x_i}\partial \mathrm{x_j}}\mathrm{p} = 0$. Calculating the divergence of Equation 3.3 reveals that the pressure p must be a solution of the Poisson's equation

$$\frac{\partial^2 \mathrm{p}}{\partial \mathrm{x_j}\partial \mathrm{x_j}} = \frac{\rho}{\Delta t}\frac{\partial \mathrm{u_i^*}}{\partial \mathrm{x_i}} \tag{3.4}$$

For the discretisation in time different methods can be chosen. Possible candidates are Crank Nicolson Schemes, Euler methods or schemes that follow the Runge-Kutta method. Runge Kutta methods are commonly used in simulations of fluid flow and are available in different orders of accuracy. The simplest and of lowest order Runge-Kutta method is also known as forward Euler method. Explicit Runge-Kutta methods are often applied to solve any ordinary differential equations numerically. In general Runge-Kutta methods can be written in the form

$$y_{\mathrm{n}+1} = y_{\mathrm{n}} + \Delta t \sum_{\mathrm{i}=1}^{s} b_{\mathrm{i}} k_{\mathrm{i}} \tag{3.5}$$

with

$$k_{\mathrm{i}} = f\left(t_n + c_{\mathrm{i}}\Delta t, y_n + \Delta t \sum_{\mathrm{i,j}=1}^{s} a_{\mathrm{ij}} k_{\mathrm{j}}\right) \tag{3.6}$$

where a_{ij}, b_{i} and c_{i} are the coefficients from the corresponding Butcher tableau of the method.

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{ss} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$$

The Butcher tableau of explicit methods is lower triangular. The tableau for the fourth order Runge-Kutta method is given by

$$\begin{array}{c|cccc} 0 & 0 & & & \\ \frac{1}{2} & \frac{1}{2} & 0 & & \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

and for the third order method the coefficients are given by

$$\begin{array}{c|ccc} 0 & 0 & & \\ \frac{1}{2} & \frac{1}{2} & 0 & \\ 1 & -1 & 2 & 0 \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$$

The solution procedure of the newly implemented Runge-Kutta fractional step algorithms is illustrated as flowchart in Figure 3.1. The code implementation of the fractional step solver with Runge-Kutta methods is handled in Section 4.

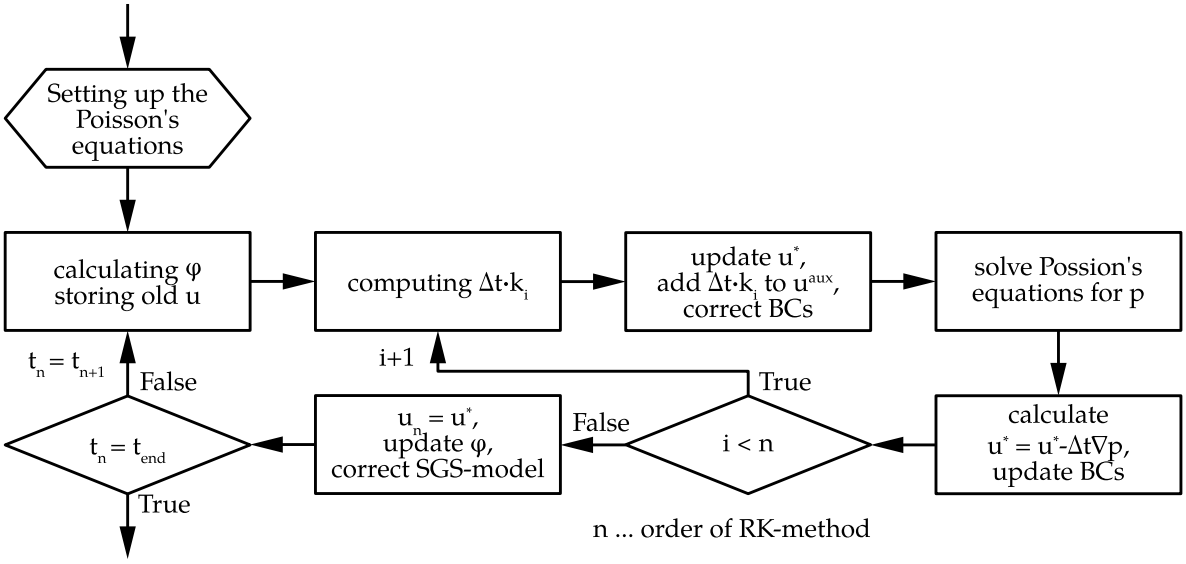

Figure 3.1: Flow diagram of solvers with Runge-Kutta algorithm (only the main solving routine is shown)

3.5 Large Eddy Simulation

The main principles of Large Eddy Simulation will be described briefly in this section. Comprehensive explanations of LES in hydraulic engineering can be found in Rodi, Constantinescu and Stoesser (2013).

All numerical methods dealing with the solution of fluid flow problems start from Equation 3.1 and 3.2. This equations form a closed set of equations describing incompressible fluid flow with all its fluctuations and turbulent motion. Direct numerical simulation (DNS) solves these equations directly without introducing models. Although we solve for discretised fields this method is not feasible for practical applications. Resolving eddies of all scales would be excessive especially at high Reynolds number. Hence, this methods field of application is restricted to research topics and low Reynolds numbers.

The opposite of DNS are Reynolds Average Navier Stokes (RANS) methods. Turbulent fluctuations and eddies are not resolved in this method and averaged out over time by filters. But several studies have shown that RANS methods are not able to reproduce complex flow situations such as the interaction between turbulence-dominated primary and secondary flow in curved channels or large anisotropic structures like recirculation zones. Therefore the generality of RANS methods is limited.

Results of RANS methods are time-averaged flow fields (flow quantities). As all fluctuations and eddies are filtered out but are still of importance for turbulent flows, a model has to be put in place. Such a model has to account for eddies of all scales and should be applicable to all practical situations. There has been effort to develop a general purpose RANS turbulence model over centuries but without any reliable result so far. This is caused by the wide range of different features of larger and smaller eddies listed in Table 3.1.

Small eddies at sufficient high Reynolds numbers are generally more isotropic and universal, hence they are easier to model. Whereas large scale turbulence depends on the geometry,

Table 3.1: Differences between large eddies and small scale turbulence (adapted from Rodi, Constantinescu and Stoesser, 2013, p. 15, Table 2.1)

Large eddies	Small scale turbulence
• produced by mean flow	• produced by large eddies
• depend on geometry and boundaries	• universal
• ordered	• random
• require deterministic description	• can be modelled statistically
• inhomogeneous	• homogeneous
• anisotropic	• isotropic
• long-living and energetic	• short living and non-energetic
• diffusive	• dissipative
► difficult to model	► easier to model
► general model impossible	► general model possible

boundary conditions and body forces. This makes general models for all scale turbulence in RANS methods dependent on the problem.
The essential idea of Large Eddy Simulation (LES) is the explicit computation of large scale motion for each flow situation. Small eddies are captured by models and practice has shown that simple models can be sufficient. Further the restriction to low Reynolds number problems inherited from DNS is removed and allows for investigating the dynamics of highly turbulent flows.

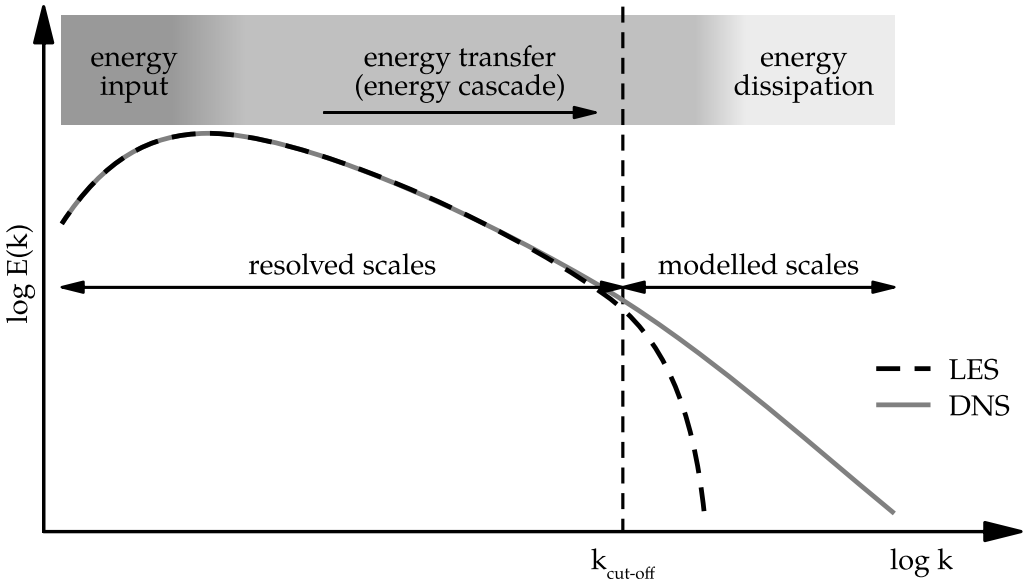

Figure 3.2: Concept of Large Eddy Simulation in context with energy content

The first step of the main principle of LES is the separation of the turbulent motion into large

and small eddies. The large scales interact with and extract energy from the mean flow and transfer energy in the energy cascade to smaller scales. The small scales withdraw kinetic energy through dissipation. The scale separation should ideally occur in the spectral region of energy transfer, so that all problem-dependent eddies are resolved and only the dissipative motion must be modelled. Figure 3.2 illustrates the idea of scale separation in relation to the energy content. In practice the problem size does not always allow for sufficient fine grids, hence LES resolves as much motion as one can afford but the resolved eddies should contain most of the energy.

In contrast to the time-averaging of RANS methods LES utilises spatial averaging or filtering to remove fluctuations. There are generally two approaches available: implicit filtering which is closely related to the volume balance approach of Schumann (1975) and explicit filtering which was first proposed by Leonard (1974). As the latter is the more general method it is used here to describe the process of filtering.

The flow quantities f are split into the resolved quantities $\bar{f}$ and unresolved fluctuations f'. For explicit filtering a normalised filter function with filter width Δ is needed. Commonly used filter functions are top-hat, Gaussian and spectral cut-off filter. The spectral cut-off works with Fourier transformation and removes all fluctuations above a specified wave number $\mathrm{k_{cut\text{-}off}}$, whereas the other two filter lead to an energy spectrum illustrated as LES in Figure 3.2. Increasing the filter width increases the amount of the removed fluctuations and filtered function becomes smoother. Filtering the governing equations 3.1 and 3.2 leads to the following filtered equations:

- continuity equation:

$$\frac{\partial \bar{\mathrm{u}}_\mathrm{i}}{\partial \mathrm{x}_\mathrm{i}} = 0 \tag{3.7}$$

- Navier-Stokes equation:

$$\frac{\partial \bar{\mathrm{u}}_\mathrm{i}}{\partial \mathrm{t}} + \frac{\partial \bar{\mathrm{u}}_\mathrm{i}\bar{\mathrm{u}}_\mathrm{j}}{\partial \mathrm{x}_\mathrm{j}} = -\frac{1}{\rho}\frac{\partial \bar{\mathrm{p}}}{\partial \mathrm{x}_\mathrm{i}} + \nu\frac{\partial^2 \bar{\mathrm{u}}_\mathrm{i}}{\partial \mathrm{x}_\mathrm{j}\,\partial \mathrm{x}_\mathrm{j}} - \frac{\partial \tau_\mathrm{ij}^{\mathrm{SGS}}}{\partial \mathrm{x}_\mathrm{j}} \tag{3.8}$$

The non-linear convective term $\mathrm{u_i u_j}$ in the Navier-Stokes Equation 3.2 originally results in the filtered $\overline{\mathrm{u_i u_j}}$. But we want to solve for the filtered velocity $\bar{\mathrm{u}}_\mathrm{i}$ so we want to express the convective term as $\bar{\mathrm{u}}_\mathrm{i}\bar{\mathrm{u}}_\mathrm{i}$. The difference

$$\tau_\mathrm{ij}^{\mathrm{SGS}} = \overline{\mathrm{u_i u_j}} - \bar{\mathrm{u}}_\mathrm{i}\bar{\mathrm{u}}_\mathrm{i} \tag{3.9}$$

represents the unresolved fluctuations which act like stresses and $\tau_\mathrm{ij}^{\mathrm{SGS}}$ is therefore called subfilter scale (SFS) stresses or more common subgrid scale (SGS) stresses. Detailed examination of Equation 3.9 in combination with $f = \bar{f} + f'$ shows three different terms within $\tau_\mathrm{ij}^{\mathrm{SGS}}$: Leonard stresses, cross stresses and LES Reynolds stresses. As the entire term of $\tau_\mathrm{ij}^{\mathrm{SGS}}$ appears in the filtered Navier-Stokes Equation 3.8 it needs to be modelled. Such a model is called SGS model and the particular one used in the present study is presented in Section 3.5.1.

With explicit filtering as described above the process of discretisation and filtering are separated. This allows for any filter width Δ larger than the mesh size and results in a continuous smoothed function $\bar{f}$. Hence, the solution obtained following Leonard's approach is independent of the discretisation.

However, fundamental studies are so far the only field of application for explicit filtering. In practice the filter width equals the mesh size and LES code does not contain any filter function. Instead the filtered Equations 3.7 and 3.8 are solved numerically on a certain grid. The main principles of FVM show a close relation to top-hat filtering with the same filter width. The quantities f are represented by the average over the control volume. As a consequence the filtered function $\bar{f}$ is no longer continuous. The scale separation depends now on the discretisation and therefore the obtained solution is dependent on the mesh and time-stepping. This approach is called implicit filtering and is implemented in most of the available LES code.
Implicit filtering is closely related to Schumann's approach, which starts from a finite volume mesh and the original Equations 3.1 and 3.2. As filtering and Schumann's approach are only symbolically different the latter is not described in this thesis and everyone interested is referred to relevant LES literature.
It should be mentioned here that OpenFOAM uses implicit top-hat filtering in LES and any filtering or filter width is implemented for the use with SGS-models.

3.5.1 WALE turbulence model

As already mentioned above subgrid scale stresses arise in the process of filtering. The SGS-stresses τ_{ij}^{SGS} represent the effect of unresolved turbulence on the large-scale motion. This effect is mainly dissipative. Therefore two different ideas exist to model SGS-stresses. One introduces an explicit model for τ_{ij}^{SGS}, the other withdraws energy through numerical dissipation by the solving algorithm. The latter is called Implicit Large Eddy Simulation (ILES) and is not further dealt with. The most common method in hydraulic engineering is the usage of explicit SGS-models.
Such a model should withdraw the right amount of energy from the flow and should give a physically realistic representation of the energy exchange within the resolved turbulent motion. However, the most important interaction a SGS-model has to account for happens between the largest subgrid scales and the smallest resolved scales. Furthermore, a well resolved Large Eddy Simulation achieves energy conservation with a proper SGS-model.
Generally the SGS-stresses consist of an anisotropic component τ_{ij} and an isotropic component.

$$\tau_{ij}^{SGS} = \tau_{ij} + \frac{1}{3}\tau_{kk}^{SGS}\delta_{ij} \tag{3.10}$$

The isotropic part contains normal stresses which act like pressure and is therefore added to the filtered pressure. This separation is convenient when relating τ_{ij} to the gradients of the velocity field including an eddy viscosity ν_t. The relationship for any eddy viscosity model is given through

$$\tau_{ij} = -2\nu_t \bar{S}_{ij} \tag{3.11}$$

with

$$\bar{S}_{ij} = \frac{1}{2}\left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) \tag{3.12}$$

which is the second invariant of the symmetric part of the velocity gradient tensor $\bar{g}_{ij}$ which is defined as

$$\bar{g}_{ij} = \frac{\partial \bar{u}_i}{\partial x_j} \tag{3.13}$$

.In consequence the determination of the eddy viscosity ν_t has become the main part for modelling. Such a eddy viscosity model can be generally written in the form

$$\nu_\mathrm{t} = \mathrm{C_m}\Delta^2\overline{OP}\,(\mathbf{u},\mathrm{t}) \tag{3.14}$$

where $\mathrm{C_m}$ is the model constant, Δ the characteristic subgrid length scale and $\overline{OP}$ an operator of space and time which represents a characteristic turbulent velocity.
The most popular SGS-model based on the eddy viscosity assumption was proposed by Smagorinsky (1963) and different modified versions are available today. Because of some disadvantages for the application in hydraulic engineering the WALE SGS-model is preferred in this study.
The Wall Adapting Local Eddy viscosity (WALE) model was introduced by Nicoud and Ducros (1999). It is based on a velocity gradient tensor invariant and is sensitive to the strain rate and rotational rate of small turbulent structures. Furthermore it shows correct scaling of ν_t close to the wall. The WALE model is described by the following equation:

$$\nu_\mathrm{t} = (\mathrm{C_w}\Delta)^2\,\frac{\left(S^d_\mathrm{ij}S^d_\mathrm{ij}\right)^{\frac{3}{2}}}{\left(\bar{S}_\mathrm{ij}\bar{S}_\mathrm{ij}\right)^{\frac{5}{2}} + \left(S^d_\mathrm{ij}S^d_\mathrm{ij}\right)^{\frac{5}{4}}} \tag{3.15}$$

with the traceless symmetric part of the square of the velocity gradient tensor S^d_ij defined as

$$S^d_\mathrm{ij} = \frac{1}{2}\left(\bar{\mathrm{g}}^2_\mathrm{ij} + \bar{\mathrm{g}}^2_\mathrm{ij}\right) - \frac{1}{3}\delta_\mathrm{ij}\bar{\mathrm{g}}^2_\mathrm{kk} \tag{3.16}$$

The model constant $\mathrm{C_w}$ was found through numerous separate investigations to suit best as $\mathrm{C_w} = 0.325$. This turbulence model is well suitable for complex geometries with structured or unstructured grid. A good choice for the characteristic length scale Δ would be $\Delta = \sqrt[3]{V}$ where V is the cell volume.

4 Implementation

OpenFOAM provides itself a high-level language which allows the implementation of custom applications, boundary conditions and much more with little effort. In order to speed up the implementation process an existing solver (pisoFoam or pimpleFoam is recommended) is selected as starting point. It is recommended to create an `application` directory within your personal project directory (`$FOAM_RUN`). First of all the entire directory of `pisoFoam` is copied from `$FOAM_SOLVERS/incompressible/` to your `application` directory and is renamed to the desired name (`rk3fracsFoam` and `rk4fracsFoam` respectively). It is highly recommended to name the directory in the same way as your source file.
In the next step the `./Make/` folder is updated to the new solver. As neither header files nor libraries outside of the current application directory are added the `options` file stays untouched. The `files` file has to be updated in the following way to match our new solver where `<APP_NAME>` is the specified application name.

```
<APP_NAME>.C

EXE = $(FOAM_USER_APPBIN)/<APP_NAME>
```

As it can be seen in Figure 3.1 some additional fields are needed for the computation. This fields have to be initialised therefore the following entry is added to the Header file `createFields.H` for each of the fields `Uold`, `Uc` and `dU`.

```
   volVectorField <FIELD_NAME>
   (
      IOobject
      (
         "<FIELD_NAME>",
         runTime.timeName(),
         mesh,
         IOobject::NO_READ,
         IOobject::NO_WRITE
      ),
      U
   );
```

The next step in the sequence of the algorithm is the Initialisation of the Poisson's matrix which is implemented as the constant Laplacian operator in the `pEqn.H` file. This file contains the following routine and is executed once before starting the iteration over time.

```
fvScalarMatrix pEqn
(
  fvm::laplacian(p)
);
pEqn.setReference(pRefCell, pRefValue);
```

The following code snippet is executed multiple times at each time step and is therefore extracted from the main source file and included as header file. The routine defined in `pCorrection.H` ensures the boundary conditions of u_i, solves the Poisson's equation and carries out the projection of the velocity into divergence-free space.

```
U.correctBoundaryConditions();
solve(pEqn == fvc::div(U)/runTime.deltaT());
#include "continuityErrs.H"
U = U - fvc::grad(p)*runTime.deltaT();
U.correctBoundaryConditions();
```

The above described implementation steps are universal to both implemented fractional step solvers. The two fractional step solvers with Runge-Kutta method differ only in the main source file (`*.C`).

4.1 rk4fracsFoam

The fractional step method with fourth order Runge-Kutta algorithm is implemented according to the flow diagram shown in Figure 3.1 and with the corresponding values from the Butcher tableau mentioned in Section 3.4.2. The `rk4fracsFoam.C` file is then modified by replacing the `pisoFoam` related code with the code for the new solver. At a first step the following line is added prior to the `while`-loop.

```
#include "pEqn.H"
```

It includes the code written in the `pEqn.H` file which is described above and executed once in the solving procedure. As the last step the PISO-related code within the `while`-loop is deleted and replaced by the following code which is the kernel of the `rk4fracsFoam` solver.

```
Uold=U; Uc=U;

phi = (fvc::interpolate(U) & mesh.Sf());
dU = runTime.deltaT()*(fvc::laplacian(turbulence->nuEff(),U) - fvc::div(phi,U) );
Uc = Uc + (1.0/6.0)*dU; U = Uold + 0.5*dU;
#include "pCorrection.H"

phi = (fvc::interpolate(U) & mesh.Sf());
dU = runTime.deltaT()*(fvc::laplacian(turbulence->nuEff(),U) - fvc::div(phi,U) );
Uc = Uc + (1.0/3.0)*dU; U = Uold + 0.5*dU;
#include "pCorrection.H"

phi = (fvc::interpolate(U) & mesh.Sf());
dU = runTime.deltaT()*(fvc::laplacian(turbulence->nuEff(),U) - fvc::div(phi,U) );
Uc = Uc + (1.0/3.0)*dU; U = Uold + dU;
#include "pCorrection.H"

phi = (fvc::interpolate(U) & mesh.Sf());
dU = runTime.deltaT()*(fvc::laplacian(turbulence->nuEff(),U) - fvc::div(phi,U) );
Uc = Uc + (1.0/6.0)*dU; U = Uc;
#include "pCorrection.H"

phi = (fvc::interpolate(U) & mesh.Sf());

turbulence->correct();
```

Finally the new solver is ready to be compiled using `wmake`. However, a detailed explanation how to compile OpenFOAM source code can be found on cfd.direct/openfoam/user-guide/ in Chapter 3.2 – Compiling applications & libraries.

4.2 rk3fracsFoam

The fractional step solver with third order Runge-Kutta method is implemented in the same way as the `rk4fracsFoam` solver. Therefore the `rk4fracsFoam`-related code is replaced by the specific code for the third order method and the file is itself renamed to `rk3fracsFoam.C`.

```
Uold=U; Uc=U;

phi = (fvc::interpolate(U) & mesh.Sf());
dU = runTime.deltaT()*(fvc::laplacian(turbulence->nuEff(),U) - fvc::div(phi,U) );
Uc = Uc + (1.0/6.0)*dU; U = Uold + 0.5*dU;
#include "pCorrection.H"

phi = (fvc::interpolate(U) & mesh.Sf());
dU = runTime.deltaT()*(fvc::laplacian(turbulence->nuEff(),U) - fvc::div(phi,U) );
Uc = Uc + (2.0/3.0)*dU; U = Uold + 2.0*(Uold - U) + 2.0*dU;
#include "pCorrection.H"

phi = (fvc::interpolate(U) & mesh.Sf());
dU = runTime.deltaT()*(fvc::laplacian(turbulence->nuEff(),U) - fvc::div(phi,U) );
Uc = Uc + (1.0/6.0)*dU; U = Uc;
#include "pCorrection.H"

phi = (fvc::interpolate(U) & mesh.Sf());

turbulence->correct();
```

5 Validation

For the validation of the fractional step solution algorithms with Runge-Kutta method (Sections 3 and 4) two classical cases of large eddy simulations are set up and a series of numerical simulations is carried out. The first case investigates turbulent channel flow and is validated with DNS data from the experiments of Moser, Kim and Mansour (1999). The second case deals with turbulent flow over a backward facing step for which validation data was created in DNS by Le, Moin and Kim (1997). The numerical simulations are performed with OpenFOAM-5.x.

5.1 Case one – turbulent channel flow

This case deals with turbulent flow in a straight channel. It follows the direct numerical simulations performed by Moser, Kim and Mansour (1999). This flow situation has been simulated by several others at different Reynolds numbers. The published data retrieved from the experiments by Moser, Kim and Mansour (1999) is used to validate the fractional step solvers. The flow simulations are carried out at Reynolds number $\mathrm{Re}_\tau = \mathrm{u}_\tau \delta/\nu = 180$, 395 and 590, where u_τ is the friction velocity, δ is the channel half-width and ν is the kinematic viscosity. This case is included as tutorial in OpenFOAM at $\mathrm{Re}_\tau = 395$. This tutorial case `channel395` is used as basis for the simulations. The flow computation with PISO algorithm at $\mathrm{Re}_\tau = 395$ is identical to this tutorial case, however the channel length is altered and it uses a different grid resolution.

5.1.1 Computational domain

The domain is a simple rectangular channel with the size of $5\delta \times 2\delta \times 2\delta$, where δ represents the channel half width. The length of the domain is altered in comparison to `channel395` allowing the mapped boundary inlet condition to be placed. The domain is illustrated in Figure 5.1 showing the reference coordinate system, dimensions and boundaries. The domain is discretised using the block grid method creating an unstructured mesh.

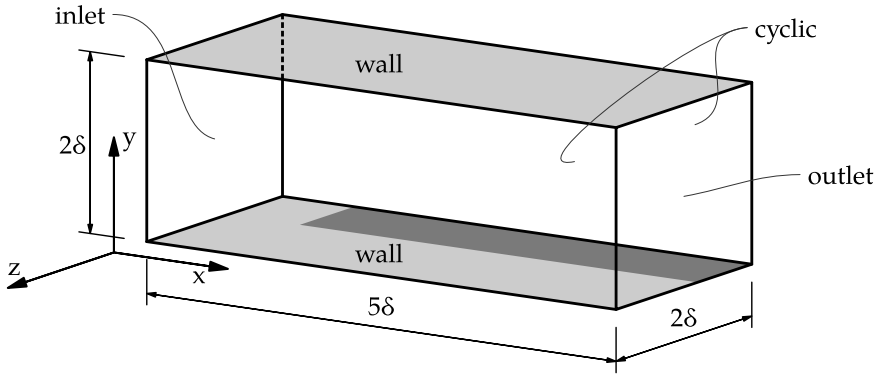

Figure 5.1: Geometry and dimensions of the channel domain

In a first step a grid refinement study is carried out for three different grid resolutions at $\mathrm{Re}_\tau = 395$ and 590. Table 5.1 lists the different models and the used resolution. Therefore L_i represents the size of the domain and N_i is the number of elements. The computational grid is kept unchanged for all simulations, however it is possible to adapt the mesh according to the flow

situation. Characteristic simulation parameters of all conducted simulations are listed in Table 5.2.

Table 5.1: Model parameters for the LES of turbulent channel flow

Model	L_x	L_z	$N_x \times N_y \times N_z$
M0.10			$50 \times 50 \times 30$
M0.07	5δ	2δ	$75 \times 76 \times 45$
M0.05			$100 \times 100 \times 60$

Figures 5.2 and 5.3 show the mean streamwise velocity profile $\mathrm{u}^+ = \mathrm{u}/\mathrm{u}_\tau$ retrieved from LES with the different models. The LES profiles are compared to the profiles of direct numerical simulation by Moser, Kim and Mansour (1999). The depicted profiles show a good convergence of the grid refinement. The LES velocity profile are at lower Reynolds number already close to the DNS profile whereas at the higher Reynolds number a larger difference is seen. The difference has different sources: Errors introduced by the numerical model as well as errors introduced by the way of computation of the velocity profile as u_τ is calculated numerically within the first cell off the wall. Additionally it is mentioned that the bulk velocity for Re_τ is given within the OpenFOAM tutorial case but has to be found in an iterative procedure for the other cases.

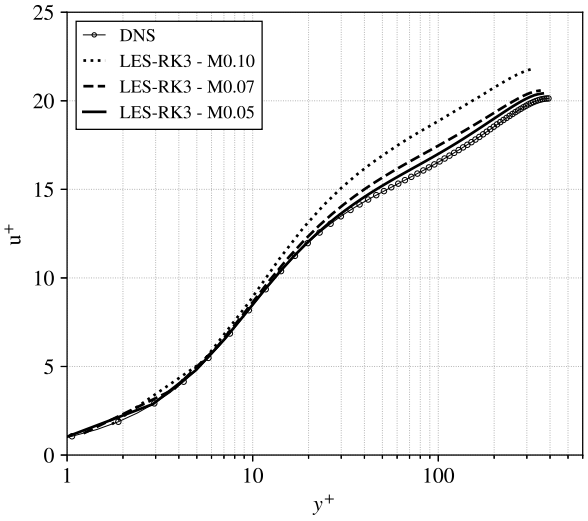

Figure 5.2: Mean velocity profile in wall coordinates for channel flow at $\mathrm{Re}_\tau = 395$ with grid refinement

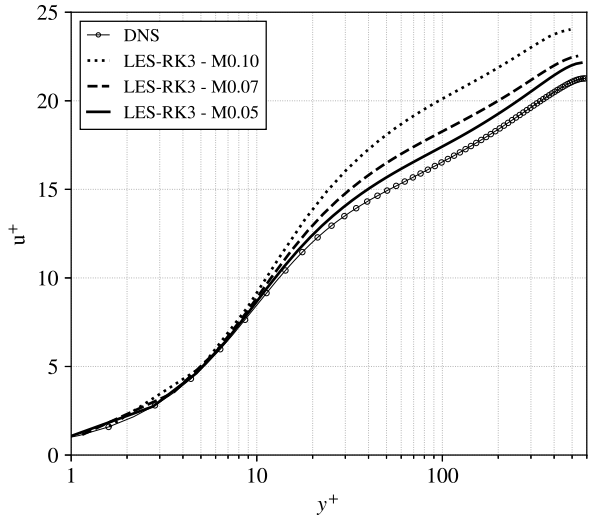

Figure 5.3: Mean velocity profile in wall coordinates for channel flow at $\mathrm{Re}_\tau = 590$ with grid refinement

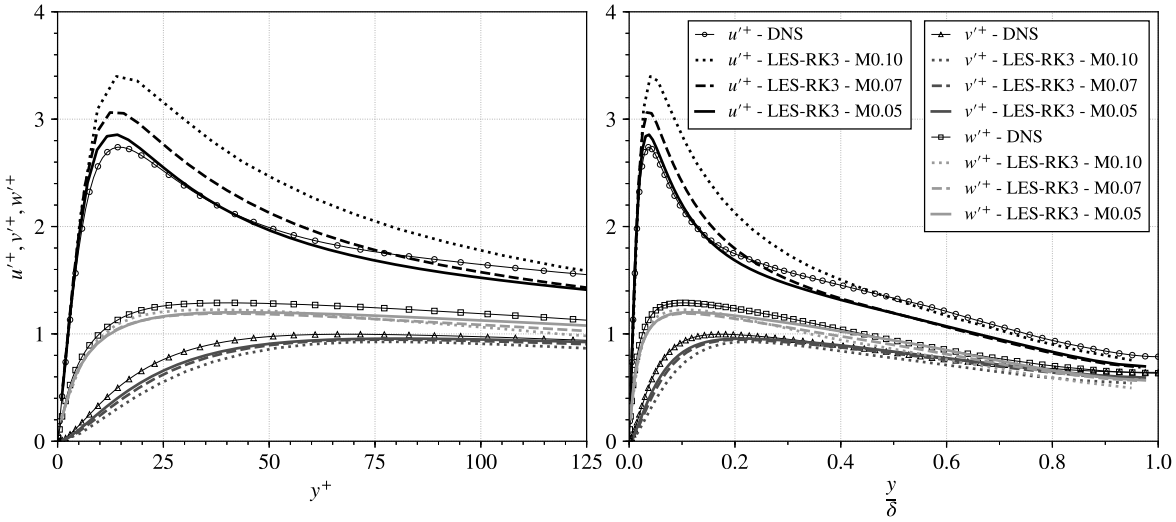

Figure 5.4: Turbulence intensity profiles in three directions u'^+, v'^+, w'^+ of fully developed channel flow at $\mathrm{Re}_\tau = 395$ for different mesh resolutions

Figures 5.4 and 5.5 illustrate the profiles of turbulence intensity in wall-normal coordinates. The graphs show a small overestimation of the intensity u'^+ close to the wall whereas the turbulence intensity in flow direction tend to be underestimated in the region of the main flow. It can be seen that for $\mathrm{Re}_\tau = 395$ a good match is achieved where the model for $\mathrm{Re}_\tau = 590$ shows higher overestimation near the wall. The profiles for the turbulence intensity v'^+ and w'^+ in flow-normal direction show a good overall match to DNS data.

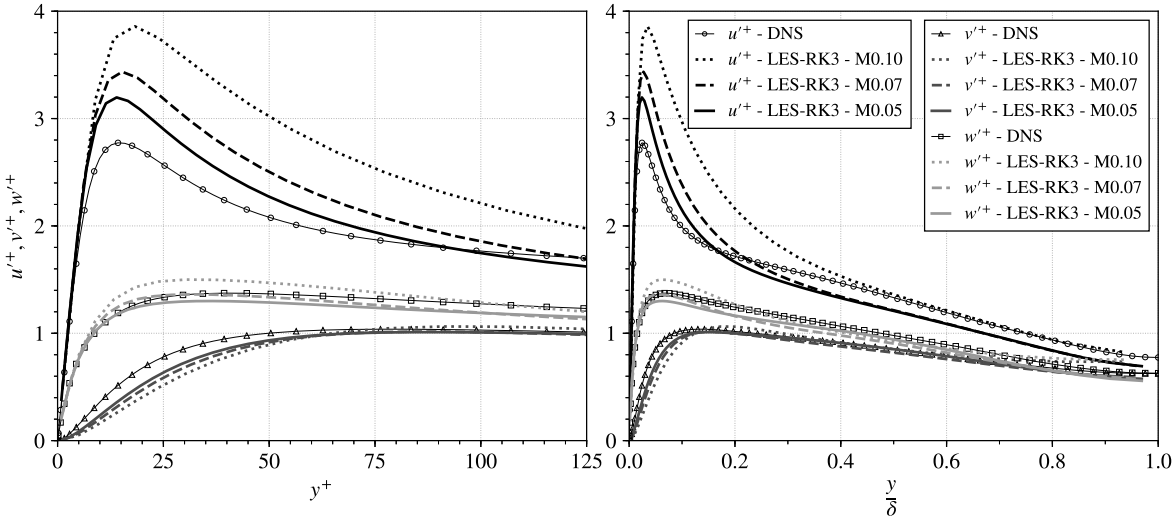

Figure 5.5: Turbulence intensity profiles in three directions u'^+, v'^+, w'^+ of fully developed channel flow at $\mathrm{Re}_\tau = 590$ for different mesh resolutions

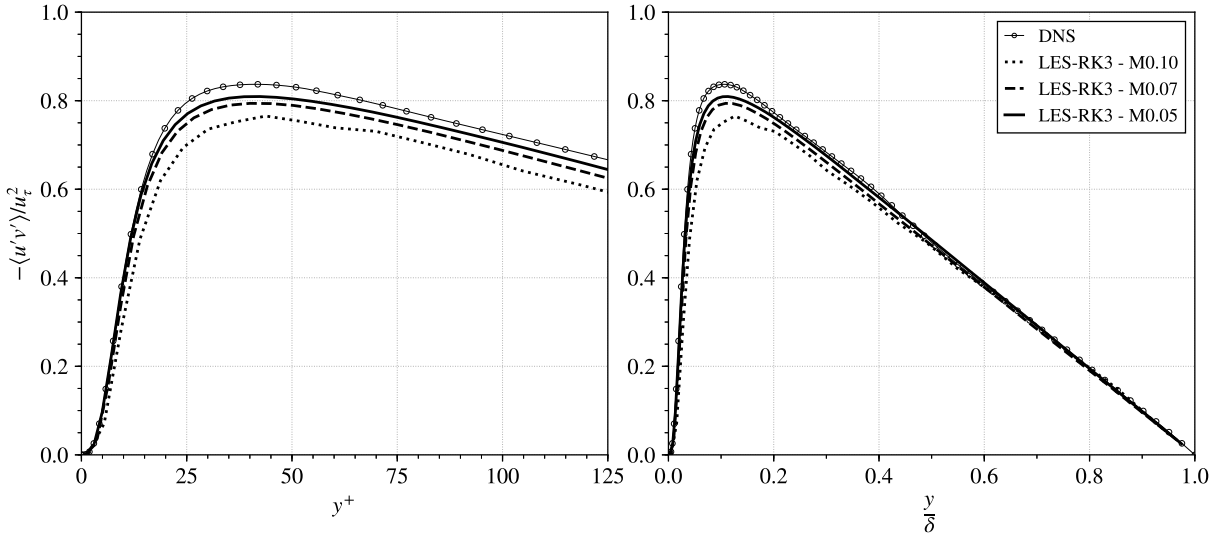

Figure 5.6: Normalised turbulent shear stress profile $\langle u'v'\rangle/\mathrm{u}_\tau^2$ of fully developed channel flow at $\mathrm{Re}_\tau = 395$ for different mesh resolutions

The turbulent shear stress profile $\langle u'v'\rangle/\mathrm{u}_\tau^2$ are depicted in Figure 5.6 and 5.7. For both cases the results show a good match with the DNS data although some little underestimation is present near the wall. In this case the $\mathrm{Re}_\tau = 590$ situation shows less difference than the flow at $\mathrm{Re}_\tau = 395$. In comparison to the turbulent intensity profiles above the Reynolds shear stress profile show less deviation from the DNS data and the difference between the grid resolutions is significant smaller.

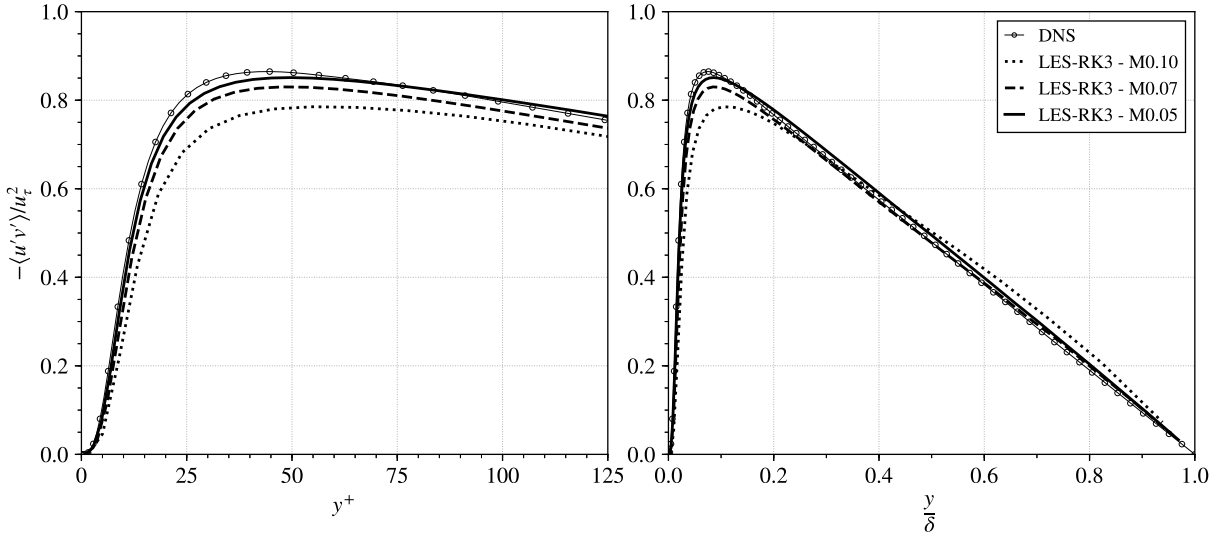

Figure 5.7: Normalised turbulent shear stress profile $\langle u'v'\rangle/\mathrm{u}_\tau^2$ of fully developed channel flow at $\mathrm{Re}_\tau = 590$ for different mesh resolutions

In conclusion the mesh refinement study shows good model convergence of both investigated cases. However, further refinement is not studied in the context of this study as computational resources are limited and the required time for solving increases even more with further refinement.

The volume of simulations and their computational effort results in the selection of model M0.07 to run all different simulations on. The introduced model error is found to be within acceptable limits. Furthermore, model M0.07 fulfils the commonly agreed minimum requirement on the near wall resolution of $y^+ \approx 1$ for LES. The simulation parameters listed in Table 5.2 show that y^+ results close to one for all three cases. Additionally Table 5.2 list more simulation parameters of all committed simulations which are of interest.
Model M0.07 consists of 256 500 cells and six boundary faces. The directional number of elements is show in Table 5.1. To resolve the larger gradients near the wall boundary the cell sizes increase from the wall to the channel mid. The expansion ratio, defined as the ratio between the size of largest cell in the middle of the channel and the smallest wall-bounding cell, is set to 10 for the cases with Reynolds number $\mathrm{Re}_\tau = 180$ and 395. Accounting for the higher turbulent flow at $\mathrm{Re}_\tau = 590$ this expansion ratio is increased to 20, thus reducing the distance between the wall and the first cell center.

Table 5.2: Simulation parameters for all Large Eddy Simulations of turbulent channel flow

Re_τ nom.	Model	Solver	Δx^+	Δz^+	y^+	Re_τ
180	M0.10	RK3	17.89	11.93	0.90	178.90
		RK4	17.95	11.97	0.90	179.49
	M0.07	RK3	11.99	8.00	0.60	179.90
		RK4	12.10	8.07	0.60	181.52
		PISO	12.00	8.00	0.60	179.94
	M0.05	RK3	8.99	6.00	0.46	179.89
		RK4	9.04	6.03	0.46	180.86
395	M0.10	RK3	34.70	23.13	1.75	347.01
		RK4	35.48	23.65	1.79	354.82
	M0.07	RK3	24.59	16.40	1.23	368.89
		RK4	24.76	16.50	1.24	371.33
		PISO	24.34	16.23	1.22	365.09
	M0.05	RK3	18.74	12.49	0.95	374.79
		RK4	18.76	12.50	0.95	375.11
590	M0.10	RK3	52.27	34.84	1.60	522.66
		RK4	52.34	34.90	1.61	523.43
	M0.07	RK3	37.49	24.99	1.15	562.32
		RK4	37.57	25.04	1.15	563.50
		PISO	37.12	24.75	1.14	556.87
	M0.05	RK3	28.92	19.28	0.90	578.30
		RK4	29.02	19.35	0.90	580.40

5.1.2 Boundary and initial conditions

The model boundaries are divided into top and bottom wall, the sides as periodic boundary, inlet and outlet. The outlet is modelled as a zero-pressure outlet. The walls are represented by a no-slip boundary condition setting the velocity magnitude to zero. The cyclic boundary condition assigned to the sides of the model treats the faces as they are physically connected. It allows for the elimination of the influence of any other boundary condition. The inlet boundary condition maps the velocity field from a specified location inside the domain onto the face. The velocity field is scaled automatically at each time step to match an average velocity. Hence, the pressure gradient at the inlet and walls is fixed to zero.
Each simulation is initialised with the pressure and velocity field from a fully developed turbulent channel flow at the specific Reynolds number which is obtained from a precursor simulation. This ensures identical initial conditions for all LES of the same flow characteristics and that mean flow quantities are computed of fully developed turbulent flow. Furthermore, this reduces the required overall time to commit the simulations and, more important, avoids the case in which no turbulence develops from a uniform initial condition. This precursor LES simulates about 230 to 2 500 flow throughs at constant time steps.
The main LES are conducted at a maximal Courant number of 0.5 to 0.6 with automatic time step adjustment. The simulation covers about 110 to 400 flow throughs and the computation of mean flow quantities starts after about 10 to 40 flow throughs.

5.1.3 Results

This section is dedicated to the presentation and discussion of the results obtained from the LES of turbulent channel flow. The results are presented following from high to low friction Reynolds number Re_τ. The results of the custom-built solver `rk3fracsFoam` are labelled as 'RK3' and similarly results obtained with `rk4fracsFoam` are labelled as 'RK4'. The newly implemented solvers are validated with DNS data published by Moser, Kim and Mansour (1999) which is available online (turbulence.ices.texas.edu/MKM_1999.html). Additionally the data is compared to results of the same model computed with the already implemented PISO algorithm of OpenFOAM.
All values presented in this section are based on the resolved quantities $\bar{f}$. For simplification the overbar notation is dropped in this section, i.e. u^+ representing $\bar{\mathrm{u}}^+$. The results are shown either as function of wall coordinates y^+ or global coordinates y/δ.
The normalised mean streamwise velocity profile u^+ is plotted in Figure 5.8 against logarithmic wall coordinates. The few computational points close to the wall show good agreement with DNS results up to $y^+ \approx 10$. With increasing wall distance the overestimation increases and stays approximately constant from $y^+ \approx 50$ on.
The overestimation may have a number of different causes. At first, as Table 5.2 shows, the friction Reynolds number is under-predicted by the simulations. Additionally Re_τ is calculated linearly between the wall and the first cell center, thus creating a significant source of error because y^+ and u^+ are strongly correlated to Re_τ. Second, as mentioned in Section 5.1.1 the bulk velocity has to be approximated because the value applied to the DNS is not available. Numerical schemes used for the spatial approximation and interpolation are another important

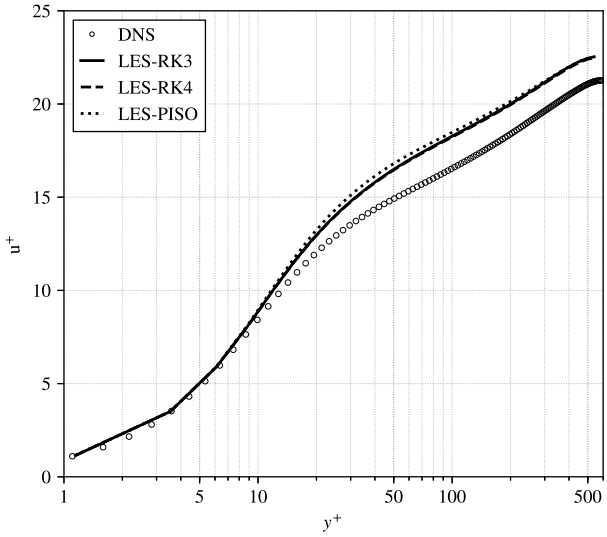

Figure 5.8: Mean streamwise velocity profile for turbulent channel flow at $\mathrm{Re}_\tau = 590$

source of error. The Figures in the Section about the grid refinement study show that more refinement is needed to obtain a better estimation of the values. In context of solver validation the grid resolution is found sufficient enough. The comparison with the results obtained with the well established PISO solver shows only little difference. It is discovered that the results obtained by RK3 and RK4 can hardly be distinguished because they almost perfectly match each other.

The normalised turbulence intensity profiles in Figure 5.9 show a good match in wall-normal and flow-normal direction with regions of small over- and underestimation. u'^+ is overestimated in vicinity of the wall by about 20 % and underestimated in the other region of the flow. The graphs reveal that the new solvers's underestimation of turbulence intensity is less than the existing PISO algorithm shows. However, all solvers show very good agreement up to $y^+ = 10$.

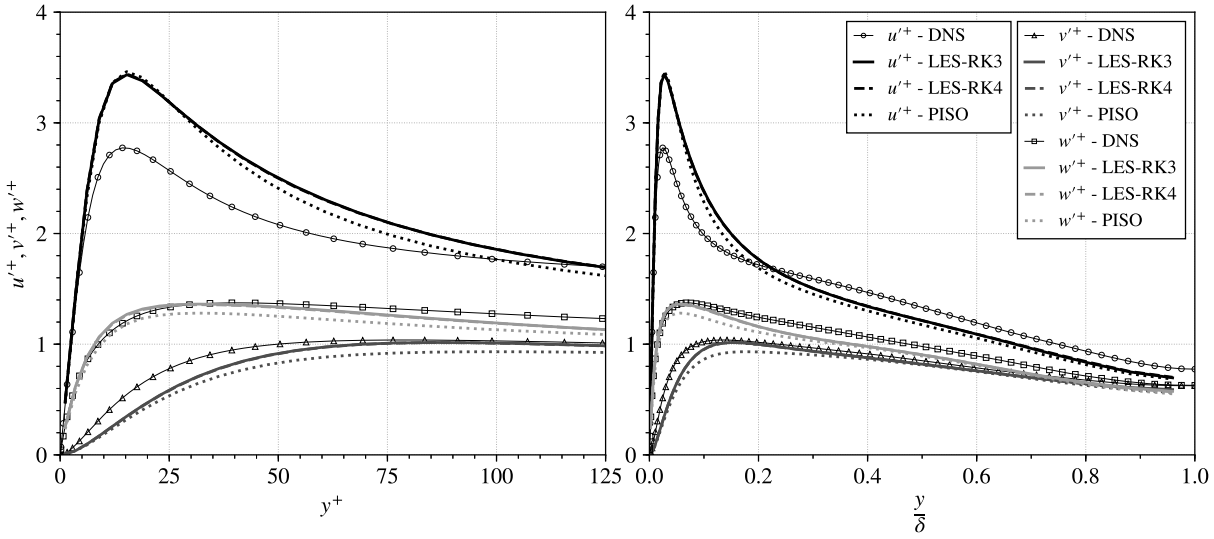

Figure 5.9: Turbulence intensity profiles for turbulent channel flow at $\mathrm{Re}_\tau = 590$

The graph of normalised turbulent shear stress depicted in Figure 5.10 shows little difference between the simulations. The results of LES obtained by PISO and the RK-solvers are in good

agreement with the DNS data across the complete range of y^+.

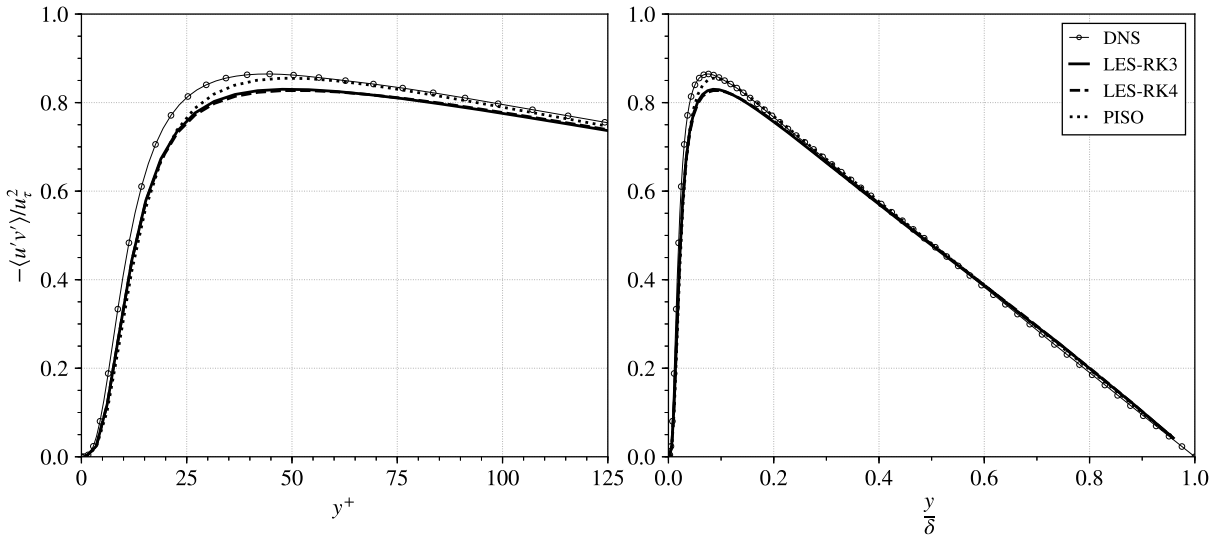

Figure 5.10: Turbulent shear stress profile for turbulent channel flow at $\mathrm{Re}_\tau = 590$

As illustrated in Figures 5.8, 5.9 and 5.10 the results obtained by the `rk3fracsFoam` solver and the `rk4fracsFoam` solver hardly differ. In consequence the results are stated to be equal and the RK4 result is omitted in the following graphs as they provide no additional value.

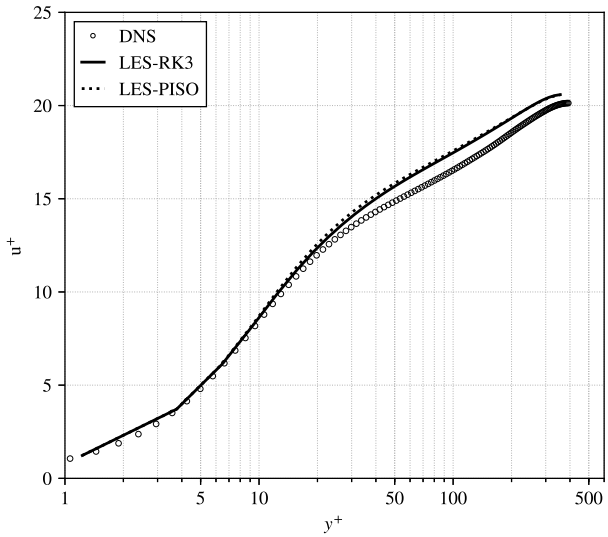

Figure 5.11: Mean streamwise velocity profile for turbulent channel flow at $\mathrm{Re}_\tau = 395$

Figure 5.11 illustrates the normalised streamwise velocity profile u^+ for the case $\mathrm{Re}_\tau = 395$. In this case the velocity profile matches better the data from Moser, Kim and Mansour (1999). As the bulk velocity is retrieved via the `channel395` tutorial packaged within OpenFOAM the overestimation of the streamwise velocity is reduced und a good match of the DNS data is achieved. A mesh refinement towards DNS resolution will finally match the data. As in the previous case the similarity of the results of PISO and RK-solvers is present also in this case.
The corresponding profiles of turbulence intensity are drawn in Figure 5.12. The over-prediction of the streamwise turbulence intensity u'^+ is reduced compared to the previous case. As it illustrated in the right graph of Figure 5.12 the region of overestimation, respective underestimation

of u'^{+} remains the same as at $\mathrm{Re}_\tau = 590$. However, the intensity in wall-normal and flow-normal direction (v'^{+}, w'^{+}) tend to be under-predicted in LES. The simulations parameters in Table 5.2 reveal that the friction Reynolds number is underestimated by a remarkable amount.

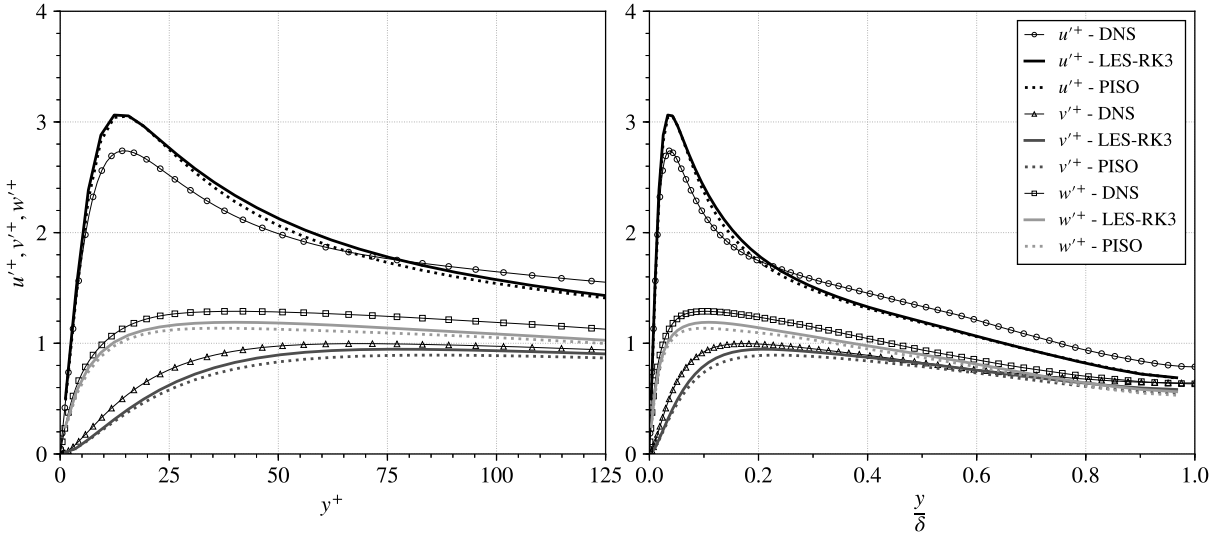

Figure 5.12: Turbulence intensity profiles for turbulent channel flow at $\mathrm{Re}_\tau = 395$

The profile of turbulent shear stresses is slightly underestimated near the wall by the Large Eddy Simulations. It can be discovered in Figure 5.13 that the under-prediction of PISO is reduced compared to RK-solvers. Furthermore LES is capable of producing a good qualitative and also a fair quantitative approximation of the turbulent shear stresses.

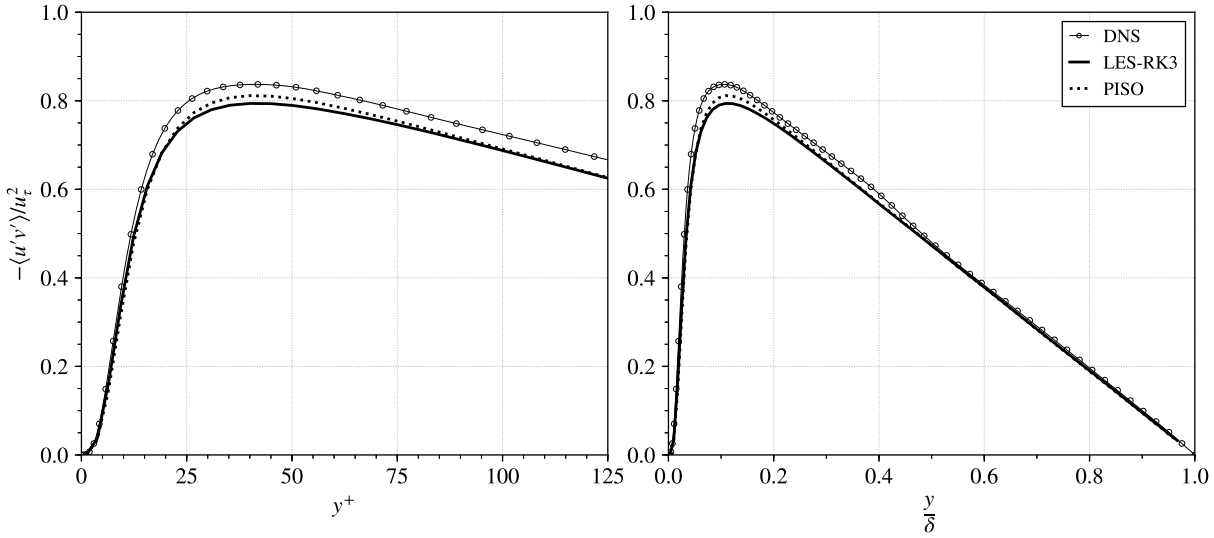

Figure 5.13: Turbulent shear stress profile for turbulent channel flow at $\mathrm{Re}_\tau = 395$

The case of turbulent channel flow at $\mathrm{Re}_\tau = 180$ shows a much better prediction of the Reynolds number. For all models and the three used solving algorithms the calculated value for Re_τ results very close to the nominal value. However, this case can not be fully taken into account because this flow problem is situated close to the laminar-turbulent limit and turbulence has less effect on the flow. The mean velocity profile illustrated in Figure 5.14 reveals a very good

agreement of the LES with DNS and only minor differences between the solvers and DNS are seen.

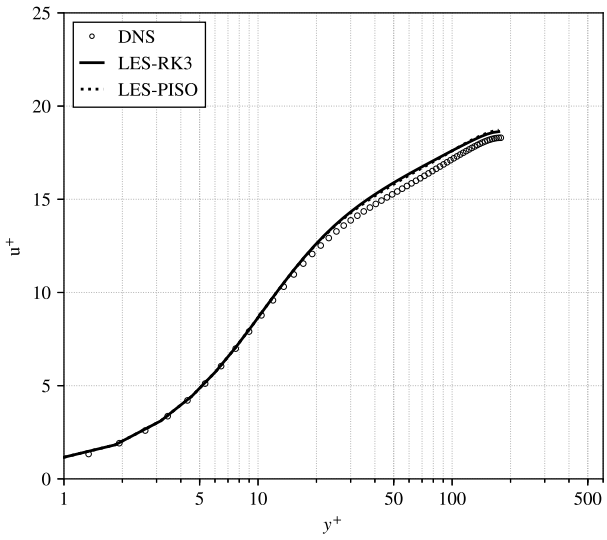

Figure 5.14: Mean streamwise velocity profile for turbulent channel flow at $\mathrm{Re}_\tau = 180$

The resulting turbulence intensity profiles for $\mathrm{Re}_\tau = 180$ depicted in Figure 5.15 show a very good match of the LES to DNS. In this case u'^+, v'^+ and w'^+ are qualitative and quantitative quite good approximations. The results of PISO and the RK-solvers can hardly be visually distinguished proving the results to be identical, although some minor differences can be seen in the right image of Figure 5.15.

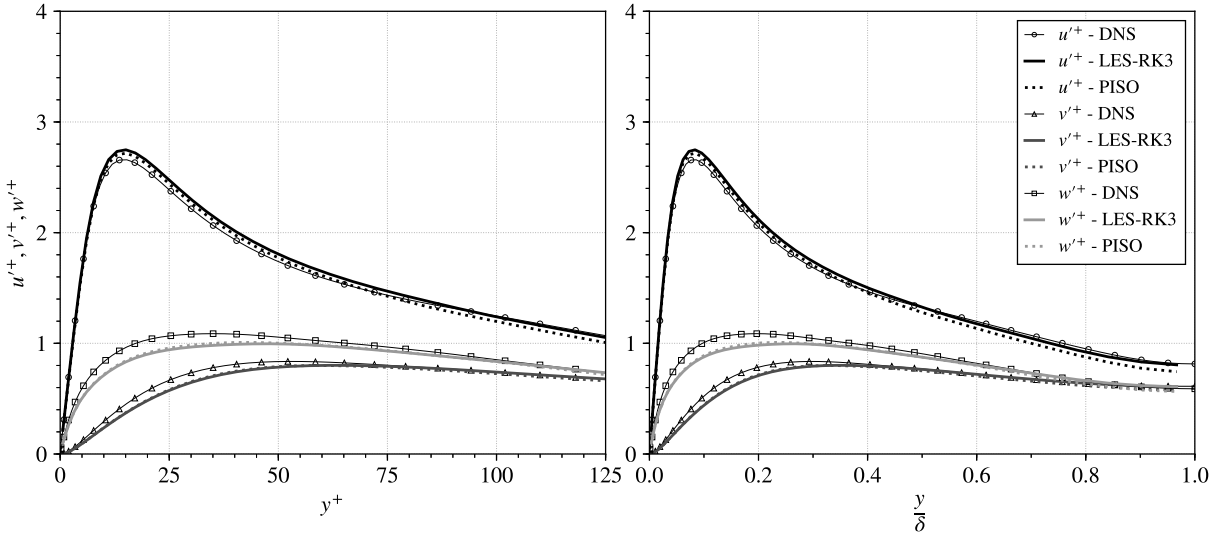

Figure 5.15: Turbulence intensity profile for turbulent channel flow at $\mathrm{Re}_\tau = 180$

The turbulent shear stress profile in Figure 5.16 shows that the difference between the Large Eddy Simulations and the Direct Numerical Simulation is decreased further compared to higher Reynolds numbers. Above $y^+ \approx 75$ the profiles perfectly match each other and closer to the wall only minor differences are present.

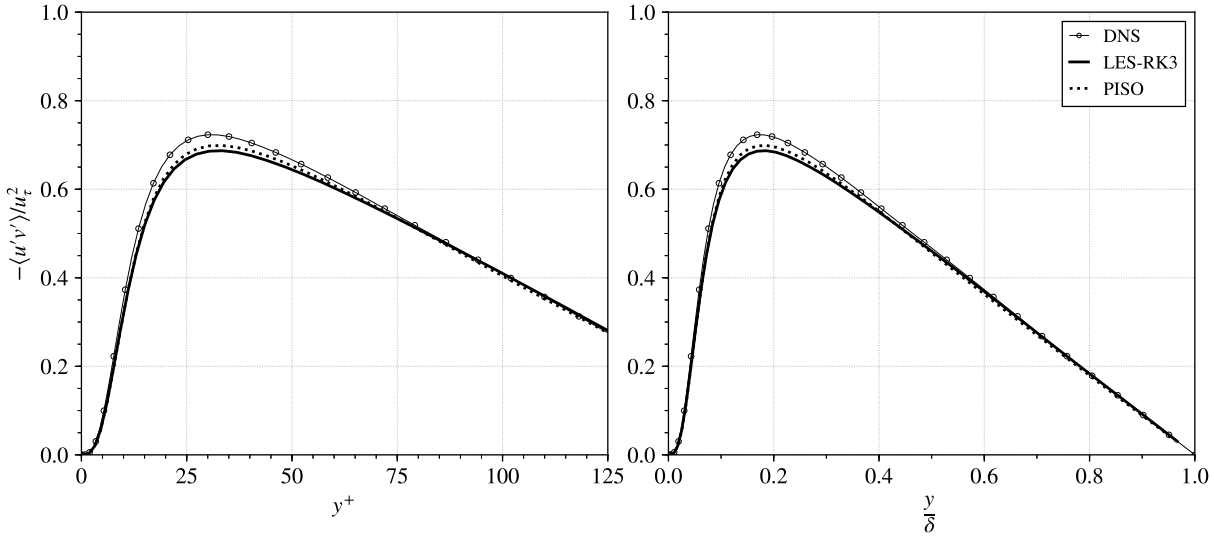

Figure 5.16: Turbulent shear stress profile for turbulent channel flow at $\mathrm{Re}_\tau = 180$

Figures 5.17 to 5.19 show contour plot of the instantaneous velocity magnitude $|\mathbf{u}^+|$ for different friction Reynolds number Re_τ. The left image shows the streamwise distribution of the velocity field in the XY-plane in the centre of the channel. The flow direction is from left to right. The right part of the figures illustrates the spanwise distribution of the velocities located as ZY-plane in the middle of the channel. The view is therefore in flow direction. For better visual comparison the colour scale and contours are kept the same for all three figures.

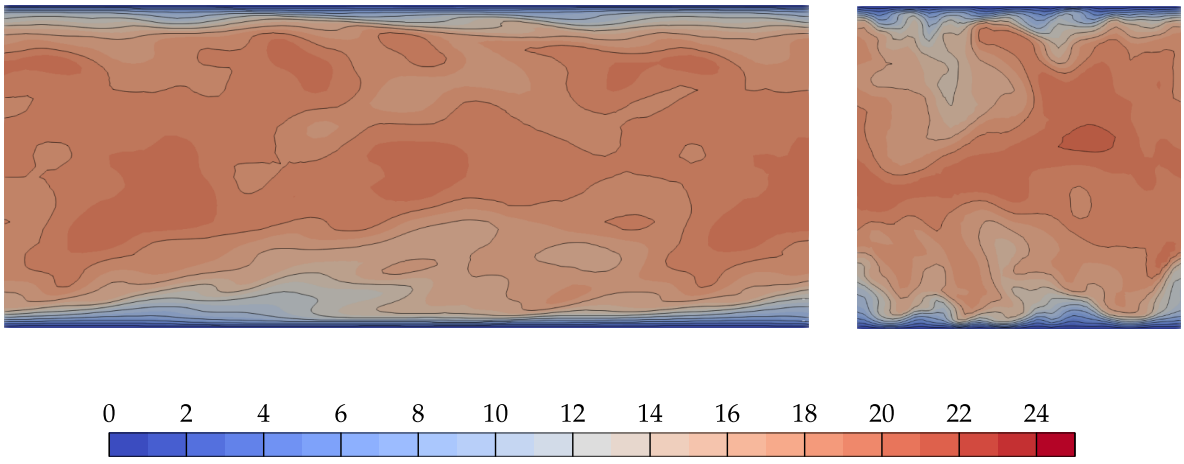

Figure 5.17: Contours of instantaneous velocity magnitude $|\mathbf{u}^+|$ at $\mathrm{Re}_\tau = 180$

The flow at $\mathrm{Re}_\tau = 180$ depicted in Figure 5.17 shows little turbulent structures and a relatively small velocity gradient at the wall. In Figure 5.18 which depicts the velocity field at $\mathrm{Re}_\tau = 395$ more turbulent structures are visible and the gradient is significantly increased. The velocity field in Figure 5.19 shows numerous eddies and a high velocity gradient at the wall.

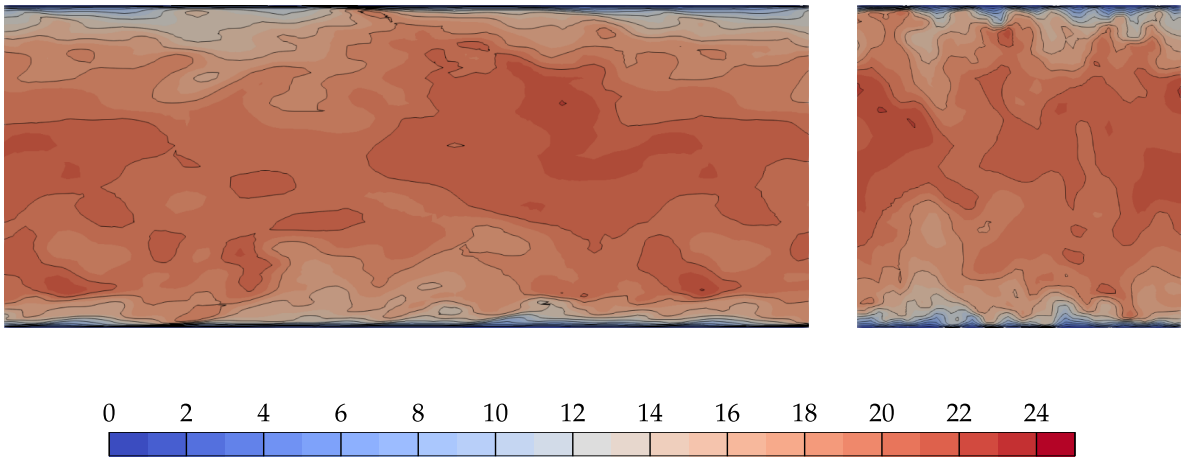

Figure 5.18: Contours of instantaneous velocity magnitude $|\mathbf{u}^+|$ at $\mathrm{Re}_\tau = 395$

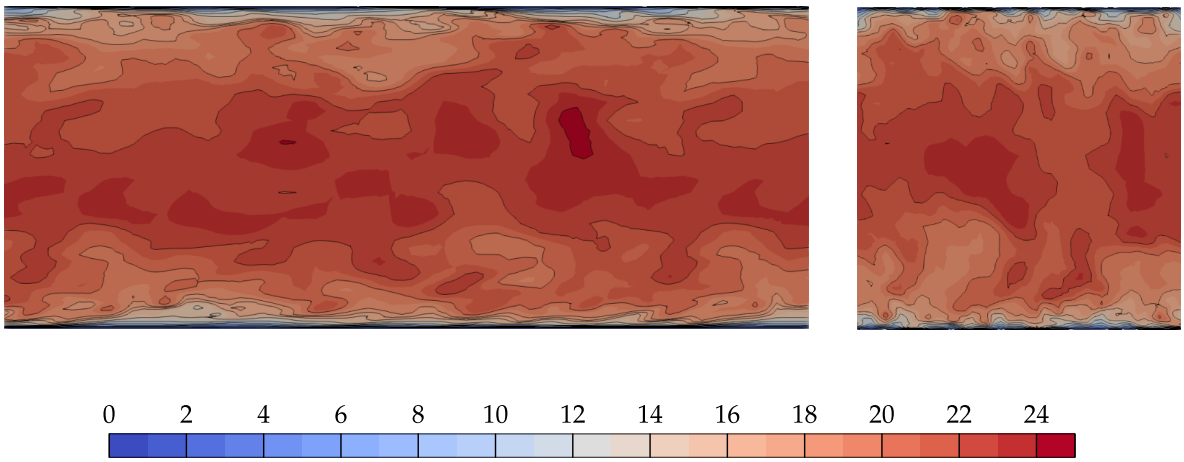

Figure 5.19: Contours of instantaneous velocity magnitude $|\mathbf{u}^+|$ at $\mathrm{Re}_\tau = 590$

5.2 Case two – backward-facing step

This second case investigates turbulent flow over a backward-facing step which is also seen as sudden expansion of the channel. The problem definition is taken from the direct numerical simulations by Le, Moin and Kim (1997). This flow situation has been simulated by several others in different derivations and for different purposes. The published data from the simulations carried out by Le, Moin and Kim (1997) and experimental data created by Jovic and Driver (1994) are used for the validation. The flow simulations are carried out at Reynolds number $\mathrm{Re_h} = \mathrm{u_0 h}/\nu = 5100$ with step height h and $\mathrm{u_0}$ as the mean inlet free stream velocity. The study is based on an expansion ratio of 1.20.

5.2.1 Computational domain

The domain is describes as rectangular channel 4h wide and in total 30h in length. The schematic view of the computational domain is illustrated in Figure 5.20 showing the dimensions, the reference coordinate system and the boundary faces. The inlet section is 10h long and measures 5h in height. The expanded section measures 20h in length and the height is 6h. The geometric discretisation of the model was developed within the process of case set-up. The model consists of 854 810 cells and seven faces forming the boundary of the domain.

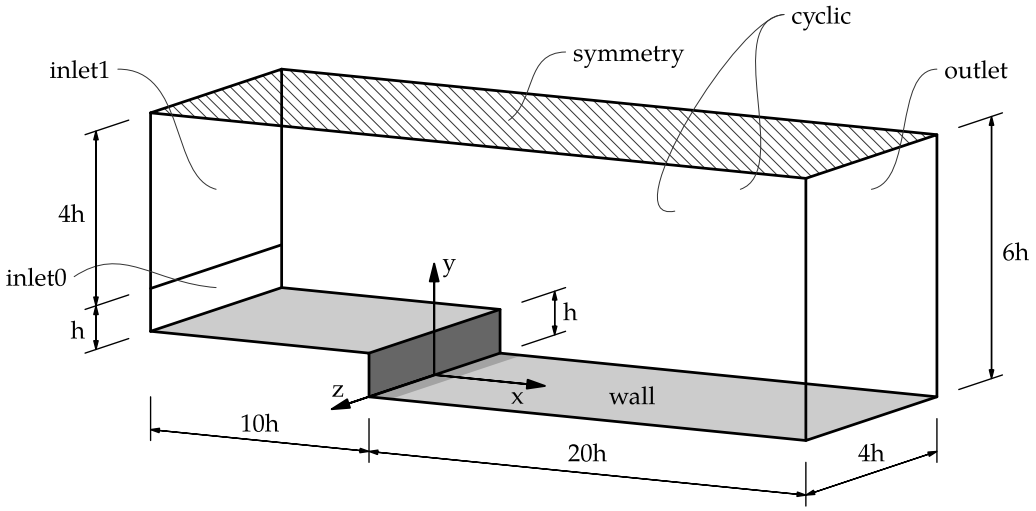

Figure 5.20: Geometry and dimension of the backward-facing step flow domain

At the inlet section the mesh size is uniform in x and z-direction, in wall-normal direction the cell size reduces closer to the wall. In the expanded section the streamwise cell size is lowered at the step and increases over a part of the section and is uniform afterwards. The spanwise uniform cell size in z-direction is continued from the inlet section. The wall-normal cell size distribution is taken from the inlet section and projected to the end of the channel and another block of vertical refinement is added in the expansion region. The dimensionless grid spacing is shown in Table 5.3. The values are calculated based on the inlet friction velocity $\mathrm{u_{\tau 0}}$ in the inlet section. Δy^+ is denoting the size of the first cell at the wall and the first value being calculated in the inlet section and the second value calculated in the expanded section of the channel.

Table 5.3: Dimensionless grid spacings

$\min \Delta x^+$	$\max \Delta x^+$	Δy_1^+	Δy_2^+	Δz^+
1.7	27.6	2.0	1.4	22.3

Figure 5.21 illustrates the mesh resolution and position of the first computational point off the wall in dimensionless coordinates calculated based on the mean wall shear stress in the expanded section. Δx^+ represents the distance between the centres of neighbouring cells. Values of y^+ result below 1.5, the maximum dimensionless grid spacing in z-direction Δz^+ is about 19 and the maximum in x-direction Δx^+ results about 23.

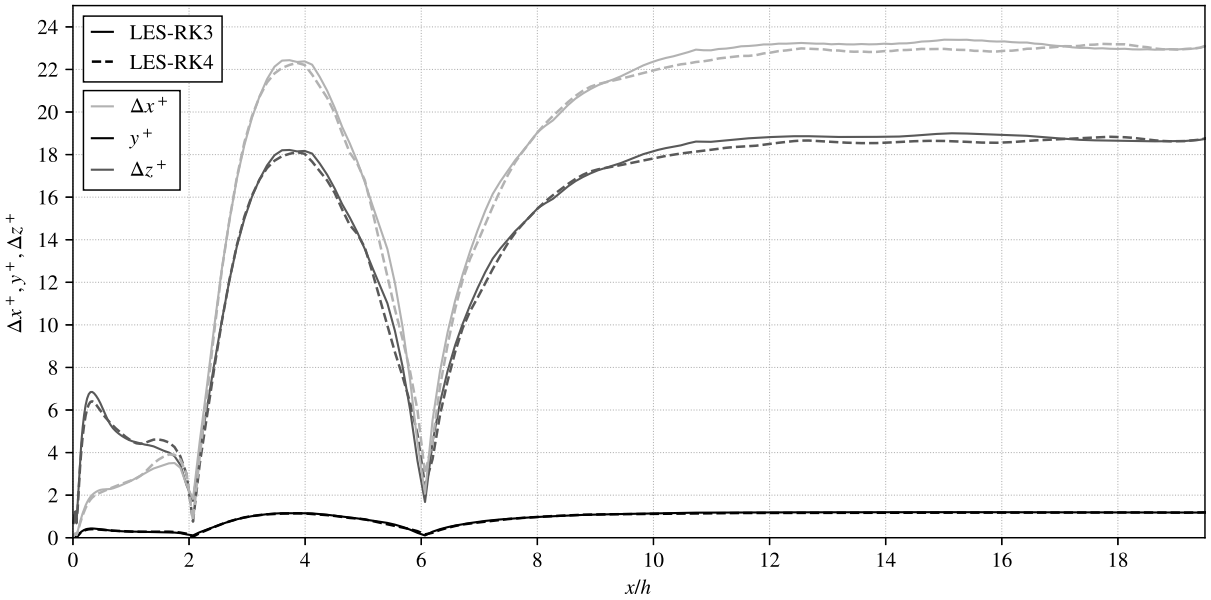

Figure 5.21: Mesh resolution Δx^+, y^+ and Δz^+ in dimensionless wall coordinates in the expanded section of the channel

5.2.2 Boundary and initial conditions

The model boundaries are divided into top and bottom wall, the sides, two inlet boundaries and the outlet. The outlet is modelled as zero-pressure outlet. At the bottom wall the no-slip condition is applied setting all velocity components to zero. The top wall is used as symmetry plane. The cyclic boundary condition is applied to the sides of the domain virtually connecting them. As the original study by Le, Moin and Kim (1997) uses a flat-plate turbulent boundary layer velocity profile at the inlet this boundary is divided into two parts as shown in Figure 5.20. This step allows the reproduction of the inlet velocity profile at $x/h = -3.0$ with already implemented boundary conditions of OpenFOAM. The upper part of the inlet gives a uniform velocity field with $\mathrm{u} = \mathrm{u}_0$. Furthermore the lower part is equipped with a mapped velocity boundary condition. This maps the velocity field from $x/h = -2.0$ on the inlet and scales it according a specified average. Figure 5.22 below shows the inlet velocity profile at $x/h = -3.0$ comparing the profile of DNS, Experiment and LES which is carried out with the new solving algorithms.

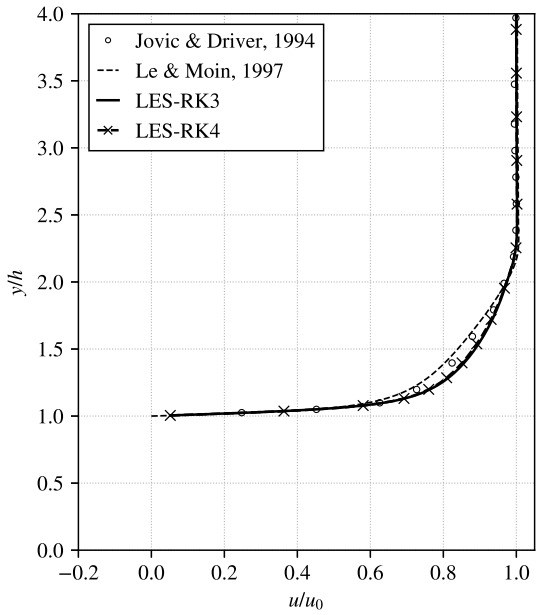

Figure 5.22: Mean streamwise inlet velocity profile at x/h = -3.0

It shows that a close match of the condition is achieved without implementing the specified velocity profile as boundary condition in OpenFOAM. The Large Eddy Simulations with Runge-Kutta fractional step solvers are initialised with the resulting pressure and velocity field of a precursor simulation. Thus guaranteeing a fully developed turbulent flow for the calculation of mean quantities and identical initial conditions for all simulations. Then the flow is simulated for a total time of approximately 4 000 h/u_0 at constant time steps at $\Delta t \approx 0.0039\ \mathrm{h}/\mathrm{u}_0$. The time for simulating the initialisation and allowing the initial transients to pass is not included in this total time. The averaged flow quantities are calculated based on the resulting fields at each time step.

5.2.3 Results

In this section the results of the study of turbulent flow over a backward-facing step are presented and discussed. Similar to the previous case of channel flow the results computed with `rk3fracsFoam` are labelled as 'RK3' and `rk4fracsFoam` as 'RK4' respectively. The results of these two solving algorithms are validated with results of the direct numerical simulation by Le, Moin and Kim (1997) and additionally compared with experimental data published by Jovic and Driver (1994).

All results of Large Eddy Simulations in this section are resolved quantities. Furthermore, as it is dealt with averaged flow quantities the notation is simplified, i.e. $\langle\bar{\mathrm{u}}\rangle$ being represented by u. The results are shown as function of global coordinates. The graphs are plotted in a way that the coordinates follow the instinctive direction, y/h being oriented vertically and x/h oriented horizontally.

As a first measure for comparison the reattachment length X_r is calculated based on the location of zero wall shear stress τ_w. Based on the LES results the reattachment length is calculated with x/h = 6.04 to 6.10. Le, Moin and Kim (1997) calculated a mean reattachment length of 6.28h and the measurements of Jovic and Driver (1994) vary between 6.0h and 6.1h. Further the skin friction coefficient is compared. The normalised friction coefficient C_f is calculated as following:

$$C_f = \frac{2\tau_\mathrm{w}}{\rho\,\mathrm{u}_0^2}$$

As illustrated in Figure 5.23 the wall shear stress, respectively the friction coefficient computed by LES is matching the experimental and DNS data quite well. The comparison with available data and Figure 10 in Le, Moin and Kim (1997) confirms good agreement although there is some quantitative underestimation by the LES.

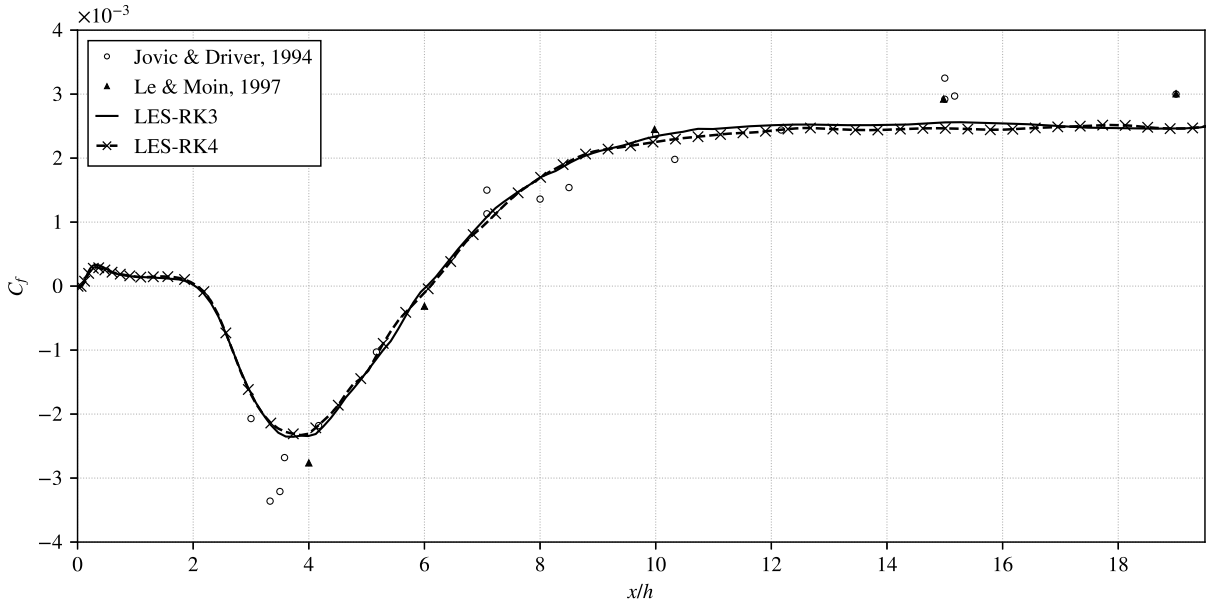

Figure 5.23: Comparison of the skin friction coefficient C_f between LES and literature

Figure 5.24 shows the streamlines of the mean flow. The primary and secondary vortex are clearly identifiable in the figure. Additionally a third small corner recirculation is present in the corner which can hardly be seen in the image. The mean flow field retrieved by DNS shows very similar vortex structures.

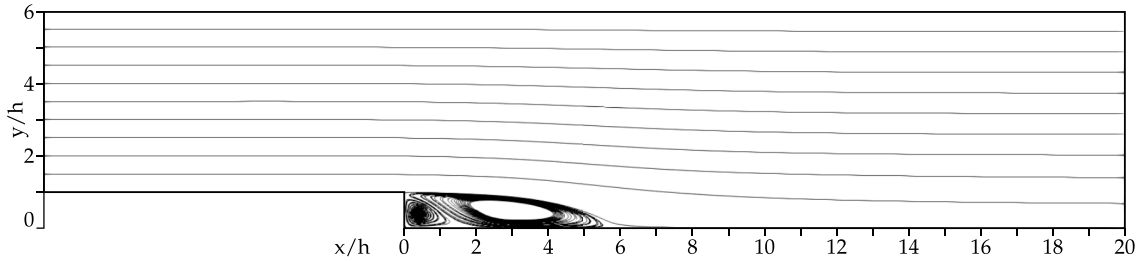

Figure 5.24: Streamlines of the averaged flow, reattachment length $X_\mathrm{r} = 6.04\mathrm{h}$

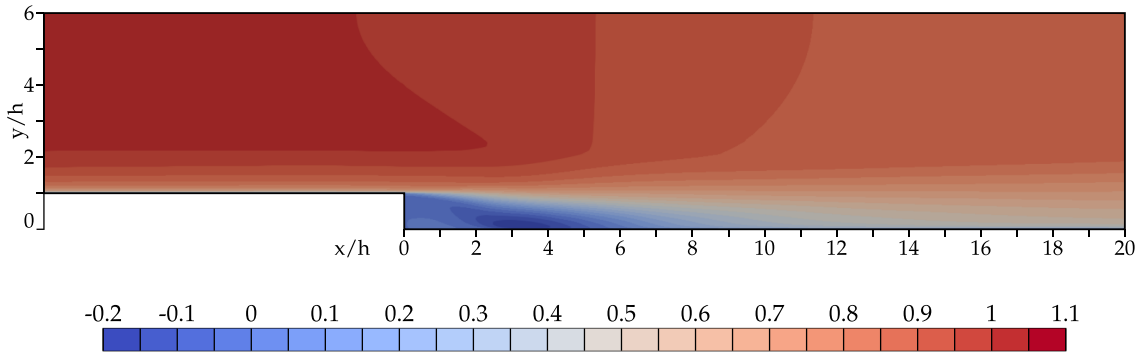

Figure 5.25: Contour plot of the time-averaged streamwise velocity u/u_0

Figure 5.25 shows the contour plot of the mean streamwise velocity u/u_0. The reduction of velocity due to the channel expansion can be seen in the upper part of the domain. Close to

the bottom wall the velocity gradient is increased and illustrating the turbulent boundary layer. The blue coloured area after the step indicates the region of recirculation and low velocities.

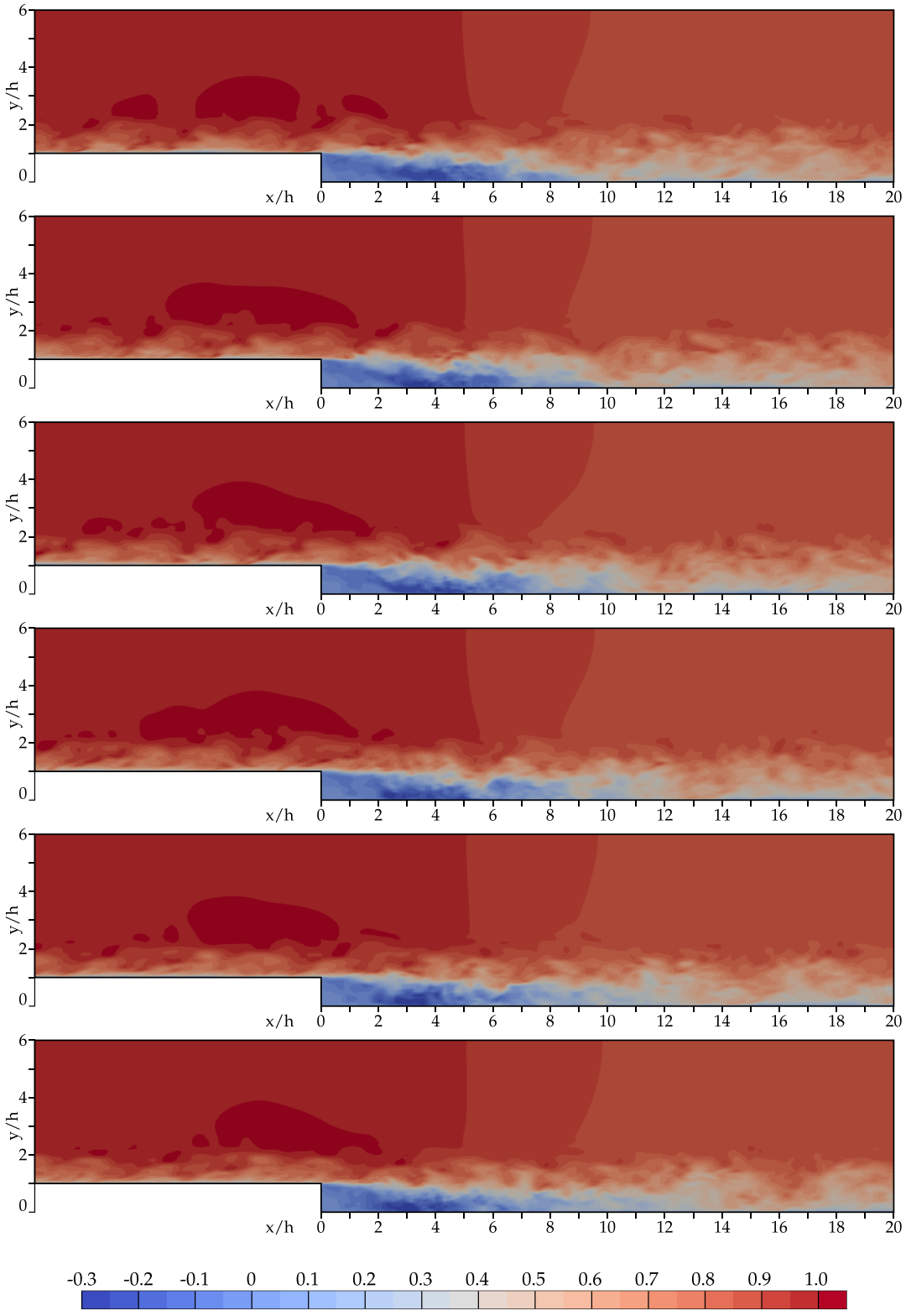

Figure 5.26: Contours of the instantaneous streamwise velocity u/u_0 at equally spaced instants

Figure 5.26 shows the normalised instantaneous velocity field in flow direction at different points in time. The time is increased from top to bottom between the consecutive images by

$\Delta t = 2.363\,\mathrm{h}/\mathrm{u}_0$. The turbulent boundary layer is clearly visible and reaches up to about y/h = 2 into the flow. Still some influence of the turbulent layer can be seen up to y/h = 4. The main flow has only little turbulent motion and is mostly uniform. The consecutive images show the propagation and the development of the different turbulent structures.

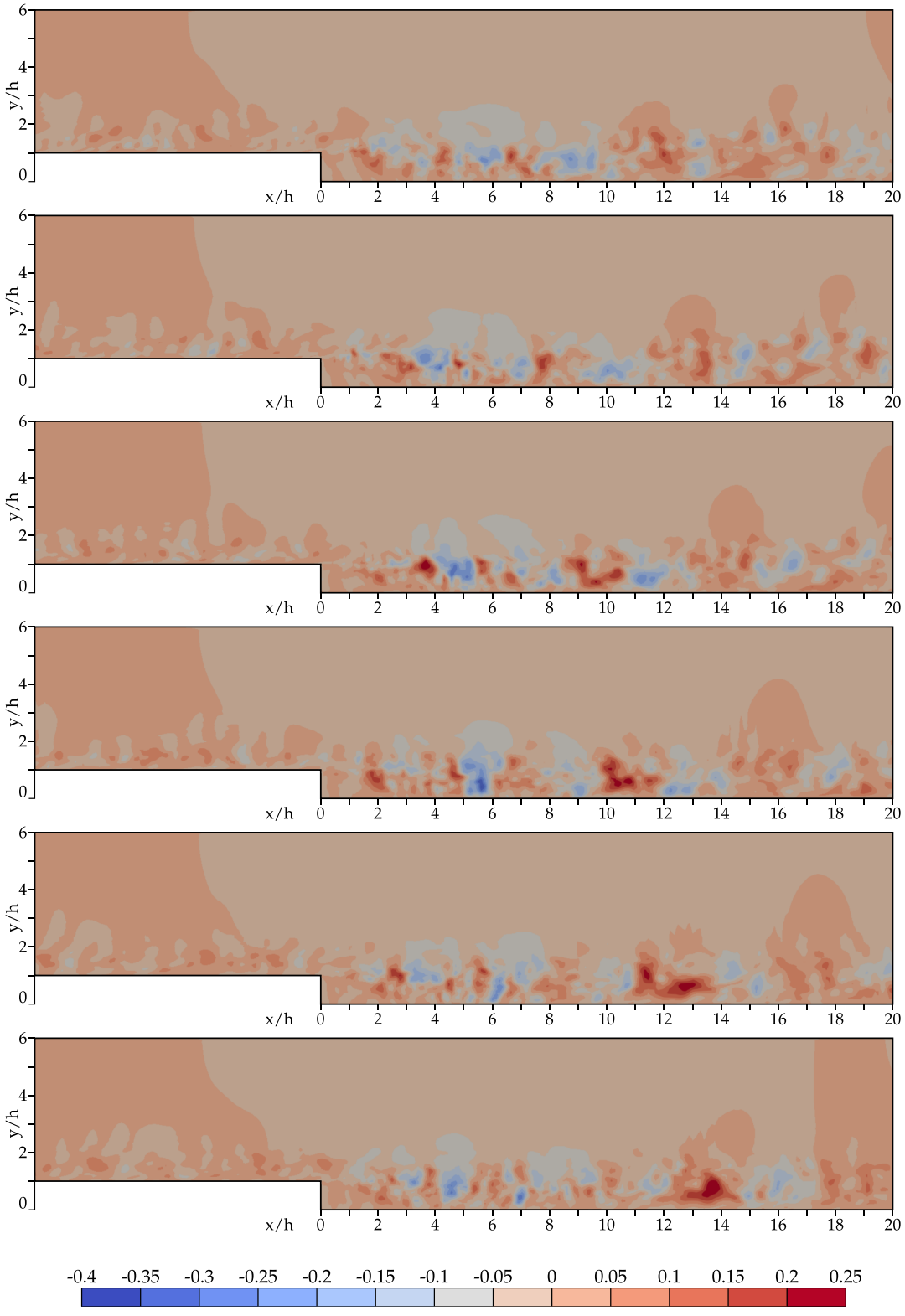

Figure 5.27: Contours of the instantaneous wall-normal velocity v/u_0 at equally spaced instants

In Figure 5.27 the normalised instantaneous wall-normal velocity is illustrated. The time difference between consecutive images is $2.363\,\mathrm{h}/\mathrm{u}_0$. As the magnitude of the velocity in wall-normal direction is much lower than in streamwise direction the turbulent structures can be identified more clearly. As in Figure 5.26 the turbulent boundary layer can be distinguished well from the uniform main flow. The turbulent boundary layer reaches up to about $\mathrm{y}/\mathrm{h} = 2$.
The contours of the mean pressure $(\mathrm{p} - \mathrm{p}_0) / \left(\rho \mathrm{u}_0^2\right)$ are illustrated in Figure 5.28. The lowest pressure in the field is located close to the center of the primary vortex caused by the expansion of the channel. The agreement with Figure 13 in Le, Moin and Kim (1997) is confirmed by visual comparison.

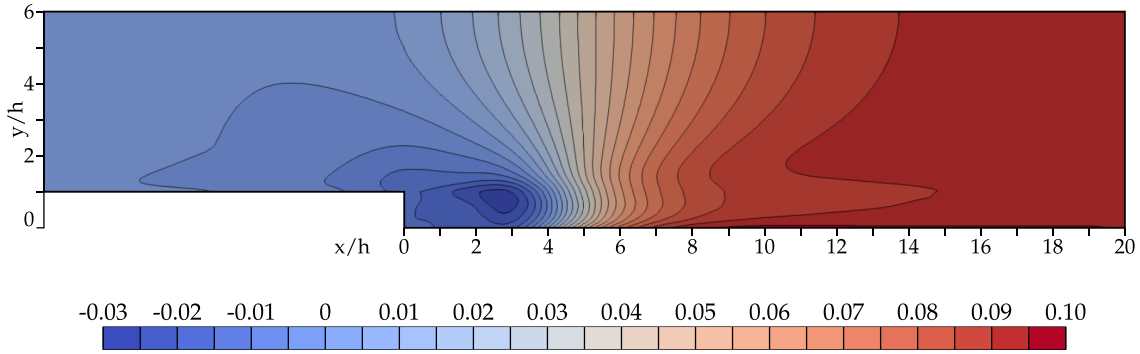

Figure 5.28: Contours of normalised mean pressure

The r.m.s. pressure fluctuations $\sqrt{\mathrm{p}'^2} / \left(\rho \mathrm{u}_0^2\right)$ are shown in Figure 5.29. The peak values are located along the dividing streamline. Influences of the boundary conditions are visible at the inlet and outlet of the domain. Compared to Le, Moin and Kim (1997) the magnitude of pressure fluctuations is increased to DNS, however, the qualitative distribution is matching well.

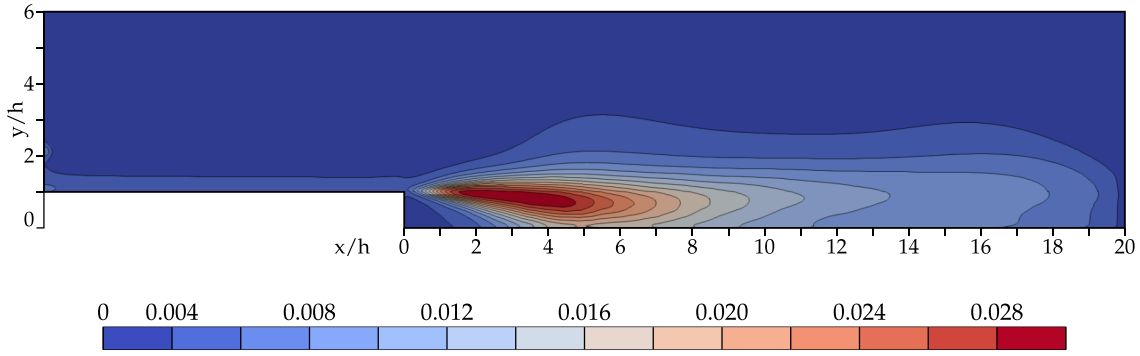

Figure 5.29: Contour plot of mean pressure fluctuations

Figure 5.30 shows contours of the turbulent kinetic energy which is calculated by

$$k = \frac{1}{2}\left(\mathrm{u}'^2 + \mathrm{v}'^2 + \mathrm{w}'^2\right)$$

The quantities are normalised by the square of the free stream inlet velocity u_0. The highest values are found along the diving stream line between the normal flow and the recirculation area. Close to the outlet boundary some influence of the condition applied there can be found in the contour plot.

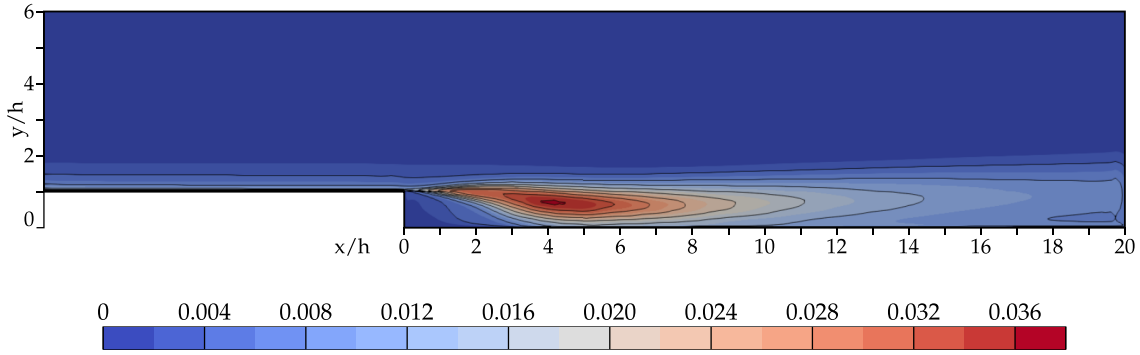

Figure 5.30: Contour plot of mean turbulent kinetic energy k/u_0^2

The following figures illustrate selected flow quantities in specified locations. The locations at which the results are extracted are shown in Figure 5.31. The profile at x/h = -3.0 is used for the calibration and validation of the inlet condition and the velocity profile is shown in Figure 5.22. The result of the other four profiles at x/h = 4.0, 6.0, 10.0 and 19.0 are shown on the next pages.

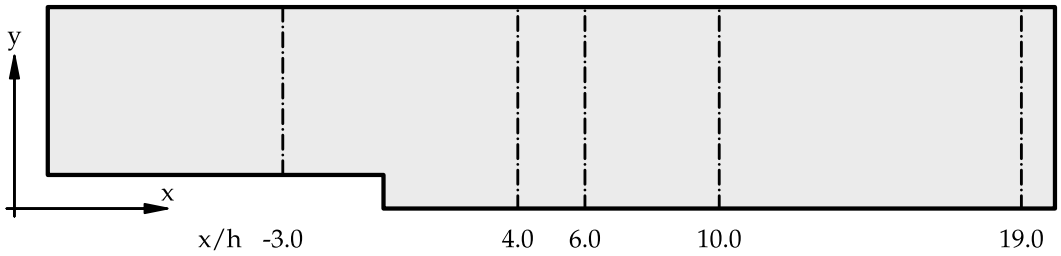

Figure 5.31: Profile locations

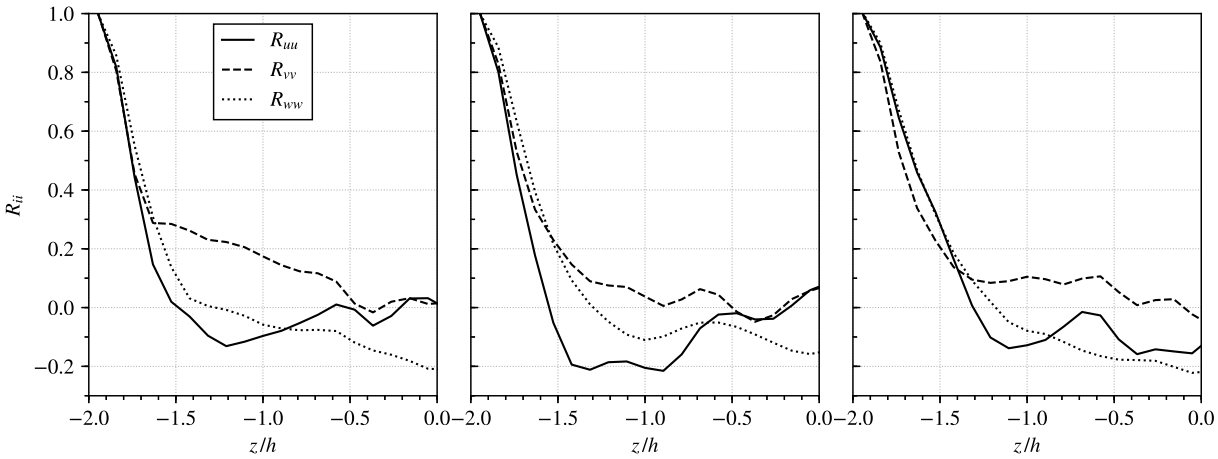

Figure 5.32: Spanwise autocorrelation coefficients at selected locations in the flow sampled over about 300 h/u_0

The graphs in Figure 5.32 show the autocorrelation for the three velocity components at three selected locations in the flow. The data is sampled at the profiles at x/h = 4, 6 and 10 in a distance of y/h = 1 and shown from left to right.

Figure 5.33 illustrates the mean streamwise velocity profiles u/u_0 at four locations. The first profile (a) is located in the area of recirculation. It shows good agreement close to the wall and at y/h = 2.0 and higher. However, the predicted velocity in between is higher as it is visible in data from DNS or experiments. The profile close to the reattachment (b) is a good match of DNS data. Furthermore, the profiles at x/h = 10 and 19 are in good agreement with the validation data although some minor differences are visible. In conclusion Figure 5.33 reveals that LES is capable of reproducing the results of DNS and experiments with a high accuracy. However, no difference between the two newly implemented fractional step solver with Runge-Kutta algorithm (RK3 and RK4) can be observed.

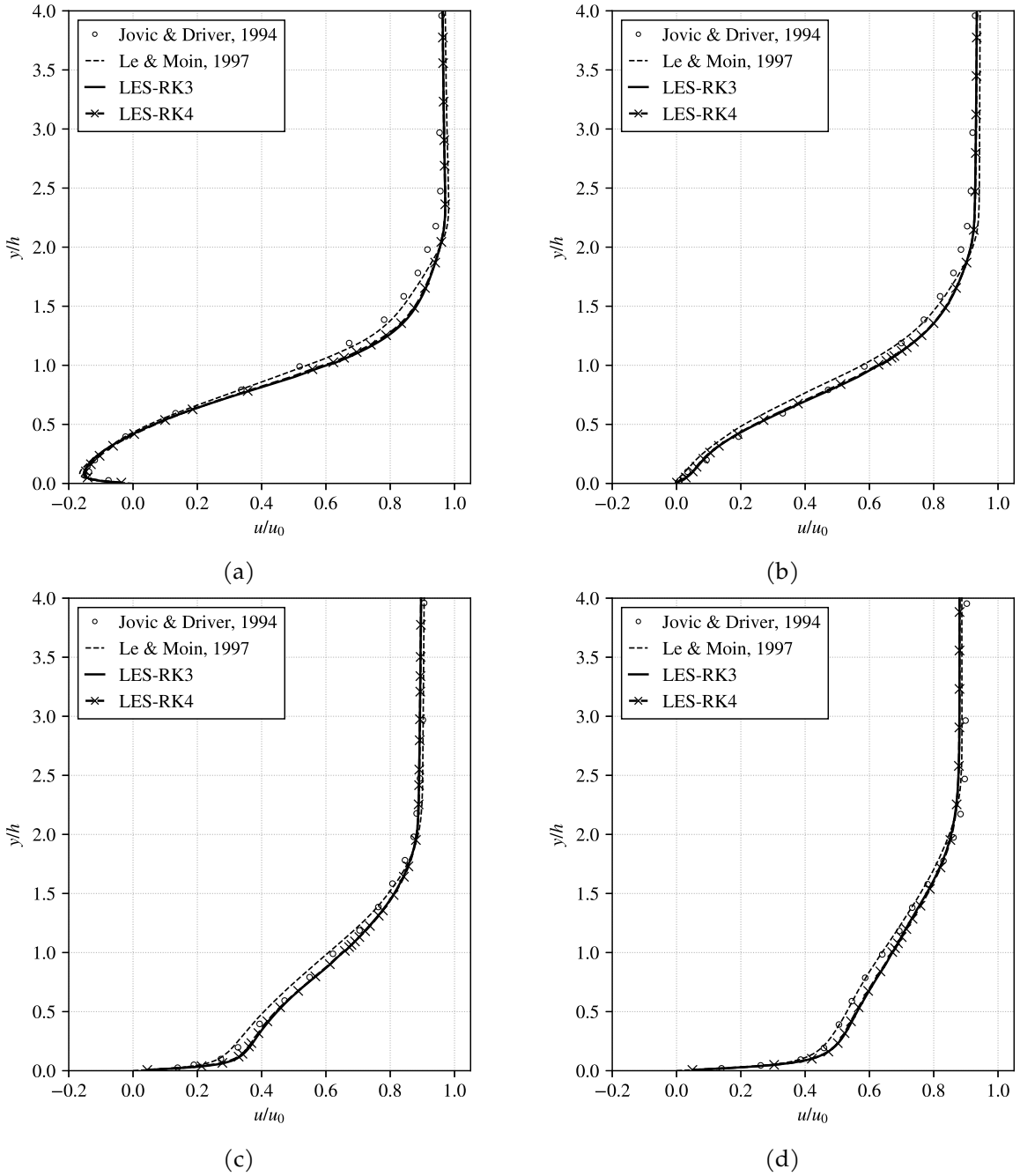

Figure 5.33: Profiles of mean streamwise velocity u/u_0 at x/h = 4.0 (a); 6.0 (b); 10.0 (c); 19.0(d)

The profiles of streamwise turbulence intensity u'/u_0 are illustrated in Figure 5.34. The profiles (a) to (c) are in good agreement of DNS data and in some regions LES reproduces the results of the experiments better than DNS. The profile in (c) is in good agreement up to y/h = 1.5 then under-predicts the intensity in greater distance from the wall but it follows the DNS data. At x/h = 19.0 (d) influence of the boundary condition is present and this leads to an increased deviation in the turbulent boundary layer of the LES results to DNS and the experimental data.

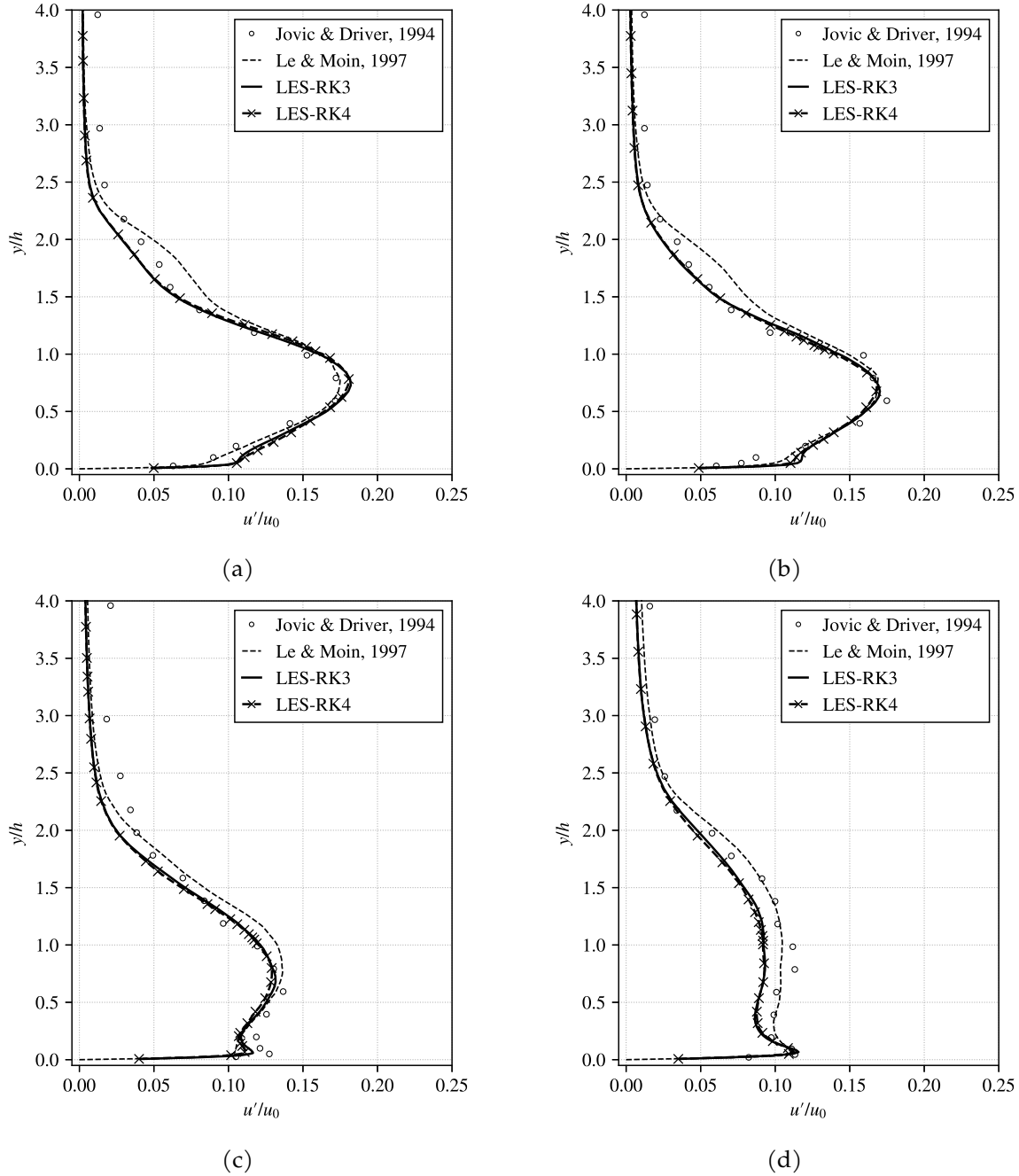

Figure 5.34: Profiles of turbulent intensity u'/u_0 at x/h = 4.0 (a); 6.0 (b); 10.0 (c); 19.0 (d)

The wall-normal turbulence intensity v'/u_0 is illustrated as profiles in Figure 5.35. The sections within the reattachment length give a good approximation of the turbulence intensity. Thus, the peak values are increased and shifted upwards relative to the data. The profiles at x/h = 10 and 19 in Figure 5.35 almost perfectly match the experimental data. It can also be seen that DNS is deviating in this case from experiment and LES.

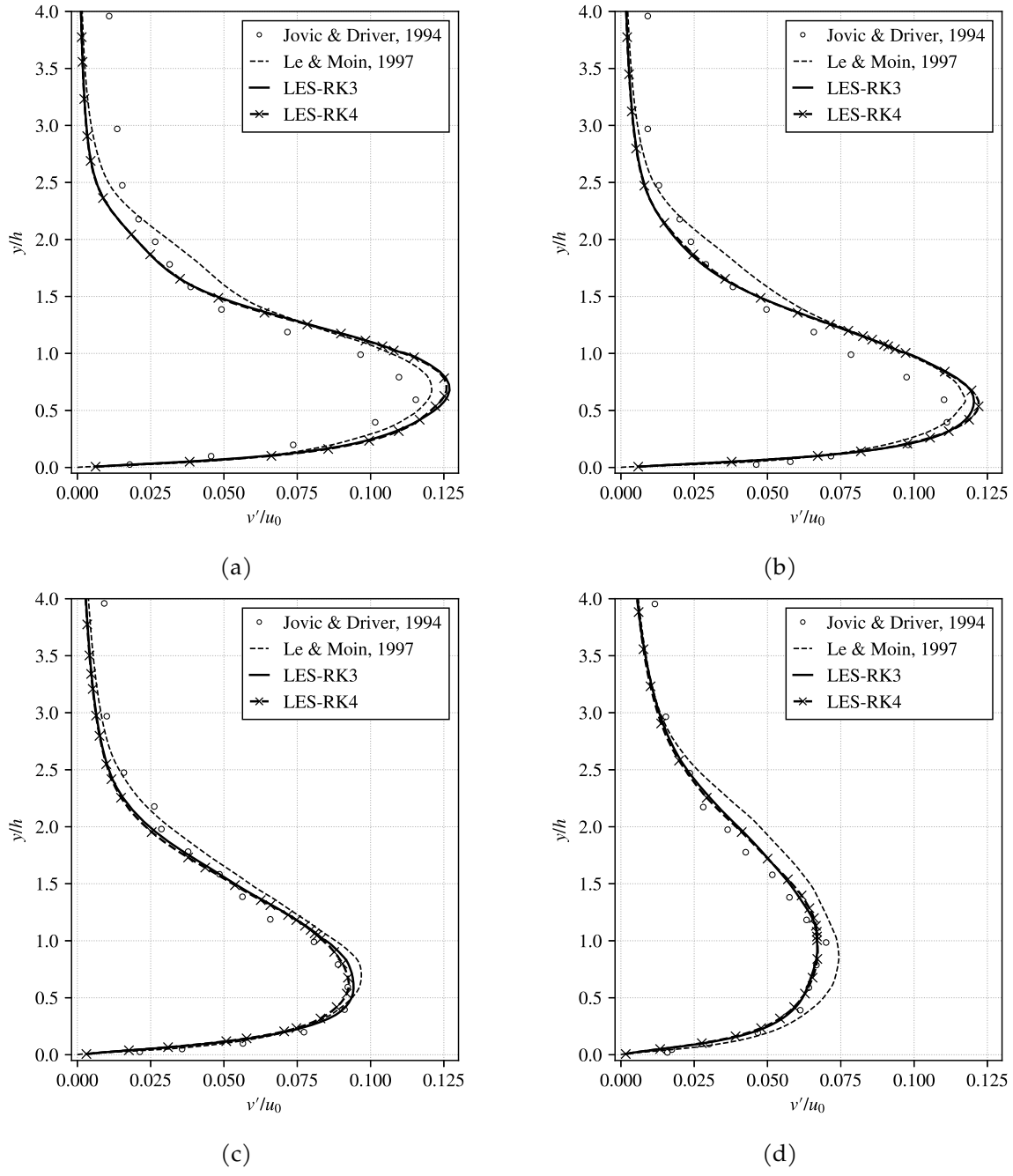

Figure 5.35: Profiles of turbulent intensity v'/u_0 at $x/h = 4.0$ (a); 6.0 (b); 10.0 (c); 19.0 (d)

Illustrating the profiles of turbulent shear stress Figure 5.36 presents the LES well matching DNS and experimental data. In some cases predictions made by the LES are closer to experimental data than DNS does. The peak turbulent shear stress within the recirculation region is higher than in DNS. Also the sharp gradient which can be interpreted from the experimental data can not be predicted. The transition from the turbulent boundary layer to the free stream region is better predicted by the LES.

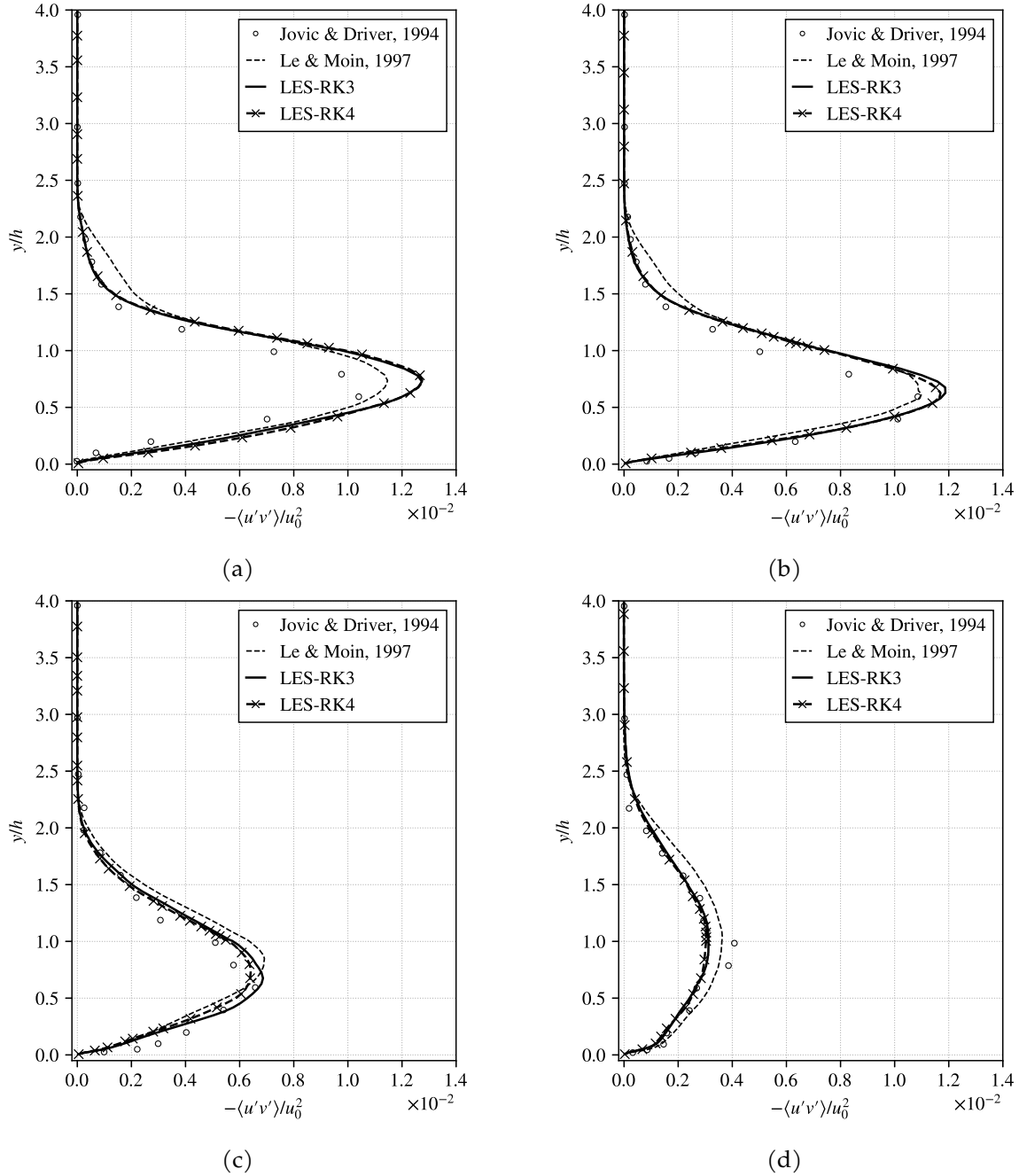

Figure 5.36: Profiles of turbulent shear stress $-\langle u'v'\rangle/\mathrm{u}_0^2$ at $x/h = 4.0$ (a); 6.0 (b); 10.0 (c); 19.0 (d)

6 Conclusion

For the Large Eddy Simulation of problems in hydraulic engineering the solving algorithm is one of the key components. Depending on the problem size, required accuracy and resolution the simulations demand more computational effort. With limited time as main resource nowadays a fast and accurate solver is very valuable. In consequence people steadily come up with new ideas and keep on developing and improving methods for solving problems in fluid flows.
In this thesis the implementation and validation of two new fractional step solving algorithms in OpenFOAM is presented. These algorithms make use of different higher order Runge-Kutta methods for the advancement in time. The solvers are designed for the flow of incompressible Newtonian fluids. The achievements of the present work are summarised in the following points:

- The high level language provided by OpenFOAM and some basic knowledge of programming in C/C++ allow for the implementation of custom solvers with ease. It should be mentioned here that the syntax provided by OpenFOAM also allows for the implementation of other customised code like boundary conditions, turbulence models and much more.

- The solvers `rk3fracsFoam` and `rk4fracsFoam` are first tested on the case of turbulent channel flow at three different friction Reynolds numbers Re_τ. A grid refinement study is carried out for the optimisation of the geometric discretisation. The results of the RK-solvers are then compared to data retrieved from DNS published by Moser, Kim and Mansour (1999). Additionally simulations of the same problem definition are committed with PISO solver and the results are added as second reference to the comparison. The difference between RK and PISO solvers is less than 1 %. The results of LES using the new solving algorithms are in good agreement with data from DNS. Furthermore there is hardly any difference observed between RK3 and RK4 solver. The overall quality of the results is well suitable for most engineering applications in daily business of hydraulic engineers.

- As a second case for the validation of the solving algorithms the turbulent flow over a backward-facing step is selected. In this case DNS data from Le, Moin and Kim (1997) and experimental data from Jovic and Driver (1994) is used as reference. The inlet condition of a flat-plate turbulent boundary layer flow is successfully modelled with basic boundary conditions already implemented in OpenFOAM. The results of `rk3fracsFoam` and `rk4fracsFoam` simulations show very good agreement with the references. Even in some parts the results of LES are in better agreement with the experimental data than DNS. The main flow features, such as reattachment length and profiles of the velocity field and the fluctuations perfectly matched the references.

- To measure the speed of the new solving algorithms the execution time is logged during the simulations of turbulent channel flow at each time step and averaged afterwards. For comparison the same case is solved with the PISO solver of OpenFOAM. The time needed

for PISO algorithm to solve the equations is taken as reference. It takes the fractional step solver with fourth order Runge-Kutta algorithm (`rk4fracsFoam`) only 77 % of the time for solving. The `rk3fracsFoam` is 40 % faster than the PISO algorithm. The computed values for the different turbulent channel cases are listed in Table 6.1. The increase in speed is achieved without lowering the accuracy of the results.

Table 6.1: Comparison of the computational speed of different solvers

Re_τ nom.	PISO	RK4	RK3
180	1.0	0.767	0.590
395	1.0	0.767	0.581
590	1.0	0.773	0.615

In conclusion the new RK-solvers are implemented and validated successfully. The results of the committed Large Eddy Simulations using RK3 and RK4 solvers are in good agreement with Direct Numerical Simulations and experimental results.

Future work

The importance of numerical simulations in hydraulic engineering will further increase in future. Not only the demand on optimising large hydraulic structures will grow but also it will be asked to find the best alternative for small projects in hydraulic engineering. Therefore the following research topics are proposed on the basis of the presented work.

- Direct numerical simulations are generally limited to low Reynolds number flows and simple geometries. However, Large Eddy Simulations are capable of simulating turbulent flows in much more complex geometries. It is therefore proposed to further validate the new solving algorithms for flows in complex geometries. For this cases the validation data can be obtained from experiments.

- For further increase in speed of solution algorithms the implementation of accelerated Runge-Kutta methods is suggested. Such methods store values from the previous time step and therefore require less computation step each time step. Udwadia and Farahani (2008) give some examples of accelerated Runge-Kutta methods and estimate the increase in speed by 20–30 % compared to classical Runge-Kutta methods.

- Problems in hydraulic engineering are basically gravity driven. Problems like the approach flow in a reservoir to an intake structure can be modelled as single-phase. For flow situations which appear in energy dissipation structures or at weirs single-phase models are not longer suitable and must be modelled with multi-phase models. It is therefore suggested to further develop the Runge-Kutta solver algorithms and making them capable of simulating multiphase flow problems. Hence, validation of these multi-phase solvers and a comparison with existing solution algorithms is part of this proposal.

References

Chorin, Alexandre Joel (1967). 'The numerical solution of the Navier-Stokes equations for an incrompressible fluid'. In: *Bulletin of the American Mathematicel Society* 73.6, pp. 928–931.

Issa, Raad I. (1986). 'Solution of the implicitly discretised fluid flow equations by operator-splitting'. In: *Journal of Computational Physics* 62.1, pp. 40–65.

Jasak, Hrvoje (1996). 'Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows'. PhD thesis. Imperial College of Science, Technology and Medicine, London.

Jovic, Srba and David M. Driver (1994). *Backward-facing step measurements at low Reynolds number, Re_h=5000*. NASA, Ames Research Center. URL: ntrs.nasa.gov/search.jsp?R=19940028784.

Le, Hung, Parviz Moin and John Kim (1997). 'Direct numerical simulation of turbulent flow over a backward-facing step'. In: *Journal of Fluid Mechanics* 330, pp. 349–374.

Leonard, A. (1974). 'Energy Cascade in Large-Eddy Simulations of Turbulent Fluid Flows'. In: *Advances in Geophysics* 18, pp. 237–248.

Moser, Robert D., John Kim and Nagi N. Mansour (1999). 'Direct numerical simulation of turbulent channel flow up to $\mathrm{Re}_\tau = 590$'. In: *Physics of fluids* 11.4, pp. 943–945.

Nicoud, Franck and Frédéric Ducros (1999). 'Subgrid-scale stress modelling based on the square of the velocity gradient tensor'. In: *Flow, Turbulence and Combustion* 62.3, pp. 183–200.

Rodi, Wolfgang, George Constantinescu and Thorsten Stoesser (2013). *Large-Eddy Simulation in Hydraulics*. CRC Press/Balkema.

Schumann, U. (1975). 'Subgrid Scale Model for Finite Difference Simulations of Turbulent Flows in Plane Channels and Annuli'. In: *Journal of Computational Physics* 18.4, pp. 376–404.

Shiklomanov, Igor A (1998). *World water resources: a new appraisal and asssesment for the 21st century*. SC.1998/SANS CODE. UNESCO.

Smagorinsky, Joseph (1963). 'General Circulation Experiments with the Primitive Equations'. In: *Monthly Weather Review* 91.3, pp. 99–164.

Sotiropoulos, Fotis (2015). 'Hydraulics in the era of exponentially growing computing power'. In: *Journal of Hydraulic Research* 53.5, pp. 547–560.

Stoesser, Thorsten (2014). 'Large-eddy simulation in hydraulics: Quo Vadis?' In: *Journal of Hydraulic Research* 52.4, pp. 441–452.

Udwadia, Firdaus E. and Artin Farahani (2008). 'Accelerated Runge-Kutta Methods'. In: *Discrete Dynamics in Nature and Society* 2008.

Vuorinen, V. et al. (2014). 'On the implementation of low-dissipative Runge–Kutta projection methods for time dependent flows using OpenFOAM'. In: *Computers & Fluids* 93, pp. 153–163.

A Appendix A – Case definitions

A.1 Turbulent channel flow

Geometry

File A.1: /system/blockMeshDict

```
convertToMeters 1;
vertices
(
   (0 0 0) (5 0 0) (5 1 0) (0 1 0) (0 0 2) (5 0 2) (5 1 2)
   (0 1 2) (5 2 0) (0 2 0) (5 2 2) (0 2 2)
);
blocks
(
   hex (0 1 2 3 4 5 6 7) (75 38 45) simpleGrading (1 10 1)
   hex (3 2 8 9 7 6 10 11) (75 38 45) simpleGrading (1 0.1 1)
);
boundary
(
   frontWall
   {
      type wall;
      faces ((0 1 5 4));
   }
   backWall
   {
      type wall;
      faces ((8 9 11 10));
   }
   sides_half0
   {
      type cyclic;
      neighbourPatch sides_half1;
      faces ((4 5 6 7)(6 10 11 7));
   }
   sides_half1
   {
      type cyclic;
      neighbourPatch sides_half0;
      faces ((0 3 2 1)(2 3 9 8));
   }
   inlet
   {
      type mappedPatch;
      offset (4.0 0 0);
      sampleRegion region0;
      sampleMode nearestCell;
      samplePatch none;
      faces ((0 3 7 4)(3 9 11 7));
   }
   outlet
   {
      type patch;
      faces ((2 1 5 6)(2 6 10 8));
   }
);
```

Initial and boundary conditions

File A.2: /0/U

```
dimensions [0 1 -1 0 0 0 0];
internalField uniform (0.1335 0 0); // Re_tau = 395
boundaryField
{
   frontWall
   {
      type fixedValue;
      value uniform ( 0 0 0 );
   }
   backWall
   {
      type fixedValue;
      value uniform ( 0 0 0 );
   }
   sides_half0
   {
      type cyclic;
   }
   sides_half1
   {
      type cyclic;
   }
   inlet
   {
      type mapped;
      value uniform (0.1335 0 0); // adjust to values from $internalField
      interpolationScheme cell;
      setAverage true;
      average (0.1335 0 0); // adjust to values from $internalField
   }
   outlet
   {
      type inletOutlet;
      inletValue uniform (0 0 0);
      value uniform (0 0 0);
   }
}
```

File A.3: /0/p

```
dimensions [0 2 -2 0 0 0 0];
internalField uniform 0;
boundaryField
{
   frontWall
   {
      type zeroGradient;
   }
   backWall
   {
      type zeroGradient;
   }
   sides_half0
   {
      type cyclic;
   }
   sides_half1
   {
      type cyclic;
   }
   inlet
   {
      type zeroGradient;
   }
   outlet
   {
      type fixedValue;
      value uniform 0;
   }
}
```

File A.4: /0/nut

```
dimensions [0 2 -1 0 0 0 0];
internalField uniform 0;
boundaryField
{
   frontWall
   {
      type zeroGradient;
   }
   backWall
   {
      type zeroGradient;
   }
   sides_half0
   {
      type cyclic;
   }
   sides_half1
   {
      type cyclic;
   }
   inlet
   {
      type calculated;
      value uniform 1e-8;
   }
   outlet
   {
      type calculated;
      value uniform 1e-8;
   }
}
```

Flow properties

File A.5: /constant/transportProperties

```
Ubar [0 1 -1 0 0 0 0] (0.1335 0 0);
transportModel Newtonian;
nu [0 2 -1 0 0 0 0] 2e-05;
```

File A.6: /constant/turbulenceProperties

```
simulationType LES;
LES
{
   LESModel WALE;
   turbulence on;
   printCoeffs on;
   delta cubeRootVol;
   cubeRootVolCoeffs
   {
      deltaCoeff 1;
   }
   PrandtlCoeffs
   {
      delta cubeRootVol;
      cubeRootVolCoeffs
      {
         deltaCoeff 1;
      }
      smoothCoeffs
      {
         delta cubeRootVol;
         cubeRootVolCoeffs
         {
            deltaCoeff 1;
         }
         maxDeltaRatio 1.1;
      }
      Cdelta 0.158;
   }
   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;
   }
}
```

Numerics

File A.7: /system/controlDict

```
application rk3fracsFoam;
startFrom latestTime;
startTime 0;
stopAt endTime;
endTime 10000;
deltaT 0.1;
adjustTimeStep yes;
writeControl adjustableRunTime;
writeInterval 1000;
purgeWrite 3;
writeFormat ascii;
writePrecision 6;
writeCompression off;
timeFormat general;
timePrecision 6;
runTimeModifiable true;
maxCo 0.6;
maxDeltaT 0.2;
functions
{
   Q
   {
      type Q;
      libs ("libfieldFunctionObjects.so");
      writeControl writeTime;
   }
   vorticity
   {
      type vorticity;
      libs ("libfieldFunctionObjects.so");
      writeControl writeTime;
   }
   yPlus
   {
      type yPlus;
      libs ("libfieldFunctionObjects.so");
      writeControl writeTime;
   }
   fieldAverage1
   {
      type fieldAverage;
      libs ("libfieldFunctionObjects.so");
      writeControl writeTime;
      timeStart 1000;
      fields
      (
         U
         {
            mean on;
            prime2Mean on;
            base time;
         }
         p
         {
            mean on;
            prime2Mean on;
            base time;
         }
      );
   }
}
```

File A.8: /system/fvSchemes

```
ddtSchemes
{
   default backward;
}
gradSchemes
{
   default Gauss linear;
}
divSchemes
{
   default none;
```

```
   div(phi,U) Gauss linear;
   div(U) Gauss linear;
   div(phi,k) Gauss limitedLinear 0.1;
   div(phi,B) Gauss limitedLinear 0.1;
   div(B) Gauss linear;
   div(phi,nuTilda) Gauss limitedLinear 0.1;
   div((nuEff*dev2(T(grad(U))))) Gauss linear;
}
laplacianSchemes
{
   default Gauss linear uncorrected;
}
interpolationSchemes
{
   default linear;
}
snGradSchemes
{
   default uncorrected;
}
```

File A.9: /system/fvSolution

```
solvers
{
   p
   {
      solver GAMG;
      tolerance 0;
      relTol 0.08;
      smoother GaussSeidel;
   }
   pFinal
   {
      $p;
      smoother DICGaussSeidel;
      tolerance 1e-06;
      relTol 0;
   }
   "(U|k|nuTilda)"
   {
      solver smoothSolver;
      smoother symGaussSeidel;
      tolerance 1e-05;
      relTol 0.08;
   }
   "(U|k|nuTilda)Final"
   {
      $U;
      tolerance 1e-05;
      relTol 0;
   }
}
PISO
{
   nOuterCorrectors 3;
   nCorrectors 1;
   nNonOrthogonalCorrectors 0;
}
```

A.2 Turbulent flow over a backward-facing step

Geometry

File A.10: /system/blockMeshDict

```
convertToMeters 0.001;
vertices
(
   (-98.0 9.8 19.6) (-98.0 9.8 -19.6) (-98.0 20.5 19.6) (-98.0 20.5 -19.6) (-98.0 58.8 19.6)
   (-98.0 58.8 -19.6) (0 0 19.6) (0 0 -19.6) (0 9.8 19.6) (0 9.8 -19.6) (0 20.5 19.6)
   (0 20.5 -19.6) (0 58.8 19.6) (0 58.8 -19.6) (24.5 0 19.6) (24.5 0 -19.6) (24.5 9.8 19.6)
   (24.5 9.8 -19.6) (24.5 20.5 19.6) (24.5 20.5 -19.6) (24.5 58.8 19.6) (24.5 58.8 -19.6)
   (196 0 19.6) (196 0 -19.6) (196 9.8 19.6) (196 9.8 -19.6) (196 20.5 19.6) (196 20.5 -19.6)
   (196 58.8 19.6) (196 58.8 -19.6)
);
blocks
(
   hex (0 8 9 1 2 10 11 3) (83 38 29) simpleGrading (1.0 1.0 10.0)
   hex (2 10 11 3 4 12 13 5) (83 38 36) simpleGrading (1.0 1.0 1.0)
   hex (6 14 15 7 8 16 17 9) (45 38 30) simpleGrading (8.0 1.0 5.0)
   hex (8 16 17 9 10 18 19 11) (45 38 29) simpleGrading (8.0 1.0 10.0)
   hex (10 18 19 11 12 20 21 13) (45 38 36) simpleGrading (8.0 1.0 1.0)
   hex (14 22 23 15 16 24 25 17) (135 38 30) simpleGrading (1.0 1.0 5.0)
   hex (16 24 25 17 18 26 27 19) (135 38 29) simpleGrading (1.0 1.0 10.0)
   hex (18 26 27 19 20 28 29 21) (135 38 36) simpleGrading (1.0 1.0 1.0)
);
boundary
(
   frontWall
   {
      type wall;
      faces ((0 1 9 8) (8 9 7 6) (6 7 15 14) (14 15 23 22));
   }
   backWall
   {
      type symmetryPlane;
      faces ((4 12 13 5) (12 20 21 13) (20 28 29 21));
   }
   sides_half0
   {
      type cyclic;
      neighbourPatch sides_half1;
      faces ((0 8 10 2) (2 10 12 4) (6 14 16 8) (8 16 18 10) (10 18 20 12)
                  (14 22 24 16) (16 24 26 18) (18 26 28 20));
   }
   sides_half1
   {
      type cyclic;
      neighbourPatch sides_half0;
      faces ((1 9 11 3) (3 11 13 5) (7 15 17 9) (9 17 19 11) (11 19 21 13)
                  (15 23 25 17) (17 25 27 19) (19 27 29 21));
   }
   inlet0
   {
      type mappedPatch;
      offset (0.0784 0 0);
      sampleRegion region0;
      sampleMode nearestCell;
      samplePatch none;
      faces ((0 1 3 2));
   }
   inlet1
   {
      type mappedPatch;
      offset (0.0784 0 0);
      sampleRegion region0;
      sampleMode nearestCell;
      samplePatch none;
      faces ((2 3 5 4));
   }
   outlet
   {
      type patch;
      faces ((22 24 25 23) (24 26 27 25) (26 28 29 27));
   }
);
```

Initial and boundary conditions

File A.11: /0/U

```
dimensions [0 1 -1 0 0 0 0];
internalField uniform (7.5 0 0);
boundaryField
{
   frontWall
   {
      type fixedValue;
      value uniform ( 0 0 0 );
   }
   backWall
   {
      type symmetryPlane;
   }
   sides_half0
   {
      type cyclic;
   }
   sides_half1
   {
      type cyclic;
   }
   inlet0 // lower inlet
   {
      type mapped;
      value uniform (6.54 0 0);
      interpolationScheme cell;
      setAverage true;
      average (6.54 0 0);
   }
   inlet1 // upper inlet
   {
      type fixedValue;
      value uniform (7.70 0 0);
   }
   outlet
   {
      type inletOutlet;
      inletValue uniform (0 0 0);
      value uniform (0 0 0);
   }
}
```

File A.12: /0/p

```
dimensions [0 2 -2 0 0 0 0];
internalField uniform 0;
boundaryField
{
   frontWall
   {
      type zeroGradient;
   }
   backWall
   {
      type symmetryPlane;
   }
   sides_half0
   {
      type cyclic;
   }
   sides_half1
   {
      type cyclic;
   }
   inlet0
   {
      type zeroGradient;
   }
   inlet1
   {
      type zeroGradient;
   }
   outlet
   {
      type fixedValue;
```

```
      value uniform 0;
   }
}
```

File A.13: /0/nut

```
dimensions [0 2 -1 0 0 0 0];
internalField uniform 0;
boundaryField
{
   frontWall
   {
      type zeroGradient;
   }
   backWall
   {
      type symmetryPlane;
   }
   sides_half0
   {
      type cyclic;
   }
   sides_half1
   {
      type cyclic;
   }
   inlet0
   {
      type calculated;
      value uniform 1e-8;
   }
   inlet1
   {
      type calculated;
      value uniform 1e-8;
   }
   outlet
   {
      type calculated;
      value uniform 1e-8;
   }
}
```

Flow properties

File A.14: /constant/transportProperties

```
Ubar [0 1 -1 0 0 0 0] (7.72 0 0);
transportModel Newtonian;
nu [0 2 -1 0 0 0 0] 1.5e-05;
```

File A.15: /constant/turbulenceProperties

```
simulationType LES;
LES
{
   LESModel WALE;
   turbulence on;
   printCoeffs on;
   delta cubeRootVol;
   cubeRootVolCoeffs
   {
      deltaCoeff 1;
   }
   PrandtlCoeffs
   {
      delta cubeRootVol;
      cubeRootVolCoeffs
      {
         deltaCoeff 1;
      }
      smoothCoeffs
      {
         delta cubeRootVol;
         cubeRootVolCoeffs
```

```
            {
                deltaCoeff 1;
            }
            maxDeltaRatio 1.1;
        }
        Cdelta 0.158;
    }
    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;
    }
}
```

Numerics

File A.16: /system/controlDict

```
application rk3fracsFoam;
startFrom latestTime;
startTime 0;
stopAt endTime;
endTime 6.0;
deltaT 0.5e-5;
adjustTimeStep no;
writeControl runTime;
writeInterval 0.01;
purgeWrite 11;
writeFormat ascii;
writePrecision 6;
writeCompression off;
timeFormat general;
timePrecision 6;
runTimeModifiable true;
functions
{
   Q
   {
      type Q;
      libs ("libfieldFunctionObjects.so");
      writeControl writeTime;
   }
   vorticity
   {
      type vorticity;
      libs ("libfieldFunctionObjects.so");
      writeControl writeTime;
   }
   yPlus
   {
      type yPlus;
      libs ("libfieldFunctionObjects.so");
      writeControl writeTime;
   }
   fieldAverage1
   {
      type fieldAverage;
      libs ("libfieldFunctionObjects.so");
      writeControl writeTime;
      fields
      (
         U
         {
            mean on;
            prime2Mean on;
            base time;
         }
         p
         {
            mean on;
            prime2Mean on;
            base time;
         }
      );
   }
}
```

File A.17: /system/fvSchemes

```
ddtSchemes
{
   default backward;
}
gradSchemes
{
   default Gauss linear;
}
divSchemes
{
   default none;
   div(phi,U) Gauss linear;
   div(U) Gauss linear;
   div(phi,k) Gauss limitedLinear 0.1;
```

```
   div(phi,B) Gauss limitedLinear 0.1;
   div(B) Gauss linear;
   div(phi,nuTilda) Gauss limitedLinear 0.1;
   div((nuEff*dev2(T(grad(U))))) Gauss linear;
}
laplacianSchemes
{
   default Gauss linear uncorrected;
}
interpolationSchemes
{
   default linear;
}
snGradSchemes
{
   default uncorrected;
}
```

File A.18: /system/fvSolution

```
solvers
{
   p
   {
      solver GAMG;
      tolerance 0;
      relTol 0.06;
      smoother GaussSeidel;
   }
   pFinal
   {
      $p;
      smoother DICGaussSeidel;
      tolerance 1e-06;
      relTol 0;
   }
   "(U|k|B|nuTilda)"
   {
      solver smoothSolver;
      smoother symGaussSeidel;
      tolerance 1e-05;
      relTol 0.06;
   }
   "(U|k|B|nuTilda)Final"
   {
      $U;
      tolerance 1e-05;
      relTol 0;
   }
}
PISO
{
   nOuterCorrectors 3;
   nCorrectors 2;
   nNonOrthogonalCorrectors 0;
}
```