NTNU
Norwegian University of
Science and Technology

Master's thesis

2019

Sondre Relling

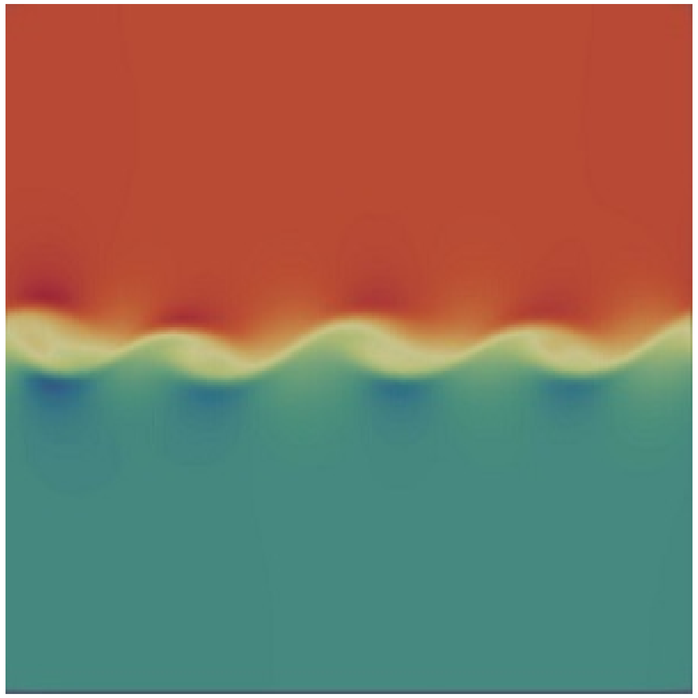
Master's thesis

NTNU
Norwegian University of
Science and Technology
Faculty of Engineering
Department of Energy and Process Engineering

Sondre Relling

Development of a low-dissipation solver for large eddy simulation based on OpenFOAM®

June 2019



NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Development of a low-dissipation solver for large eddy simulation based on OpenFOAM®

Sondre Relling

Mechanical Engineering
Submission date: June 2019
Supervisor: Terese Løvås
Co-supervisor: Tian Li

Norwegian University of Science and Technology
Department of Energy and Process Engineering

Norwegian University of Science and Technology

Faculty of Engineering
Department of Energy and Process Engineering

EPT-P-2018-97

NTNU

PROJECT WORK

for

student Sondre Relling

Autumn 2018

Development of a low-dissipation solver for large-eddy simulation based on OpenFOAM
Implementation of the Runge-Kutta projection method

Background and objective

Open source Field Operation And Manipulation (OpenFOAM$^{@}$) is a C++ toolbox for the development of customized numerical solvers, and pre-/post-processing utilities for the solution of continuum mechanics problems, including computational fluid dynamics (CFD).
Benefiting from its wide variety of implemented turbulent models and its high customizability, OpenFOAM$^{@}$ has been extensively used to simulate complex fluid flows across most areas of engineering and science, from both commercial and academic organizations. Like in other typical commercial codes, such as Ansys Fluent$^{@}$ and STAR-CD$^{@}$, OpenFOAM$^{@}$ also primarily utilizes low-order integration and discretization schemes in order to get robust simulations on low quality meshes. The low-order methods are acceptable in most of the Reynolds-averaged Navier–Stokes (RANS) simulations. However, high-order time-integration and spatial discretization methods are preferred for ensuring minimal influence of numerical diffusion and dispersion on the flow in large-eddy simulation (LES).
Vuorinen et al. (Computers & Fluids 2014(93), 153–163) has developed a low-dissipative Runge–Kutta projection methods that can be used together with the standard pressure correction approach PISO (Pressure Implicit with Splitting of Operators) in OpenFOAM$^{@}$. This master project aims at implementing such high-order methods into the OpenFOAM$^{@}$ version 5.0. The developed LES solver will be tested in various numerical experiments to investigate its performance including a direct comparison to the widely used second order time integration methods. The project student will be granted access to High-Performance Computing (HPC) resources at NTNU.

The following tasks are to be considered:

1. Be familiar with the general CFD theories and the common time integration schemes.
   - Understand the principal concepts of CFD
   - Understand the common numerical methods for solution of ODEs
2. Be familiar with the default OpenFOAM solvers and tools.
   - Understand the file structure
   - Understand how to use the pre- and post-processing tools, such as blockMesh, paraFoam, postprocessing command line interface, sampling and monitoring data
   - Understand how to use pisoFoam solver
   - Perform simulation of a lid-driven cavity flow and compare the results with literature data (V. Vuorinen et al. Computers & Fluids 93, 153,163)
3. Be able to construct and run own solver in OpenFOAM.

- Construct a transient incompressible solver using Runge-Kutta projection methods
- Compare results with the default pisoFoam solver using the lid-drive cavity flow case (V. Vuorinen et al. Computers & Fluids 93, 153,163)
- Compare results with the default pisoFoam solver using the Taylor Green vortices case (V. Vuorinen et al. Computers & Fluids 93, 153,163)

-- " --

The project work comprises 15 ECTS credits.

The work shall be edited as a scientific report, including a table of contents, a summary in Norwegian, conclusion, an index of literature etc. When writing the report, the candidate must emphasise a clearly arranged and well-written text. To facilitate the reading of the report, it is important that references for corresponding text, tables and figures are clearly stated both places.
By the evaluation of the work the following will be greatly emphasised: The results should be thoroughly treated, presented in clearly arranged tables and/or graphics and discussed in detail.

The candidate is responsible for keeping contact with the subject teacher and teaching supervisors.

Risk assessment of the candidate's work shall be carried out according to the department's procedures. The risk assessment must be documented and included as part of the final report. Events related to the candidate's work adversely affecting the health, safety or security, must be documented and included as part of the final report. If the documentation on risk assessment represents a large number of pages, the full version is to be submitted electronically to the supervisor and an excerpt is included in the report.

According to "Utfyllende regler til studieforskriften for teknologistudiet/sivilingeniørstudiet ved NTNU" § 20, the Department of Energy and Process Engineering reserves all rights to use the results and data for lectures, research and future publications.

Submission deadline: 21 December 2018.

□ Work to be done in lab (Water power lab, Fluids engineering lab, Thermal engineering lab)
□ Field work

Department for Energy and Process Engineering, *2018*

Terese Løvås

Terese Løvås
Supervisor

Co-Supervisor(s): Tian Li

Master`s Agreement

Faculty	IV - Fakultet for ingeniørvitenskap
Institute	Institutt for energi- og prosessteknikk
Programme code	MTPROD
Course code	194_TEP4925_1

Personal information

Family name, first name	Relling, Sondre
Date of birth	22.02.1994
Email address	sondrrel@stud.ntnu.no

The Master`s thesis

Starting date	15.01.2019
Submission deadline	13.06.2019
Thesis working title	Development of a low-dissipation solver for large-eddy simulation based on OpenFOAM
Thematic description	Open source Field Operation And Manipulation (OpenFOAM@) is a C++ toolbox for the development of customized numerical solvers, and pre-/post-processing utilities for the solution of continuum mechanics problems, including computational fluid dynamics (CFD). Benefiting from its wide variety of implemented turbulent models and its high customizability, OpenFOAM@ has been extensively used to simulate complex fluid flows across most areas of engineering and science, from both commercial and academic organizations. Like in other typical commercial codes, such as Ansys Fluent@ and STAR-CD@, OpenFOAM@ also primarily utilizes low-order integration and discretization schemes in order to get robust simulations on low quality meshes. The low-order methods are acceptable in most of the Reynolds-averaged Navier–Stokes (RANS) simulations. However, high-order time-integration and spatial discretization methods are preferred for ensuring minimal influence of numerical diffusion and dispersion on the flow in large-eddy simulation (LES). Several previous works have demonstrated that the default OpenFOAM solver pisoFoam (an incompressible solver using the pressure implicit with splitting of operator method) has a

	relatively high dissipative nature. This master thesis, therefore, aims at developing an incompressible solver with low numerical diffusion using OpenFOAM. The developed solver will be thoroughly compared to the pisoFoam under various conditions, for example, two- and three-dimensional spatially developing mixing layers. The comparison results as well as the implementation details will be documented in the thesis.

Supervision and co-authors

Supervisor	**Terese Løvås**
Any co-supervisors	**Tian Li**
Any co-authors	

Topics to be included in the Master`s Degree (if applicable)

Guidelines – Rights and Obligations

Purpose

Agreement on supervision of the Master's thesis is a cooperation agreement between the student, supervisor and the department that governs the relationship of supervision, scope, nature and responsibilities.

The master's program and the work of the master's thesis are regulated by the Act relating to universities and university colleges, NTNU's study regulations and current curriculum for the master's program.

Supervision

The student is responsible for

- Agre upon supervision within the framework of the agreement
- Set up a plan of progress for the work in cooperation with the supervisor, including the plan for when the guidance should take place
- Keep track of the number of hours spent with the supervisor
- Provide the supervisor with the necessary written material in a timely manner before the guidance
- Keep the institute and supervisor informed of any delays

The supervisor is responsible for

- Explain expectations of the guidance and how the guidance should take place
- Ensure that any necessary approvals are requested (REC, ethics, privacy)
- Provide advice on the formulation and demarcation of the topic and issue so that the work is feasible within the standard or agreed upon study time
- Discuss and evaluate hypotheses and methods

- Advice on professional literature, source material / data base / documentation and potential resource requirements
- Discuss the presentation (disposition, linguistic form, etc.)
- Discuss the results and the interpretation of them
- Stay informed about the progression of the student's work according to the agreed time and work plan, and follow up the student as needed
- Together with the student, keep an overview of the number of hours spent

The institute is responsible for

- Make sure that the agreement is entered into
- Find and appoint supervisor(-s)
- Enter into an agreement with another department / faculty / institution if there is a designated external supervisor
- In cooperation with the supervisor, keep an overview of the student's progress, an overview of the number of hours spent, and follow up if the student is delayed by appointment
- Appoint a new supervisor and arrange for a new agreement if
    - supervisor will be absent due to research term, illness, travel, etc., and if the student wishes
    - student or supervisor requests to terminate the agreement because one of the parties does not follow it
    - other circumstances make the parties find it appropriate with a new supervisor
- Notify the student when the guidance relationship expires.
- Inform supervisors about the responsibility for safeguarding ethical issues, privacy and guidance ethics
- Should the cooperation between student and supervisor become problematic for one of the parties, a student or supervisor may ask to be freed from the Master`s agreement. In such case, the institute must appoint a new supervisor

This Master`s agreement must be signed when the guidelines have been reviewed.

Signatures

Terese Løvås	Terese Løvås	Sondre Relling
Institute	**Supervisor**	**Student**
place and date	place and date	place and date
Trondheim NTNU 08.06.19	Trondheim NTNU 08.06.19	Trondheim NTNU 08.06.19

PREFACE

This thesis is a part of my master's degree in mechanical engineering, with specialization in fluid mechanics. Written in spring 2019 at the department of Energy and Process Engineering at the Norwegian University of Science and Technology.

I would like to thank both of my supervisors Terese Løvås and Tian Li for excellent guidance and support. Also the advices from my fellow student Simen Havneraas Røstum have been very helpful in this project.

I also want to express my gratitude for the impressive and thorough work in the article ”On the implementation of low-dissipative RungeKutta projection methods for time dependent flows using OpenFOAM” written by Vuorinen *et al.*

Sondre Relling

ABSTRACT

A necessity in computational fluid dynamics (CFD) is to provide accurate results at a reasonable cost. This has been a challenge since the industry started utilizing the engineering tool in 1960's [1]. Due to its complexity, it requires strong computational power. Therefore, it gained popularity along with the growth of computer power during the 90's and until today. The Industry started to take advantage of this tool, replacing expensive experiments and facilities, and the potential is to keep extending this trend.

In order to make that happen, two main things are needed. First a broader knowledge of CFD and the concepts behind it are essential. Therefore, this thesis covers some of the fundamental theory of numerics and fluid mechanics, including a brief study of the OpenFOAM® solver "pisoFoam", which is an iterative transient incompressible solver based on the SIMPLE-algorithm. The application of it is tested in several cases.

Secondly, more low-dissipative methods need to become more commonly available, providing accurate and beneficial results. The theory presented is used to reconstruct the solver in the article: "On the implementation of low-dissipative RungeKutta projection methods for time dependent flows using OpenFOAM®" and implementing it into OpenFOAM® [2].This is a solver with the projection method, and Runge-Kutta method in time. The procedure is presented in this thesis. Verification and a comparison between the results from the article showed that the "pisoFoam" solver and the results from the newly constructed solver were a close match in the lid driven cavity, and therefore strengthens the credibility of the newly constructed solver. Also, a numerical dissipation of energy test was conducted, confirming that the Runge-Kutta projection method had significantly less numerical dissipation. In the temporal mixing layer case, the projection method was able to capture the energy characteristics, such as the 5/3 law, vortex structures, confirming it models the turbulence well.

SAMMENDRAG

Det er en absolutt nødvendighet å ha nøyaktige resultater innenfor numerisk fluidmekanikk(CFD). Dette har vært en utfordring helt siden industrien tok nytte av dette verktøyet på 60-tallet. På grunn av de komplekse og krevende kalkulasjonene, fikk CFD en oppsving på 90-tallet i lag med prosessorene som stadig kunne takle flere kalkulasjoner per sekund. Industrien kunne dermed redusere antall eksperiment og potensiale ligger i å videreføre denne utviklingen.

For å kunne videreføre denne trenden, er det to ting som står sentralt. Den første er at det trengs kunnskap om CFD og teorien bak, altså i hovedsak fluidmekanikk og numerikk. Derfor er teorien brukt i denne masteroppgaven presentert. I tillegg er to sentrale numeriske metoder presentert. En kort oppsummering av stegene til "pisoFoam" er også inkludert. Dette er en standard "solver" i OpenFOAM®, altså en algoritme laget for å løse Navier Stokes ligningene. Det har blitt gjennomført flere tester av denne algoritmen for å danne et sammenligningsgrunnlag.

Det andre er at nøyaktige algoritmer burde være lettere tilgjengelig, slik at resultatene er mer reelle og troverdige. Rekonstruering av en algortime fra artikkelen: "On the implementation of low-dissipative RungeKutta projection methods for time dependent flows using OpenFOAM®" er derfor gjennomført i denne oppgaven. Dette er en eksplisitt algortime, med Runge-Kutta metoden som tidsintegrasjon. Prosedyren og koden for å gjøre dette er presentert i denne masteren, sammen med en verifisering der den blir sammenlignet med tidligere resultater og "pisoFoam". I "lid driven cavity" viste den samme resultater som tidligere arbeid og "pisoFoam". En undersøkelse av bevaring av energi ble testet i "Taylor Green Vortex". I denne oppgaven ble viskositeten satt til null, derfor burde det ikke være noe tap av energi. Den nye algortimen bevarte betydelig mer energi enn "pisoFoam", som bekrefter en mer nøyaktig algoritme. I "temporal mixing layer" klarte

den nye algoritmen blant annet å følge den anerkjente 5/3-loven og hadde kjente turbulente strukturer, som viser at den også kan simulere turbulente strømninger.

CONTENTS

Preface I

Abstract II

Sammendrag III

1. Introduction 2

2. Theory 5

2.1 Explicit and implicit schemes 5

2.1.1 Backward Euler 6

2.1.2 Runge-Kutta method 7

2.1.3 Comments on the implicit and explicit methods 8

2.2 Incompressible Navier Stokes equations 9

2.3 Large eddy simulation 9

2.3.1 Smagorinsky 10

2.3.2 One equation eddy-viscosity model 11

2.3.3 Dynamic one equation eddy-viscosity model 12

2.3.4 WALE-model 12

2.4 Temporal Mixing layer 13

3. Method 15

3.1 Cases 15

3.1.1 The lid driven cavity case 15

3.1.2 2D Taylor Green Vortex 16

3.1.3 Mixing layer 17

3.2 Solvers 20

3.2.1 Projection method and its implementation 20

3.2.2 The OpenFOAM® solver ”pisoFoam” 22

3.3 Modification, pre and post-processing 23

4. Results . . . 26
4.1 The lid driven cavity . . . 26
4.2 2D Taylor Green Vortex . . . 27
4.3 The temporal mixing layer . . . 31
4.3.1 Turbulence behaviour . . . 31
4.3.2 Momentum thickness . . . 36
4.3.3 Energy . . . 36
4.3.4 Numerical error . . . 43
4.3.5 Reynold stresses . . . 43

5. Conclusion . . . 45

Appendices . . . 50

A. Projection method solver RK4 . . . 51

B. Modifying pisoFoam and the projection method solver for the Taylor Green vortex . . . 54

C. Risk assesment . . . 55

1. INTRODUCTION

Describing the behavior of liquids and gases, is the main purpose of computational fluid dynamics (CFD). Applications of this, are essentially in all sections where fluids have an influence, from small scale micro fluid problems to big scale global weather forecasts. Maybe the most obvious and most known example is aerodynamics. Drag forces and air resistance seem to be a source of fascination. Reducing these forces on for example cars, planes and rockets can be experimented with in CFD [3, 4, 5]. These industries have implemented CFD in theirs design phase, recognizing its power and advantages. They also use it in combustion processes like in engines. Simulations are also used to prevent damaged equipment and accidents in the oil industry, for example, ensuring that neither critical pressure is reached nor that the flow induced vibration matches the natural frequency of the pipe. In medical science, hemodynamics is the study of blood flow. CFD has a great influence, simulating and presenting the dynamics of blood flow. Predicting strokes, based on the hemodynamics in the cerebral hemisphere is a research field [6].

An advantage of CFD is the unlimited possibility to sample data from the whole domain, since a value is stored in each cell or cell face. This cannot be done in a real life because of the obvious reasons that you need nearly infinite measuring devices, and this will even disrupt the flow. A full scale test is an expensive arrangement, especially if there is an uncertainty in a design, and changes are likely to occur, requiring multiple experiments. Also, there are limitations in the conditions of the flow, for example the wind speed in a wind tunnel and the tunnels size. This is not the case in CFD, boundary conditions are easily changed, so is the size of your domain. Therefore, in CFD, a full-scale test is not a problem, nor testing in extreme conditions. However, experiments are still a necessity in parts of the industry. With a rapid set up, CFD has already replaced numerous experiments, but has potential to replace even more, this opportunity should be explored.

As one can imagine, the problems can be quite complex and impossible to do by hand. A computer is therefore strongly associated with CFD, which can handle a huge amount of computations per second. The domain is divided into a grid or collection of points, where numerical schemes are used to solve the governing equations, mainly the Navier Stokes equations. This division of physical space and the numerical schemes are an approximation. This leads to the main concern in CFD, namely the accuracy. Like in other aspect in science, CFD is under constant improvement, and a lot of research points out the numerical weaknesses and propose more accurate solvers and models in CFD. Two of those articles are to be researched in this thesis.

How to solve the Navier Stokes equations is the main question in CFD. The amount of alternative solvers is large covering all the aspects of fluid mechanics, from laminar to turbulent, incompressible to compressible and transient to steady state etc. Coupling velocity and pressure is the main task. There are several suggestions on how to do so. In this thesis the projection method [2] and the PISO-algortihm [7] are two methods to be studied. The former is an explicit algorithm decoupling the velocity and pressure. After the temporary velocity field is obtained it is projected back into the decomposition, leaving an equation for the pressure field. The latter algorithm is an iterative method and is used by a default solver in OpenFOAM®, namely "pisoFoam". A drawback of this solver can be numerical dissipation [2]. The dissipation of energy is a great challenge. Therefore, the main goal is to reduce its influence. A solution can be to refine the mesh,i .e increasing the amount of grid points or use a more accurate solver. Not acknowledging this dissipation is a source to get inconsistent result, making CFD an inaccurate tool. An alternative low-dissipative solver of high order is needed.

Whereas high order methods using Large eddy simulations (LES) or Direct numerical simulations (DNS) are more common to find in scientific and academic studies, Low order methods with RANS are still widely used and are often set by default in commercial softwares [2], like OpenFOAM®. If the high order methods were more available it would also benefit the industry [2]. Therefore, an alternative to the widely used PISO-algorithm with 2^{nd} order time integration is proposed in the reference article, and instructions on how to set up the new 4^{th} order projection method is also presented. Higher order than 4 requires more computational steps "n" than the "n^{th}" order[8]. For example an order of 5 requires 6 steps which makes a "4^{th}" order method a natural point to choose, when time efficiency is a factor.

The object of this thesis is to reconstruct the solver in the reference article [2] and present the essential theory, in order to enlighten the opportunities mentioned. Both methods are to be explained further, along with an implicit time scheme and the explicit Runge Kutta method. Thereby a deeper understanding of the solver and how to reconstruct it are presented in this thesis. Verifying is done on a basic lid driven cavity case and the energy dissipation is investigated with the Taylor-Green vortex case. The main case in the thesis is a turbulent mixing layer with LES, which is a classical turbulent free shear flow. LES solves the resolved part of the domain, whereas DNS solves the domain at every scale. Since the small scales are of importance, they need to be modeled in LES. The goal of each model is to replicate the DNS with less computational effort. Therefore a study of how well the subgrid models perform is done, along with a comparison of the solvers. In the mixing layer two streams with different velocities interact, creating a shear layer. Applications of this can be seen in fuel injections in engines [9], where the main goal is to maximize the efficiency. Wakes behind a wind turbine also contain mixing layers [10], where understanding these effects are important because it gives an indication of where the windmills can be located without a negative influence of the other mills.

The setup in this thesis is with periodic boundary conditions which is called ”temporal mixing layer”. The main advantage is saving computational time because the mixing layer develops with time within the domain, opposed to the ”spatial mixing layer” where it develops in the streamwise direction, forcing a long domain. However, one can study the same effects in both cases, making it possible to test the projection method’s ability to capture turbulence characteristics such as the energy cascade, momentum thickness and more, which are presented in the ”Result” chapter.

2. THEORY

2.1 Explicit and implicit schemes

For solving differential equations numerically, there are two main techniques to consider: an explicit approach or an implicit approach. In both categories there are numerous variants to choose from, some of them even combine the two techniques. Crank-Nickolson is such a method, combing backward and forward Euler, which is later used in the mixing layer case. An explicit method can be simply defined by when a future quantity only depends on the current state, opposed to implicit methods where the equation contains elements from the same time step as the desired quantity. Since the explicit cases use extrapolation, the time step becomes a critical quantity. If it is chosen too large, the extrapolation is inaccurate, and a lot of information disappears within the time step. Too small, the computational time grows. Typically in CFD, the "Courant Friedrichs Lewy" condition is used for setting a limitation for the time step compared to the grid spacing [11]:

$$\mathrm{C} = \frac{\mathrm{u}\Delta \mathrm{t}}{\Delta \mathrm{x}} \leqslant \mathrm{Cmax}, \tag{2.1}$$

where *C* is the *Courant* number, and *Cmax* is the maximum Courant-number allowed according to the Courant Friedrichs Lewy condition. Intuitively, equation (2.1) can be understood as how far the information propagates expressed with the propagation speed multiplied with the change in time, divided by the spatial step, i.e. the length of space in between two adjacent grid points. A general concept of this condition in explicit methods is that the distance covered by the advection of information should not exceed the distance between two adjacent grid points. That means a fluid particle is bounded to move from maximum one cell to another within a time step. Otherwise having a fluid particle move through multiple cells can cause divergence [11], hence the condition Cmax = 1. Typically it is set significantly lower, which increases the accuracy [2]. In the implicit case Cmax can typically obtain larger values, since extrapolation is not used, but rather iteration containing information from the next time step. In this chapter, one implicit and one explicit method are to be investigated to get a deeper analysis and understanding of the differences between the methods.

2.1.1 Backward Euler

Backward Euler is an implicit method. Consider this simple ODE, describing a particle position:

$$\frac{\mathrm{dx}}{\mathrm{dt}} = \mathrm{u}(\mathrm{x}, \mathrm{t}). \tag{2.2}$$

To obtain the next x-position of the particle, a discretized version of backward Euler method (2.2) is used [8]:

$$\mathrm{x}^{\mathrm{n}+1} = \mathrm{x}^{\mathrm{n}} + \Delta\mathrm{t} \cdot \mathrm{u}(\mathrm{x}^{\mathrm{n}+1}, \mathrm{t}^{\mathrm{n}+1}), \tag{2.3}$$

where Δt is the step size, in this case change in time. In equation (2.3) elements for step $n + 1$ appear on both sides, i.e. two unknowns. Solving this can be done for instance by iteration techniques, or Newtons method. The local truncation error is $O(\mathit{\Delta} t)^2$ and the global truncation error $O(\mathit{\Delta} t)$, thus making the backward Euler method to a 1_{st} order method [8]. The advantage of the backward Euler method is the large stability region, which can be found by setting:

$$\mathrm{z} = \frac{\Delta\mathrm{t} \cdot \mathrm{u}(\mathrm{x}^{\mathrm{n}+1}, \mathrm{t}^{\mathrm{n}+1})}{\mathrm{x}^{\mathrm{n}+1}}. \tag{2.4}$$

Notice the resemblance between (2.1) and (2.4). Inserting equation (2.4) in (2.3) and rearrange yields:

$$\mathrm{x}^{\mathrm{n}+1} = \frac{\mathrm{x}^{\mathrm{n}}}{1 - \mathrm{z}}. \tag{2.5}$$

To bound the next position of the particle it is clear from (2.5) that the condition: $|1 - z| \leqslant 1$ needs to be satisfied, hence covering the whole complex plane except a region of a circle with radius 1 centered around $(0, 1)$, illustrated in figure 2.1. Thereby the whole left complex plane is covered, which is a preferred quantity for bounded cases and it is generally accepted that it is essential for stiff problems as well [8].

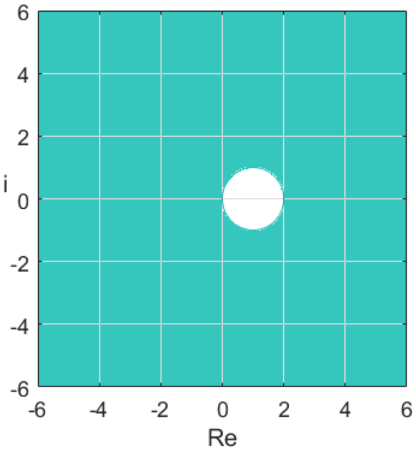


Fig. 2.1: Backward Euler stability region

2.1.2 Runge-Kutta method

There are several different types of Runge-Kutta methods. In this thesis the explicit classical Runge-Kutta method is used. Keeping the same fluid particle case as the previous section, the setup is:

$$\mathrm{x}^{\mathrm{n}+1} = \mathrm{x}^{\mathrm{n}} + \frac{1}{6} \cdot (\mathrm{k}_1 + 2\mathrm{k}_2 + 2\mathrm{k}_3 + \mathrm{k}_4), \tag{2.6}$$

Where:

$$\begin{aligned}
\mathrm{k}_1 &= \Delta \mathrm{t} \cdot \mathrm{u}(\mathrm{x}^{\mathrm{n}}, \mathrm{t}^{\mathrm{n}}), \\
\mathrm{k}_2 &= \Delta \mathrm{t} \cdot \mathrm{u}(\mathrm{x}^{\mathrm{n}} + \frac{\mathrm{k}_1}{2}, \mathrm{t}^{\mathrm{n}}), \\
\mathrm{k}_3 &= \Delta \mathrm{t} \cdot \mathrm{u}(\mathrm{x}^{\mathrm{n}} + \frac{\mathrm{k}_2}{2}, \mathrm{t}^{\mathrm{n}}), \\
\mathrm{k}_4 &= \Delta \mathrm{t} \cdot \mathrm{u}(\mathrm{x}^{\mathrm{n}} + \mathrm{k}_3, \mathrm{t}^{\mathrm{n}}).
\end{aligned}$$

The four k's can be seen as four different estimates of x^{n+1}, where k_1 is computed by forward Euler method. k_2 and k_3 are found by evaluating u at a predicted midpoint by the previous $k's$ respectively k_1 and k_2. The final k_4 is determined by evaluating u at the estimate of k_3's final location. After all these steps, equation (2.6) takes a weighted average to receive the final value of x^{n+1}.Although this method has four times the computational steps of Euler's method, it returns the favor providing a 4_{th} order accuracy.

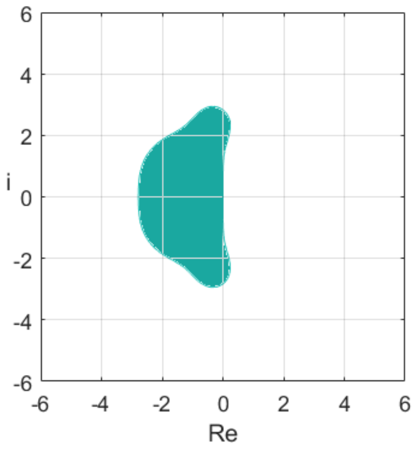


Fig. 2.2: Explicit Runge-Kutta stability region

Following the same procedure as for the Backward Euler method, one can obtain the stability region for the explicit Runge-Kutta method as well [8]. Figure 2.2 presents it. The resulting region clearly restrict the case setup and has to be strictly followed during the computation and the initial settings of the grid and the time step.

2.1.3 Comments on the implicit and explicit methods

Seemingly implicit method should be for choice, due to its preferable stability region. However, it is not that simple. As implicit methods require iterations per time step, simulating a transient flow with a small time step is expensive to calculate. Therefore, explicit methods are generally preferred in this case with less computations per time step. On the other hand, if a steady state solution is expected and the transition is not of great importance, rather large time steps can be used in order to get the final steady state solution. Here an implicit method is generally preferred. Although, it is slower per time step, it can be quicker than the explicit method since it has fewer time steps. Increasing the time step in an explicit method is not an option, as it leads to unstable solutions as previous discussed.

2.2 Incompressible Navier Stokes equations

The incompressible Navier Stokes equation in vector form is as follows:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\nabla \mathrm{p} + \nu \Delta \mathbf{u}. \tag{2.7}$$

Approaching the Navier Stokes numerically, there is a need for coupling the velocity and the pressure. This can be done by taking divergence [2] on both sides of (2.7). Due to the continuity equation:

$$\nabla \cdot \mathbf{u} = 0 \tag{2.8}$$

the unsteady and the viscous term disappear. Therefore, equation (2.7) leads to the following elliptic relationship behavior between the velocity and pressure :

$$\Delta \mathrm{p} = \nabla \cdot ((\mathbf{u} \cdot \nabla)\mathbf{u}). \tag{2.9}$$

2.3 Large eddy simulation

As discussed in the introduction, sometimes there is a need for more accurate solutions than the RANS-modelling can offer. For example in complex geometry cases, DNS is possible to use, but it has an extremely high computational cost, and is therefore often not an option. Fortunately, there is an option that lies in between those two, namely LES. Since the small scale dynamics stand for the majority of the computation time in DNS, there is a potential to only explicitly solve the large scale dynamics, in order to save time [12]. It is also very important that the large scales are dominating the transport equations for mass, momentum and energy. This idea is introduced in LES, with a filter decomposing the velocity field in two terms. The effects of the small scales velocity field are not irrelevant for the flow, therefore they must be modeled. This is a whole field of research and there are various models to choose from, where four of them are used in this thesis. They are briefly introduced in this chapter to present a slight overview of their concepts.

The filter in LES is defined by:

$$\bar{\mathrm{U}}(\mathrm{x}, \mathrm{t}) = \int \mathrm{G}(\mathrm{r}, \mathrm{x}) \cdot \mathrm{U}(\mathrm{x} - \mathrm{r}, \mathrm{t}) \mathrm{dr}. \tag{2.10}$$

The filter function $G(r, x)$ contains a length scale Δ, which decompose the velocity into two components, the large scale velocity $\bar{U}$ and the remaining small velocity residual u', with the following relation:

$$U(x,t) = \bar{U}(x,t) + u'(x,t). \tag{2.11}$$

This is known as the filter width and marks the difference between the resolved domain and the subgrid domain. In default settings in OpenFOAM®, this is the length of a single cell, or more accurately the third root of the volume of the cell if the cell is not cubical.

The filtered momentum equation becomes [12]:

$$\frac{\partial \bar{U}_j}{\partial t} + \frac{\overline{U_j U_i}}{\partial x_i} = -\frac{\partial \bar{p}}{\partial x_j} + \nu \frac{\partial^2 \bar{U}}{\partial x_i \partial x_i}. \tag{2.12}$$

It differs from the Navier Stokes equation (2.7) because the filter product is not equal to the product of the filter velocities:

$$\overline{U_j U_i} \neq \bar{U}_j \bar{U}_i, \tag{2.13}$$

where the difference is the residual stress tensor:

$$\tau_{ij} = \overline{U_j U_i} - \bar{U}_j \bar{U}_i. \tag{2.14}$$

This formulation is a similar reasoning and analogous to the Reynolds stress tensor in RANS, but not the same. Thereby, equation (2.12) and (2.14) yields following momentum equation:

$$\frac{\partial \bar{U}_j}{\partial t} + \frac{\bar{U}_j \bar{U}_i}{\partial x_i} = -\frac{\partial \bar{p}}{\partial x_j} + \nu \frac{\partial^2 \bar{U}}{\partial x_i \partial x_i} - \frac{\partial \tau_{ij}}{\partial x_i}. \tag{2.15}$$

The residual tensor is unknown, and is modelled in order to solve equation (2.15).

2.3.1 Smagorinsky

The classic subgrid model of LES is the "Smagorinsky" model, which is based on the eddy viscosity assumption that proposes a linear relation between the subgrid shear stress and the strain tensor [13]:

$$\tau_{ij} - \frac{1}{3}\tau_{kk}\partial_{ij} = -2\nu_{sg}\bar{S}_{ij}, \tag{2.16}$$

where ν_t is the eddy viscosity and $\bar{S}_{ij}$ is the strain rate tensor given by:

$$\frac{1}{2}\Big(\frac{\partial \bar{U}_i}{\partial x_j} + \frac{\partial \bar{U}_j}{\partial x_j}\Big), \tag{2.17}$$

and the subrgid energy is:

$$\kappa_{sg} = \frac{1}{2}\tau_{kk}, \tag{2.18}$$

and ν_{sg} is computed by the following equation:

$$\nu_{sg} = C_k\sqrt{\kappa_{sg}} \cdot \Delta, \tag{2.19}$$

where C_k has a default value of 0.094, which is also used in this thesis. The subgrid kinetic energy κ_{sg} is calculated assuming a local equilibrium between subgrid production and dissipation:

$$\tau_{ij}\frac{\partial(\overline{U}_j)}{\partial x_j} \; + \; C_\epsilon\frac{\kappa_{sg}^{3/2}}{\Delta} = 0. \tag{2.20}$$

A weakness of the Samgorinsky model is that it can be to dissipative in the laminar region [14], which is caused by the fact of that equation (2.19) outputs an excessive eddy viscosity. From equation (2.20), as long as a velocity gradient is present, it will induce subgrid kinetic energy and therefore produce eddy viscosity through equation (2.19).

2.3.2 One equation eddy-viscosity model

This model follows the same procedure as the Smagorinsky model from equation (2.16) to (2.19). The difference lies in the calculation of the subgrid kinetic energy κ_{sg}. As the name suggests it uses a transport equation instead of assuming local equilibrium. The balance of the subgrid dissipation and production in the "Smagorinsky" model can be questioned, especially because the velocity can be obtained independently of the pressure [15], which contradicts the Navier Stokes equations (2.7). The transport equation is as follows:

$$\frac{\partial(\kappa_{sg})}{\partial t} + \frac{\partial(\overline{U}_j\kappa_{sg})}{\partial x_j} - \frac{\partial}{\partial x_j}\left[(\nu + \nu_{sg})\frac{\partial\kappa_{sg}}{\partial x_j}\right] = -\tau_{ij}\frac{\partial(\overline{U}_j)}{\partial x_j} \; - \; C_\epsilon\frac{\kappa_{sg}^{3/2}}{\Delta}, \tag{2.21}$$

where C_ϵ is an additional constant and has a default value of 1.048. From left to right the terms are referred to as: time derivative, convective, diffusion, production and dissipation term. The later used term "subgrid dissipation" is a part of the production term in this equation. That means subgrid dissipation is a transfer of energy between the resolved domain and the subgrid domain. Hence it appears as a part of the production term in the transport equation of κ_{sg}. The dissipation term in (2.21) is therefore not the same as subgrid dissipation.

2.3.3 Dynamic one equation eddy-viscosity model

It follows the same procedure as the One equation eddy-viscosity model, but the coefficients in the transport equation (2.21) for the subgrid kinetic energy κ_{sg} is dependent on the local flow, i.e C_ϵ is not a constant. The same regards for C_k [16].

2.3.4 WALE-model

WALE model also utilize the eddy viscosity assumption (2.16), but the eddy viscosity is modeled as follows [17]:

$$\nu_{sg} = C_w^2 \Delta^2 \frac{(S_{ij}^d S_{ij}^d)^{\frac{3}{2}}}{(\bar{S}_{ij}\bar{S}_{ij})^{\frac{5}{2}} + (S_{ij}^d S_{ij}^d)^{\frac{5}{4}}} \tag{2.22}$$

where S_{ij}^d is a constructed operator considering the symmetric part of the square velocity gradient tensor. C_w^2 is a constant and in OpenFOAM® set to 0.325. The length scale is set equal to the third root of a single cell $\Delta = V^{\frac{1}{3}}$. This model is often preferred to the Smagorinsky model, due to the eddy viscosity naturally goes to zero at the wall, and also handles the transition between laminar and turbulence better, since the WALE-model also outputs zero eddy viscosity in laminar regions [17].

2.4 Temporal Mixing layer

The temporal mixing layer is a free shear flow, where two parallel stream with opposite velocity profiles pass each other, and shear arises in the interaction zone. This can be seen in figure 2.3 in the transitional region. Where the upper free stream velocity is U_1. Red color indicates velocity rightwards. The lower velocity stream is referred to as U_2, where $U_2 = -U_1$ and has blue colouring in the figure.

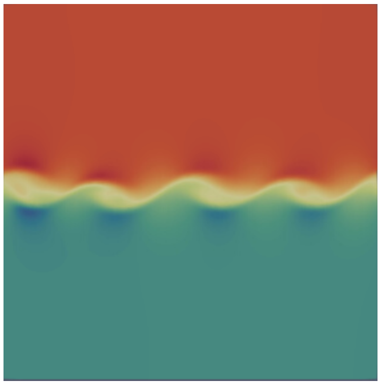


Fig. 2.3: Velocity for the temporal mixing layer in the transition to turbulence

Naturally, the dominant direction is the streamwise direction, which is denoted by x. The normal direction is y, which is the direction responsible for the shear according to newton viscosity law. This and the 2D modes induces the coherent vortices, which is the main feature of the mixing layers. The last is the spanwise direction z, which plays an important part in creating the second instabilities because of initialized longitudinal vortices of 3D modes. A more detailed explanation of the modes can be found later in the chapter "Method".

A mixing layer is known for self similarity [18] from experiments [19, 20], which means the momentum thickness is expected to grow linearly in time when turbulence is present:

$$\delta = \frac{1}{(\Delta U)^2} \int_{-\infty}^{+\infty} (U_1 - U(y)) \cdot (U(y) - U_2)\, dy, \tag{2.23}$$

which is a length, describing the momentum lost due to the dissipation in the mixing layer. Due to the random motion of turbulence and its spreading, the momentum thickness is expected to grow faster in time in the turbulent regime, compared to in the laminar region.

An important length scale for the initial settings and the description of the mixing layer is vorticity thickness, which is defined by:

$$\delta_\omega = \frac{U_1 - U_2}{\frac{\partial U_0}{\partial y}_{max}}, \tag{2.24}$$

which leads to the vorticity thickness Reynolds number :

$$Re_{\delta_\omega} = \frac{(U_1 - U_2) \cdot \delta_\omega}{\nu}. \tag{2.25}$$

It is as all Reynolds number a measurement of the relation between the advection and viscous momentum term and in this case based on a relevant length scale in the mixing layer. It is later used to define the flow in this thesis.

3. METHOD

A description of the cases and how they are set up in OpenFOAM® are to be presented in this chapter. Although the domains are fairly simple with 2D squares and 3D cubical domains, mesh geometry and size should be chosen with caution in CFD. Therefore a verification function of the mesh was executed, and the Courant number were kept at a sufficiently low level. In addition to mesh geometry, boundary conditions are often a source of errors in CFD and are therefore presented and explained. As mentioned in the introduction, low dissipative solvers need to be more commonly available in commercial softwares. How the solver is programmed is therefore presented in this chapter along with a brief description of how the "pisoFoam" solver operates.

3.1 Cases

3.1.1 The lid driven cavity case

The lid driven cavity flow is a well-known benchmark in CFD. The case is set in a 2D enclosed box with the top wall moving at a constant velocity, while the other three walls are fixed. Boundary conditions need to be implemented with awareness of fluid mechanics theory and the specific case. The boundary conditions for the lid driven cavity are presented in table 3.1.

No slip is a well-known assumption, where shear stress from solid walls are far superior to fluid momentum, keeping the fluid at rest. The same regards for the moving wall, but here the no slip condition is implemented with equalizing the velocity of the fluid with the velocity of the moving wall.

Tab. 3.1: Boundary conditions in the lid driven cavity

Property	Fixed Walls	Moving Wall	Front and Back
Velocities	No slip	Fixed Value	Empty
Pressure	Zero gradient	Zero gradient	Empty

Because of the assumption of the flow being attached to the wall, an order of magnitude analysis from boundary layer theory suggest that pressure gradient close to the wall is approximately zero [1]. An intuitive approach to this is to evaluate the velocity component perpendicular to the wall. Due to the no penetration condition and the flow being attached as mentioned, it is close to zero in practical sense. Inserting zero velocity into the Navier Stokes equation (2.7) leads to a zero-pressure gradient. The empty condition is used in OpenFOAM® to reduce the number of dimensions, from three to two. In this case the front and back panels need to be eliminated.

Between the steps in the projection method, the boundary conditions need to be corrected back to its settings. This is done by hard coding [2], with the following syntax:

```
U.correctBoundaryConditions();
```

which is a built-in function in OpenFOAM® that updates the boundary conditions to the ones set in the case files, i.e. the 0-folder, where the initial settings and boundary conditions are in OpenFOAM® .

3.1.2 2D Taylor Green Vortex

A 2D Taylor Green vortex is a case of a unsteady fluid vortex, which can be simulated with cyclic boundary conditions. This periodic phenomenon can be used to study a large or infinite system in an enclosed small domain. The original Taylor-Green vortex reduced to 2D is a problem consisting these initial conditions [21]:

$$\mathrm{u} = \mathrm{A} \cdot \cos(\mathrm{ax})\sin(\mathrm{by}), \tag{3.1}$$

$$\mathrm{v} = \mathrm{B} \cdot \sin(\mathrm{ax})\cos(\mathrm{by}), \tag{3.2}$$

Where u and v is velocity in x and y direction respectively. If $A = 1, a = 1, b = 1, B = -1$, $0 < x < 2\pi$ and $0 < y < 2\pi$, the solution reads [21]:

$$\mathrm{u} = \cos(\mathrm{x})\sin(\mathrm{y})\mathrm{e}^{-2\nu\mathrm{t}}, \tag{3.3}$$

$$\mathrm{v} = -\sin(\mathrm{x})\cos(\mathrm{y})\mathrm{e}^{-2\nu\mathrm{t}}, \tag{3.4}$$

$$\mathrm{p} = -\frac{1}{4}(\cos(2\mathrm{x}) + \cos(2\mathrm{y}))\mathrm{e}^{-4\nu\mathrm{t}}. \tag{3.5}$$

A reason for this case being chosen, is its simple setup along with the possibility of studying a solver's energy conservation. Exit energy enters at opposite side, due to the periodic boundaries. Thus making it possible to monitor the change in total energy, since the energy is kept within the domain.

3.1.3 Mixing layer

The simulations are done in a 3D cubical domain configured like $([0, L], [-\frac{L}{2}, \frac{L}{2}], [0, L])$, where L is approximately set to 4 times the most unstable wavelength according to linear stability theory [14], that is $L \approx 14 \cdot \frac{\delta_\omega}{2}$. Thereby, the domain can contain 4 large vortex rollers and it is possible to observe two different pairings. The grid size is $n = 32^3$ for the coarse mesh and $n = 108^3$ for the fine mesh which should be sufficiently accurate [14]. Uniform spacing has been implemented in x and z directions. In the y-direction a built-in function called "simpleGrading" is used. The input to this function is the expansion rate between the first cell in the block and the last. This is used in order to increase the resolution in the mixing area. The aspect ratio is set to be 2 at the center line. Following the reference article,the velocity profile is initialized by [14]:

$$\frac{U(y)}{U_1} = tanh(\frac{2y}{\delta_\omega}). \tag{3.6}$$

The figure 3.1 shows the initial profile varying with y/L and the length of the arrows corresponds to the value U/U_1. Noise is added as [22]:

$$u' = ar \cdot e^{-y^2}, \tag{3.7}$$

where a is amplitude and chosen to be 0.0001 of the free stream velocity and r is a random function uniformly distributed from -0.5 to 0.5. This ensures that the mean velocity profile (3.6) is intact. In addition the free stream is not disrupted, due to the exponential function eliminates any perturbations in that area. The same procedure for the perturbations is done for the velocities in the remaining directions.

In order to induce turbulence, mixing layer disturbances are imposed with modes with a specific wavelength and a phase shift. This can be initialized with vortices in z direction, which creates Kelvin-Helmholtz instabilities in the xy-plane. Vortices in the streamwise direction is added too, and 3D instabilities occur in the yz-plane. A single disturbance mode in 2D

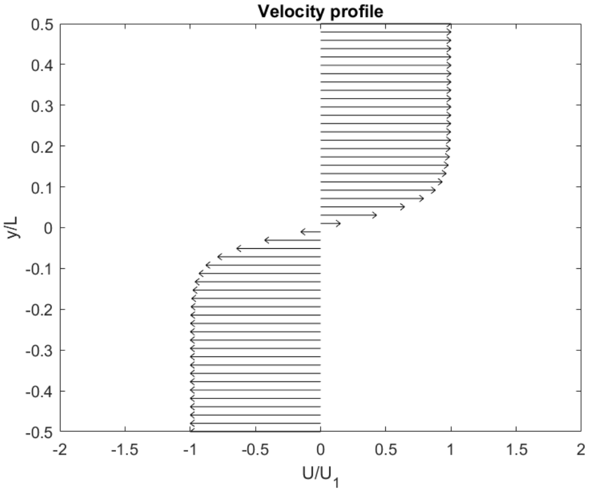


Fig. 3.1: Velocity profile for the temporal mixing layer

is added with a stream function defined as [23]:

$$\psi_z = A \cdot e^{-(\frac{2y}{\delta_\omega})^2} \cdot \sin(k_x \cdot x + \phi_x). \tag{3.8}$$

Taking the appropriate derivatives of equation (3.8) lead to the following disturbances:

$$u' = A \cdot \frac{-8y}{\delta_\omega^2} \cdot e^{-(\frac{2y}{\delta_\omega})^2} \cdot \sin(k_x \cdot x + \phi_x), \tag{3.9}$$

$$v' = A \cdot k_x \cdot e^{-(\frac{2y}{\delta_\omega})^2} \cdot \cos(k_x \cdot x + \phi_x), \tag{3.10}$$

where k_x is the wave number:

$$k_x = \frac{2\pi}{\lambda_x}, \tag{3.11}$$

which describes the frequency of vortices in streamwise direction x (λ is wave length in x-direction). Further on, ϕ_x is the phase shift number, which can relocate the position of the vortices and therefore avoid symmetry if needed. Finally, A is an amplitude set as:

$$A = U_1 \cdot \frac{\lambda_x}{2\pi}, \tag{3.12}$$

which is scaled such that the v' disturbances never exceed the free stream velocity.

Multiple modes are added just by summation of all the individual modes. They are denoted with (α, β) where α and β are scaled wave numbers in x and z direction respectively, so that they represent the number of periods a specific mode provides within the domain. The stream function for multiple modes becomes:

$$\psi_z = e^{-(\frac{2y}{\delta_\omega})^2} \sum_n A_{x,n} \cdot \xi_n \sin\left(k_{x,n} \cdot x + \phi_{x,n}\right), \tag{3.13}$$

and the disturbances are therefore:

$$u' = \frac{-8y}{\delta_\omega^2} \cdot e^{-(\frac{2y}{\delta_\omega})^2} \sum_n A_{x,n} \cdot \xi_n \sin\left(k_{x,n} \cdot x + \phi_x\right), \tag{3.14}$$

$$v' = \cdot e^{-(\frac{2y}{\delta_\omega})^2} \sum_n A_{x,n} \cdot k_{x,n} \cdot \xi_n \cos\left(k_{x,n} \cdot x + \phi_x\right), \tag{3.15}$$

where $0 < \xi_n < 1$ is the individual scaling factor. The 3D-modes are very similar to the 2D modes. So starting off with the stream function with multiple vortices in x-direction reads:

$$\psi_x = e^{-(\frac{2y}{\delta_\omega})^2} \sum_n \sum_m B_{z,m} \cdot \xi_{n,m} \sin\left(k_{x,n} \cdot x + \phi_{x,n,m}\right) \sin\left(k_{z,m} \cdot z + \phi_{z,n,m}\right). \tag{3.16}$$

Hence, $v' = \frac{\partial \psi_x}{\partial z}$ and $w' = \frac{\partial \psi_x}{\partial y}$ become:

$$v' = e^{-(\frac{2y}{\delta_\omega})^2} \sum_n \sum_m B_{z,m} \cdot k_{z,m} \cdot \xi_{n,m} \sin\left(k_{x,n} \cdot x + \phi_{x,n,m}\right) \cos\left(k_{z,m} \cdot z + \phi_{z,n,m}\right), \tag{3.17}$$

$$w' = \frac{-8y}{\delta_\omega^2} \cdot e^{-(\frac{2y}{\delta_\omega})^2} \sum_n \sum_m B_{z,m} \cdot \xi_{n,m} \sin\left(k_{x,n} \cdot x + \phi_{x,n,m}\right) \sin\left(k_{z,m} \cdot z + \phi_{z,n,m}\right). \tag{3.18}$$

B is set with the similar idea to the 2D-case such that the v' disturbances do not exceed the free stream velocity:

$$B = U_1 \cdot \frac{\lambda_z}{2\pi}. \tag{3.19}$$

It is worth pointing out that:

$$A_{x,n} \cdot k_{x,n} = B_{z,m} \cdot k_{z,m} = U_1, \tag{3.20}$$

and whenever these products occurred, it was kept for illustrative purposes.

The main ideas in the reference article is followed [14], but it is not the purpose to replicate the exact results, mainly because simulations are dependent on the initial settings, and for example information about phase shift numbers are withheld in the article. Also, temperature and viscosity modeling are not included in this thesis. The vorticity Reynolds number is set to 200 [14]. Note that its value is 50 there, but that is due to a different definition of the viscosity Reynolds number. Further on, the same modes are chosen, and that is the (4,0), (2,0), (1,0) for the 2D modes and (4,4), (4,-4), (2,2), (2,-2), (1,1), (1,-1) for the 3D modes. In order to break symmetry, phase shift are chosen randomly within the range of $[-\frac{\pi}{2}, -\frac{\pi}{2}]$ for the 3D-modes. Scaling factors are set to 0.05 and 0.15 in 2D and 3D respectively [14].

Tab. 3.2: Boundary conditions in the mixing layer

Property	XZ-Walls	XY-Planes	YZ-planes
Velocities	Slip	cyclic	cyclic
Pressure	Zero gradient	cyclic	cyclic

As one can see from table 3.2 the XY and YZ - planes in the cubical domain are cyclic. This is not physical correct, but is done to simulate an infinite large region. Slip boundary condition has been set at the XZ-walls, where the walls are sufficiently far away to avoid wall effects also known as ground effects. The reason why cyclic boundaries are not implemented at the XZ walls is that properties such as velocity is strongly dependent on y. That means the two different free stream velocity would meet at the boundary and probably create an additional mixing layer. As explained in the lid driven cavity case, pressure is set to zero gradient at the walls.

3.2 Solvers

3.2.1 Projection method and its implementation

It is quite challenging solving Navier Stokes equations numerically as the velocity in the next time step depends on the pressure in the next time step and vice versa. The idea of the Chorin's projection method is based on the procedure from section 2.2 and the Helmholtz-Hodge decomposition theorem, which states that a vector field can be decomposed into the sum of a divergence free part and a curl free part [24]. Applied in this method, one can construct a temporary velocity field:

$$\mathbf{u}^* = \mathbf{u}^{\mathrm{n+1}} + \Delta\mathrm{t}\cdot\nabla\mathrm{p}, \tag{3.21}$$

where $\mathbf{u}$ is the real velocity field and becomes the divergence free part because of the continuity equation (2.8). According to calculus, the curl of a gradient is zero, hence $\nabla\mathrm{p}$ becomes the curl free part of the decomposition. The temporary velocity $\mathbf{u}^*$ be calculated with discretization of equation (2.7) [24] :

$$\mathbf{u}^* = \mathbf{u}^n + \Delta t\left((-\mathbf{u}^n\cdot\nabla)\mathbf{u}^n + \nu\Delta\mathbf{u}^n\right), \tag{3.22}$$

```
dU=runTime.deltaT()*(-fvc::div(phi,U)+...
...turbulence->nuEff()*fvc::laplacian(U));
U=Uold+dU;
```

where the syntax shows the numerical setup of equation (3.22) in OpenFOAM®. ”runTime.deltaT” and ”turbulence-nuEff” and change in time and the sum of eddy viscosity and molecular viscosity respectively. ”div” is naturally divergence and ”laplacian” is divergence squared. Finite volume discretization in OpenFOAM® is done by using ”fvc::” followed by any term that needs to be discretized [25]. Euler method is chosen for simplicity of presenting. The Runge Kutta method is implemented using the same principles, but four times. The idea is to use this temporary velocity field in order to solve the Poisson equation for the pressure, similar to equation (2.9). Taking the divergence of equation (3.21), $\mathbf{u}$ disappears because of continuity, leaving:

$$\Delta\mathrm{p} = \frac{\nabla\cdot\mathbf{u}^*}{\Delta\mathrm{t}}. \tag{3.23}$$

```
solve(fvm::laplacian(p)==fvc::div(U)/runTime.deltaT());
```

The ”fvm::” syntax is finite volume method, where fvm::laplacian(p) returns a coefficient matrix based on the finite volume discretization of the pressure field [25]. After the field is obtained, the pressure and the temporary velocity field is projected back to equation (3.21) to ensure a divergence free velocity field:

$$\mathbf{u}^{\mathrm{n+1}} = \mathbf{u}^* - \nabla\mathrm{p}\cdot\Delta t \tag{3.24}$$

```
U=U-fvc::grad(p)*runTime.deltaT();
```

The flowchart of one time step in the projection method is shown in figure 3.2, and the complete solver can be found in the appendices.

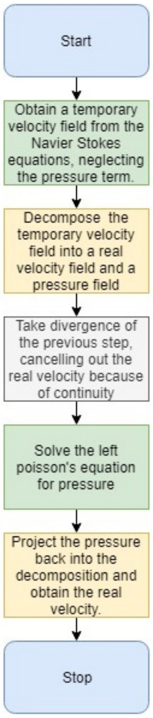

Start
Obtain a temporary velocity field from the Navier Stokes equations, neglecting the pressure term.
Decompose the temporary velocity field into a real velocity field and a pressure field
Take divergence of the previous step, cancelling out the real velocity because of continuity
Solve the left poisson's equation for pressure
Project the pressure back into the decomposition and obtain the real velocity.
Stop

Fig. 3.2: Flowchart of the projection method

3.2.2 The OpenFOAM® solver "pisoFoam"

The PISO (Pressure-Implicit with Splitting of Operators) is an extension of the iterative method SIMPLE with an additional corrector step. A brief summary of the PISO-algorithm can be seen in figure 3.3. The performance of a PISO-solver, i.e how fast it converges, is dependent on the flow case [1], which makes it challenging to generally compare solvers. Therefore, in this thesis, specific cases are to be represented and compared.

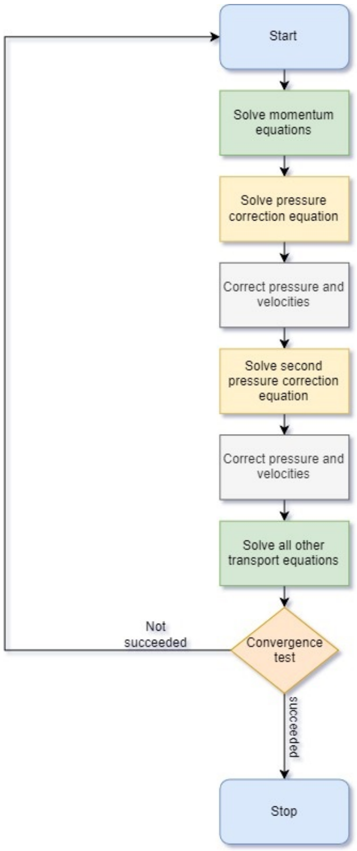

Start
Solve momentum equations
Solve pressure correction equation
Correct pressure and velocities
Solve second pressure correction equation
Correct pressure and velocities
Solve all other transport equations
Convergence test
Not succeeded
succeeded
Stop

Fig. 3.3: Flowchart of the PISO algorithm

3.3 Modification, pre and post-processing

In order to extract results of the desired quantities, some modifications of the built-in functions in OpenFOAM® have been done. The energy decade in the mixing layer case is normalized by the initial energy which is defined before the main loop as:

```
forAll(U,cellI)
   {
       initialEnergy = initialEnergy + 0.5*sqr(U[cellI].x()) + 0.5*
           sqr(U[cellI].y())+0.5*sqr(U[cellI].z());
   }
```

This loops through all cells and adds all the filtered kinetic energy into the total one. In order to detect numerical dissipation, the two viscous terms in the energy equation are calculated directly from the filtered velocity field inside the main loop. Also the kinetic energy from each time step is calculated:

```
forAll(U,cellI)
     {
energy = energy + 0.5*sqr(U[cellI].x()) + 0.5*sqr(U[cellI].y())+0.5*
    sqr(U[cellI].z());
    }
    volScalarField subepsilon(turbulence->nut()*(fvc::grad(U)&& dev(
        twoSymm(fvc::grad(U)))));
forAll(subepsilon,cellI)
    {
        subtotal=subtotal+subepsilon[cellI]*dt;
    }
volScalarField molepsilon(laminarTransport.nu()*(fvc::grad(U)&& dev(
    twoSymm(fvc::grad(U)))));

forAll(molepsilon,cellI)
    {
        moletotal=moletotal+molepsilon[cellI]*dt;
    }
```

here ”energy” is the total kinetic energy for each time step, ”subepsilon” is subgrid dissipation, and ”molepsilon” is molecular dissipation. The subgrid dissipation and the molecular dissipation are the only dissipating terms in the energy equation for the filtered variables [12]. Hence the difference in energy between the steps should be:

$$E_t^n - E_t^{n+1} = E_m^n + E_s^n + E_\xi, \tag{3.25}$$

where subscript ”t” is for total energy, ”m” for molecular dissipation, ”s” for subgrid dissipation, and E_ξ has to be numerical diffusion.

Both the momentum thickness and the Reynold stress are obtained with modification of the built in function ”postChannel”. Momentum thickness (2.23) is programmed as:

```
forAll(UMeanXvalues,I)
  {
  momThick=momThick+(freeStreamU-UMeanXvalues[I])*(freeStreamU-
      UMeanXvalues[I]+70)*(1/sqr(deltaU));
  }
```

where ”UmeanXvalues” are spatial averaged velocities in streamwise and spanwise directions, and the remaining constants are self-explanatory. Fortunately Reynold stresses are already in ”postChannel”, but only filtered and not subgrid. Subgrid Reynold stresses are expressed in equation (2.16) together with (2.18) and can be implemented by this code:

```
volTensorField DU = fvc :: grad (U);
volSymmTensorField DR = -nut*twoSymm (DU) +((2.0/3.0))*k*I ;
```

where DR becomes the sub grid shear tensor and ”nut” is eddy viscosity, ”k” is subgrid kinetic energy and ”I” is the identity matrix.

4. RESULTS

4.1 The lid driven cavity

The vorticity magnitude after the simulation is visualized in figure 4.1. Further on, in this case the cell number is 128^2 and Reynolds number is set to 2500, matching the setup in the reference article [2]. This is a suitable problem for a laminar backward Euler "pisoFoam" solver, since as previously discussed, an implicit method is chosen since a steady state solution is expected. The result of this can be seen in figure 4.2, where the velocity in x-direction on the vertical center line is plotted.

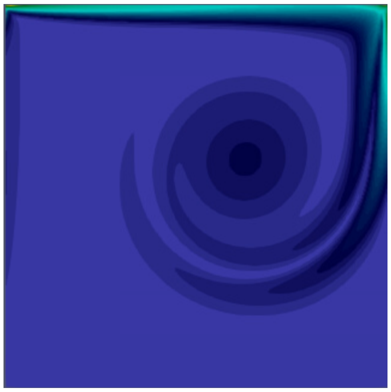


Fig. 4.1: Vorticity magnitude over the whole domain

The projection method was found to correlate well with the results found with the "pisoFoam" solver as seen in figure 4.2. Where a maximum deviation of 0.5% was found relative to the reference velocity. Both cases conform to previous work [2]. The simulations of the cases were running on a grid divided in four sections and parallel computing. The hardware used was: Dell PE630 with 2 x E5-2630 v4 10 cores (20 cores per node), 2.20GHz, 128GB RAM, using only 1 node and 4 tasks per node. The same time step was used in both solvers, therefore the fairness of these results are questionable, because as explained in the theory, implicit schemes have iterations per time step, but can handle a larger time step in return, whereas explicit schemes are limited to the "CFL" condition. Therefore having the time step shorter may favour the explicit scheme, since it can be quicker per time step. These ratios are only used to present a slight overview of the time usage. The times are presented in table 4.1. Where the execution times are normalized to the execution time of the "pisoFoam" solver.

Tab. 4.1: Execution times

Solver	Time
Projection method	0.83
"pisoFoam"	1

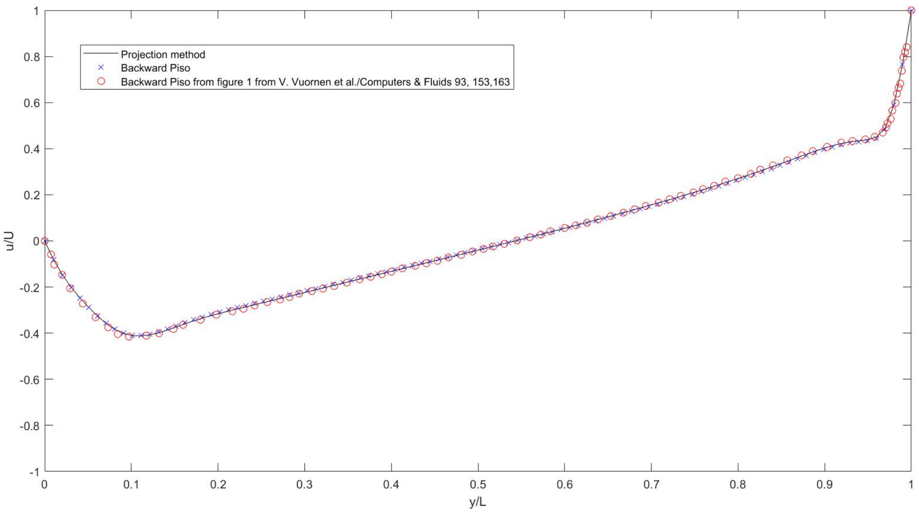


Fig. 4.2: Velocity plotted over the x-center line after the simulation of the lid driven cavity

4.2 2D Taylor Green Vortex

An important note in this case is that the viscosity is set to zero. The only source of energy dissipation is therefore numerical, thus making this an ideal set up for error analysis. An inviscid case yields according to equations (3.3) to (3.5) a steady solution identical to the initial conditions.

All of the original coefficients such as the amplitude and frequency have been set to 1, because of illustrative purposes (except $B = -1$ in equation (3.2)). Keep in mind that as long as consistency is kept, it is possible to flip or reverse the initial settings as desired. This is done in the reference case and therefore also in this thesis, with the initial settings of:

$$\mathrm{u} = \sin(\mathrm{x})\cos(\mathrm{y}), \tag{4.1}$$

$$\mathrm{v} = -\cos(\mathrm{x})\sin(\mathrm{y}), \tag{4.2}$$

$$\mathrm{p} = \frac{1}{4}(\cos(2\mathrm{x}) + \cos(2\mathrm{y})), \tag{4.3}$$

which is also the analytic solution. The grid is set to $n = 128^2$. A visual presentation of this solution is in figure 4.3. Initial settings are set by editing and recompiling both solvers, with the procedure attached in the appendices.

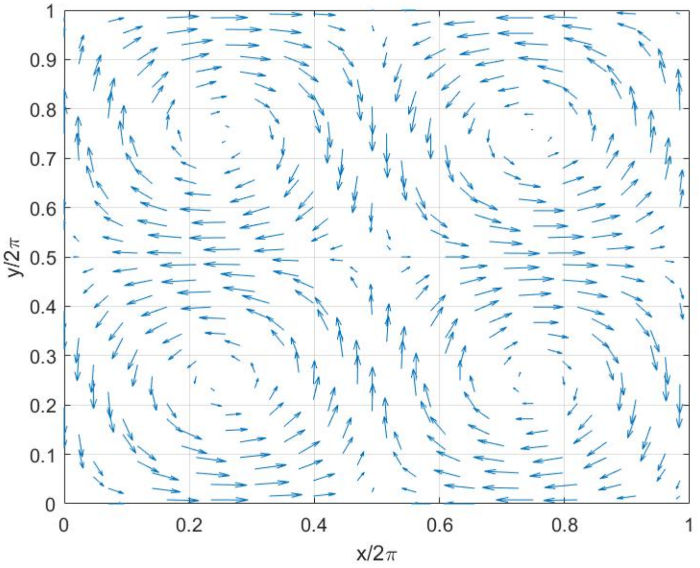


Fig. 4.3: Flow field in the Taylor Green vortices case

A ratio between energy and initial energy is shown in figure 4.4 where the projection method conserved 99.98% and the "pisoFoam" solver 97.220% after ten seconds with the Courant number at maximum around 0.035. Although the Courant number and duration of the simulation do not correspond to the reference case, it still shows the same trend; a significant gap between the solvers. It is stated that the "pisoFoam" solver dissipates by least an order more than the projection method [2]. This is confirmed in this thesis with an energy error of order 10^{-4} compared to 10^{-2} for "pisoFoam". This is found true for the velocity as well, which can be observed in figures 4.5 and 4.6. Notice the change of order in the z-axis.

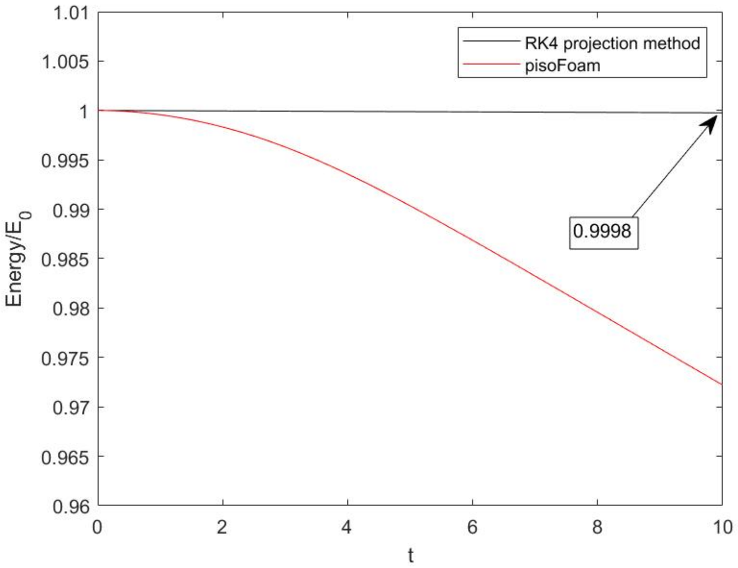


Fig. 4.4: The ratio between initial energy and the current energy

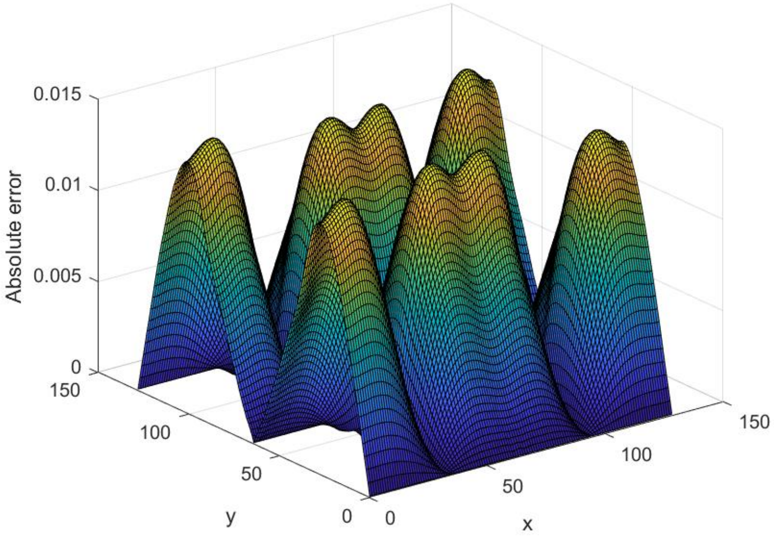


Fig. 4.5: Absolute error of velocity in x-direction for "pisoFoam"

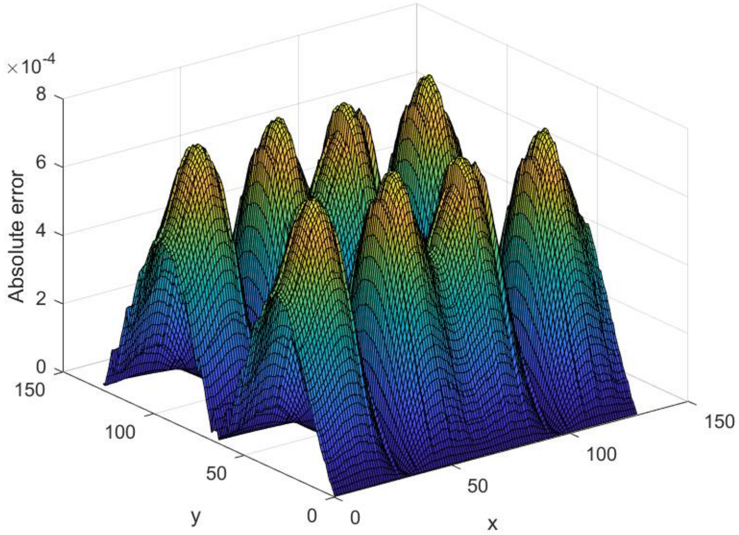


Fig. 4.6: Absolute error of velocity in x-direction for projection method

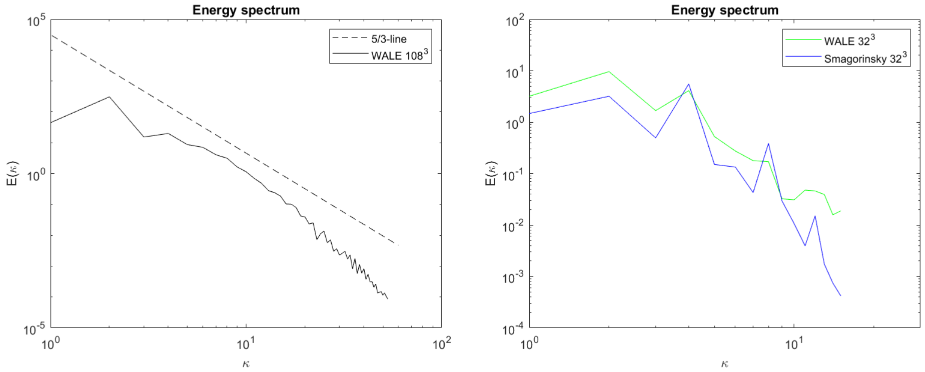
4.3 The temporal mixing layer

Simulations were carried out in a similar manner as described in a previous work [14], but different initial settings, numerical schemes and filter width were used, so the exact same results are not expected. A ratio of about 4/5 in simulation time was found between "pisoFoam" and the projection method, favoring the "pisoFoam" solver. The most computational expensive procedure in the projection method is solving the laplacian equation (3.23) four times. Allthough "pisoFoam" has an iteration process for each step, it is still quicker in this case. In the original work [14], a filter width of twice the cell size was used, but in this thesis it was kept at the default setting in OpenFOAM® as the third root of the cell volume, meaning one cellwidth if the cell is cubical. Four different subgrid models were tested by both solvers on two grids, so 16 different cases in total, hence making it possible to investigate differences between the solvers, mesh dependency, and also between different subgrid models. Of the subgrid models tested in this thesis, the "Smagorinsky" model is the only one present from the reference article [14]. The results from the "Smagorinsky" model are therefore often compared to the ones found in the article. It turns out to be the least accurate model both there and in this thesis.

The numerical scheme used in space is "cubic" which is a $3rd$ order option in OpenFOAM®. Crank Nicholson is used for the "pisoFoam" solver in time, which is second order as mentioned in the introduction. The variable t in this case is a time scale calculated by $t = \delta_\omega/\Delta U$, which is the vorticity thickness divided by the difference between the velocity free streams.

4.3.1 Turbulence behaviour

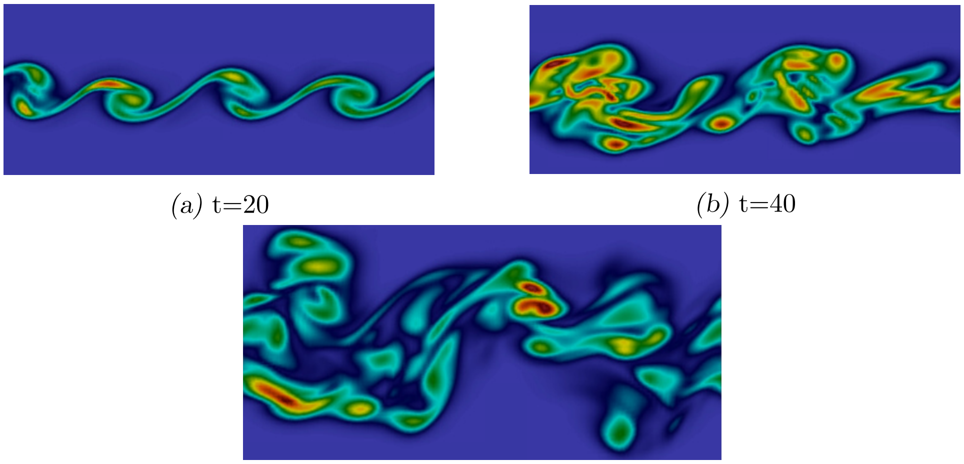
In terms of turbulence behaviour, every solver and subgrid model seem to capture the characteristic energy cascade, meaning large eddies are cascading into smaller ones. This can be seen in figure 4.7, where figure 4.7(a) shows the streamwise energy spectrum for the fine mesh case with the "WALE" model and figure 4.7(b) shows the coarse mesh case. In the figures, nondimensionalization is done by the velocity in the upper stream and the initial vorticity thickness. The "Smagorinsky" model has limited contributions from small scales compared to the "WALE" model. A previous simulation has similar graph for the "Samgorinsky" model where it goes far below the DNS solution [14]. The "WALE" model performs better in the small scale region having more energy resolved. None of the eddy-viscosity



(a) Streamwise energy spectrum, projection method, "WALE", $n = 108^3$, 5/3-law is included.

(b) Streamwise energy spectrum, projection, "Smagorinsky", "WALE", $n = 32^3$.

Fig. 4.7: Energy spectrum

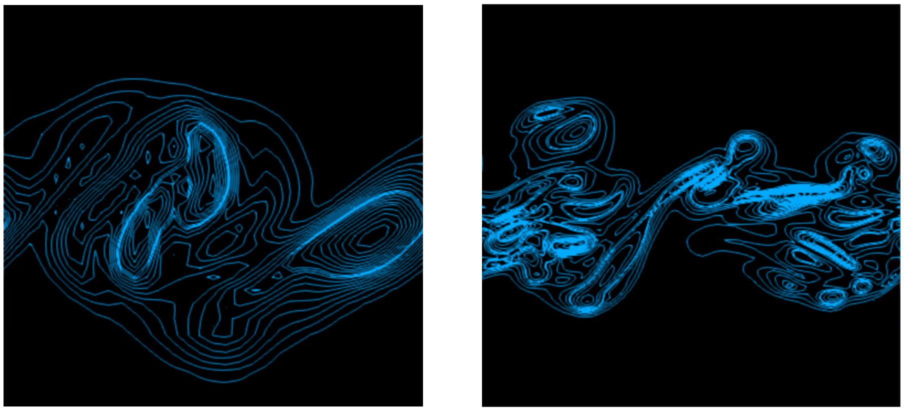


(a) t=20

(b) t=40

(c) t=80

Fig. 4.8: Magnitude of vorticity for three different time steps, Projection method with "WALE" with $n = 108^3$, zoomed in on shear layer and rescaled colors for visual purposes



(a) "One equation" $n = 32^3$

(b) "One equation" $n = 108^3$

Fig. 4.9: Contour plot of the magnitude of the vorticity by the "one equation" subgrid model by projection method at t=80

models reaches the levels of the DNS case in the previous article, however the "Dynamic Smagorinsky" is pretty close. A lack of backscatter, transferring energy from the subgrid to the resolved domain, could be a reason for this. Clearly this happens in a real fluid and in DNS, making a contradiction to the eddy viscosity assumption, where subgrid dissipation is always positive. The result from the fine mesh case can be seen in figure 4.7a. It is approximately parallel to the 5/3 law, and probably would be even closer with a higher Reynolds number [14]. The spectrum shows the typical cascade where energy transfers from large scale eddies into the small scale scales.

As one can see from figure 4.8 the large vortices successfully start to take form at around t=20, which is expected [2]. Pairings start occurring around t=40 which figure 4.8 confirms. At time 80, one large structure is left due to an additional pairing between the structures from $t = 40$. In terms of different solvers, they are fairly similar and it is hard to point out any weaknesses in any of them visually, i.e by the vortex structures. The differences in mesh resolution can be seen in figure 4.9 at $t = 80$. Where the shape is somewhat equal, but the coarse does not capture as much detail as the fine mesh naturally. The coarse mesh predicted only two vortex rollers instead of four in the transition.

The Q-criteron is an acknowledged method to investigate vortex structures and their development [26]. It uses the second invariant of the velocity gradient tensor, Q, which describes the rate between the rotation rate and strain tensor. Both lambda-vortices in figure 4.10 and hairpin-vortices in figure 4.11 are occurring in this simulation. These coherent structures are typical in the mixing layer [26]. The lambda-vortices are initiated by shear layers in the beginning, thereby it is claimed that these vortices evolve into "hairpin"-vortices by stretching in the shear layer. The theory and detailed development of these are not studied in this thesis. The most important in this case, was verifying that the projection method was able to predict these structures in the flow.

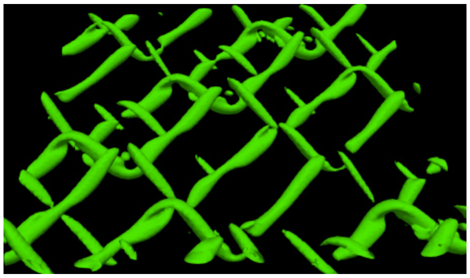


Fig. 4.10: Isosurfaces of Q=10, normalized by the time scale, at t=15, positive streamwise direction is forward into the picture.

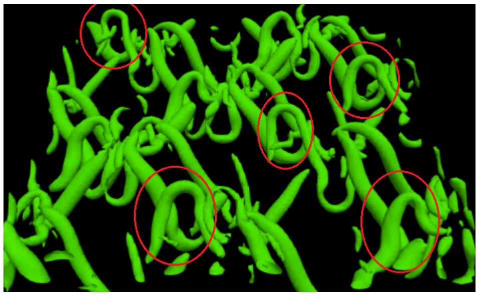


Fig. 4.11: Isosurfaces of Q=10, normalized by the time scale, at t=30, positive streamwise direction is forward into the picture.

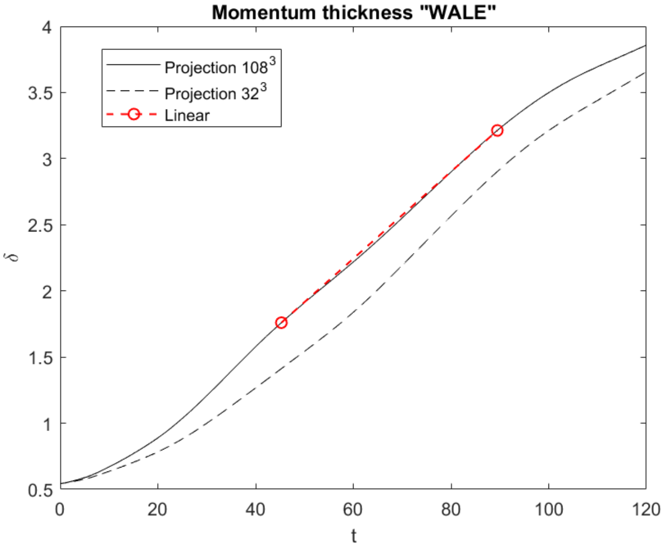


Fig. 4.12: Momentum thickness for the "WALE" model, solid lines represents the $n = 108^3$ mesh and dashed lines are $n = 32^3$.

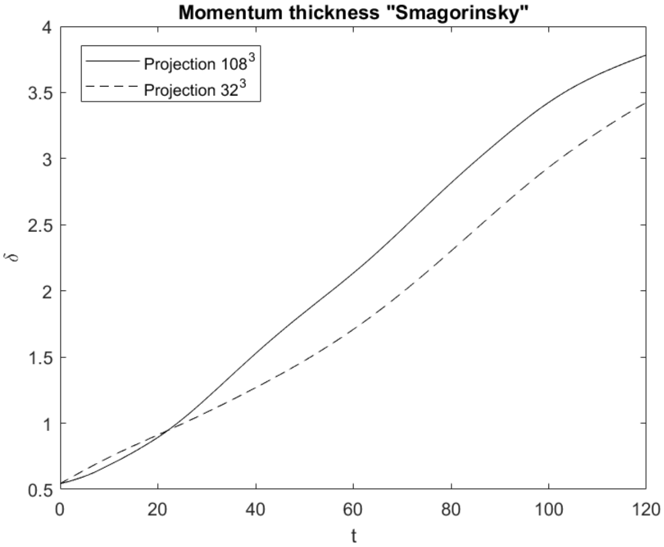


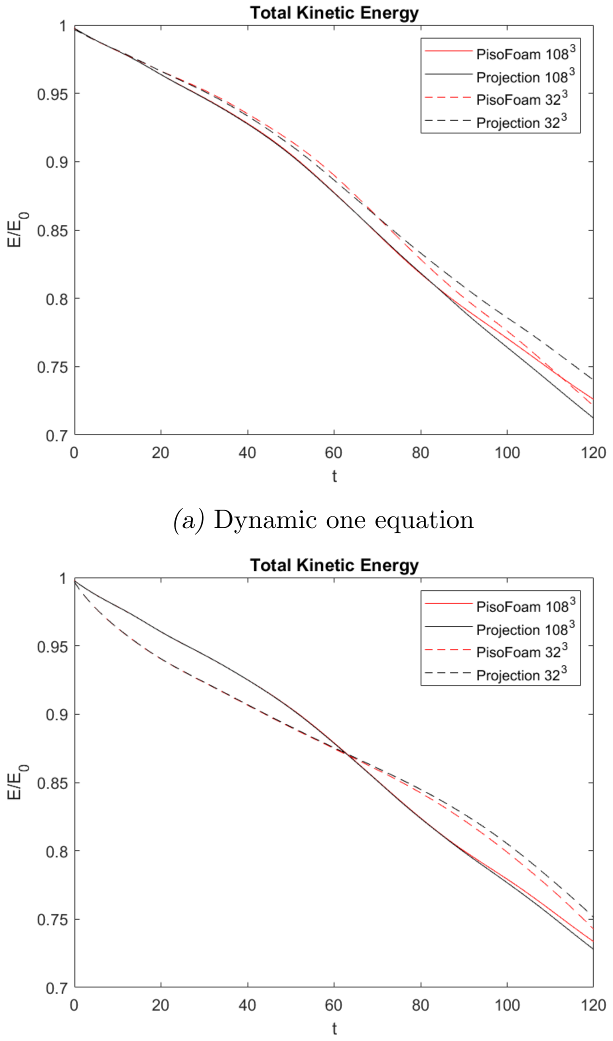
Fig. 4.13: Momentum thickness for the "Smagorinsky" model, solid lines represents the $n = 108^3$ mesh and dashed lines are $n = 32^3$.

4.3.2 Momentum thickness

As described in the theory, momentum thickness is a length scale containing information about how much momentum is lost due to viscous dissipation and spreading of the mean velocity profile. In order to compare with the fine mesh in the figures, the fine mesh momentum thickness is scaled appropriately to the coarse case. Momentum thickness should grow slower in the laminar region and linear in the turbulent region. These effects were captured by the fine mesh as shown in figure 4.12, where the red dashed line is linear. When it comes to the coarse mesh, where subgrid scales have more influence, none of the solvers combined with models had a linear growth in the turbulence region. However, there were differences in the laminar region where the "Smagorinsky" model grows rapidly and surpasses the fine mesh case in figure 4.13, this does not happen in figure 4.12 which is the "WALE" model. This is caused by the overestimated subgrid dissipation and thereby too rapidly spreading of the mean velocity profile. Further on, it grows too slow in the transition and the turbulent region, which indicates that it is hindering turbulence [14]. The "WALE" model however, is closer to the fine mesh case, but clearly grows slower than it should in the turbulence region too, but has a better turbulence modelling since it is not hindering as much spreading as the "Smagorinsky" model. In terms of comparison in between "pisoFoam" and projection method they showed similar results.

4.3.3 Energy

Conservation and dissipation of energy is another benchmark in testing of solvers and models. For the fine mesh, the energy left after a period of 120 timescales ranges from approximately 71% to 74% and for the coarse mesh the range is 72-75%. This is expected as more of the turbulence motion is resolved in the finer mesh causing higher dissipation. The least dissipative model was the "Smagorinsky" model. On the opposite side, the "WALE" and "Dynamic one equation" models were rather similar in being the most dissipative models. They have more molecular dissipation than the "Smagorinsky" and "One equation" models, but lesser subgrid dissipation. A further discussion on this can be read in the molecular and subgrid dissipation sections. The two solvers showed different behaviour on the energy decade. For the coarse mesh, "pisoFoam" has the most dissipation of energy, making a closer fit to the fine mesh case. However, the reason behind it turns out to be numerical dissipation, which means in other cases, "pisoFoam" could be wrong and dissipate more than it should. In the fine mesh



(a) Dynamic one equation

(b) ”Smagorinsky”

Fig. 4.14: Total kinetic energy dissipation

case, the accuracy between the solvers were more similar. However, the projection method has more dissipation, which can indicate better turbulence modeling.

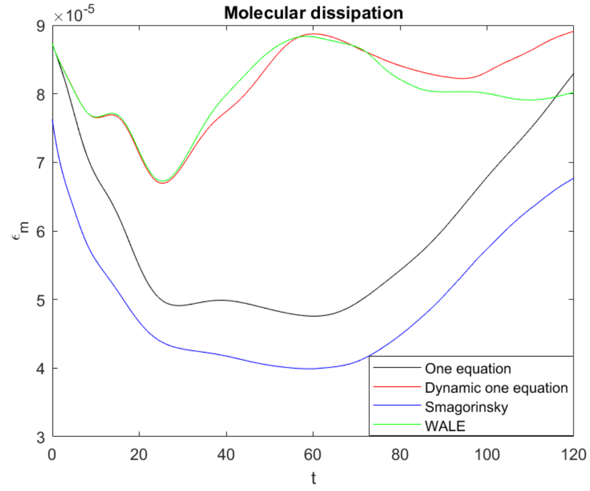


Fig. 4.15: Differences of molecular dissipation between the models on the coarse mesh 32^3 with projection method.

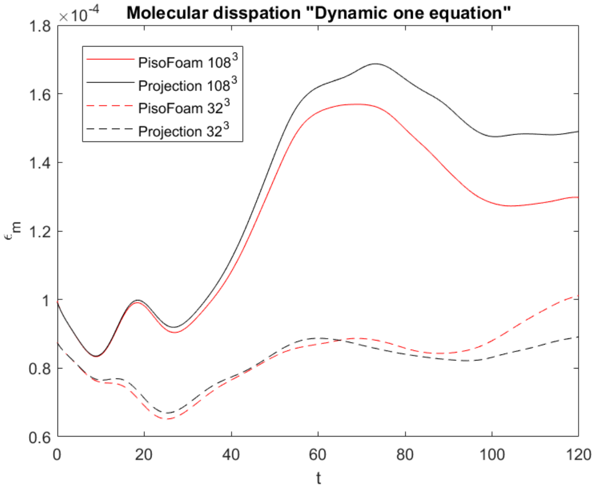


Fig. 4.16: Molecular dissipation for the "Dynamic one equation" model for both solvers and meshes.

Molecular dissipation

Molecular dissipation is caused by the viscous term in the Navier Stokes equations in the resolved region of the domain. All figures are nondimensionalized by the initial energy, i.e they show the amount of energy dissipated and not the rate. This regards for every dissipation figure in this chapter. For the coarse mesh, the molecular dissipation is decreasing for every solver in the start from figure 4.15. A possible explanation is too large gradients due to large grid spacing, causing overestimated dissipation. The fine mesh cases from figure 4.16 have an oscillating curve in the laminar/transition phase and proceeds to grow in the turbulence region. This should be logical due to instabilities and molecular dissipation is known to grow along with turbulence.

The "WALE" and "Dynamic one equation" models have a minor growing phase as turbulence develops seen in figure 4.15. It is not as significant as in the fine mesh case. As the coarse mesh has a different separation point between the resolved and the subgrid domain, it is hard to conclude whether it is growing rapidly enough or not, compared to the fine mesh case. Figure 4.14 suggests that it is not, as it is not able to follow the decreasing energy as in the turbulence region in the fine mesh case, but it could be due to underestimated subgrid dissipation as well.

A convex curve is the result of the "One equation" model and the "Smagorinsky" model. It is possible that this comes from an excessive subgrid dissipation and it overrules the molecular dissipation where it is not supposed to. From around $t = 60$, it finally starts to grow. It seems that these two models have delayed the turbulence influence too far, and the assumption of local equilibrium for the "Smagorinsky model" is more appropriate for a turbulent region rather than in a laminar/transition region. This is a counter-productive behaviour, which hinders the transition to where the model performs the best. Although the "One equation" model has a transport equation for the subgrid kinetic energy, it seems to behave very similar to the "Smagorinsky" model. Most likely, this is due to the dissipation and the production term are the most dominant terms in the transport equation for the subgrid kinetic energy (2.21), creating a resemblance to the "Smagorinsky" model which only contains those two terms. This is exactly the point of the "Dynamic one equation" model, which reduces the dissipation term in the transport equation. This is not to be confused with subgrid dissipation which is a part of the production term in that equation, which is further discussed in the next section.

The ”Smagorinsky” and the ”One equation” model are the furthest away from the profile that the DNS from previous work and the fine mesh case in this thesis displayed [14]. This conclusion is also supported by figure 4.14 where the ”Dynamic one equation” model behaves similar to the fine mesh case, whereas ”Samgorinsky” model differs more. A similar conclusion was drawn in the previous work, where the results were compared to a filtered DNS-solution on the coarse mesh [14].

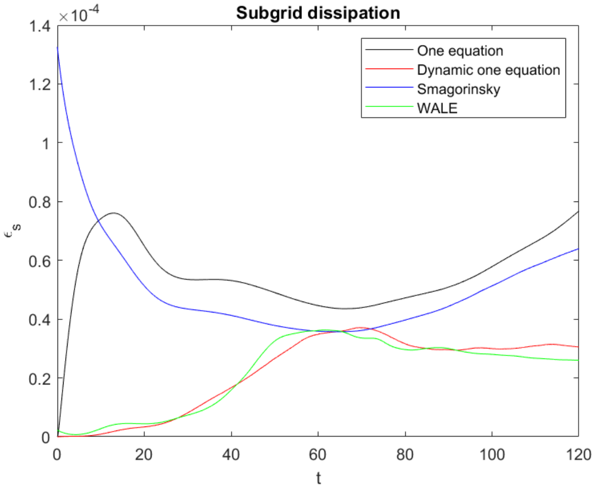


Fig. 4.17: Differences of subgrid dissipation between the models on the coarse mesh 32^3 with projection method

Subgrid dissipation

The models should output minimal eddy viscosity at $t = 0$ and this is successfully done by every model except the ”Smagorinsky” model seen in figure 4.17. It assumes local equilibrium, as mentioned in theory, this means the dissipated subgrid kinetic energy is equal to the production term which contains subgrid dissipation. As long as velocity gradients are present, it produces eddy viscosity. Therefore it is excessive at the beginning of the simulation when large velocity gradients exists. Further on, it decreases in the process of transition to turbulence. Finally, after $t = 70$ it is increasing. This development is almost opposite compared to previous DNS-results [14]. For the ”One equation” model the subgrid dissipation is rightfully minimal

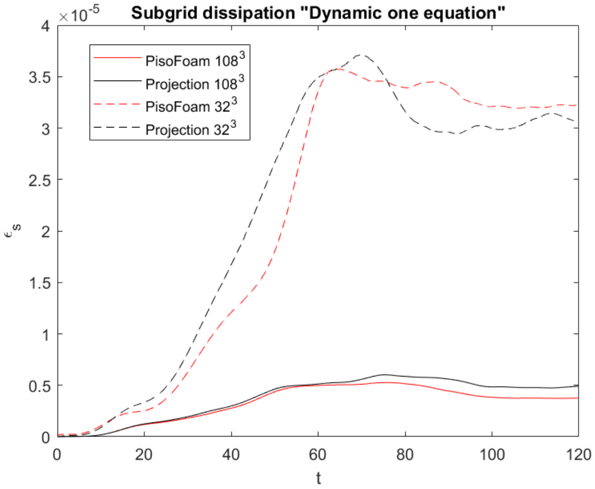


Fig. 4.18: Subgrid dissipation for the "Dynamic one equation" model for both solvers and meshes

at the start, because equation (2.21) is able to output a minimal eddy viscosity. One of the reasons can be that the diffusion term reduces the amount of subgrid kinetic energy, which lessen the influence of the dissipation term (growing proportional with $k_{sg}^{\frac{3}{2}}$), oppose to the "Smagorinsky" model where there is no diffusive term. A dimension analysis of the diffusion versus the dissipation term in the "One equation" and the "Dynamic one equation" models can be derived as:

$$\frac{\partial}{\partial x_j}\left[(\nu+\nu_{sg})\frac{\partial \kappa_{sg}}{\partial x_j}\right] \sim C_\epsilon \frac{\kappa_{sg}^{3/2}}{\Delta} \tag{4.4}$$

$$\frac{(\nu+\nu_{sg})\kappa_{sg}}{\Delta} \sim C_\epsilon \kappa_{sg}^{3/2} \tag{4.5}$$

$$\frac{dissipation}{diffusion} \sim C_\epsilon \Delta \frac{\sqrt{\kappa_{sg}}}{(\nu+\nu_{sg})} \tag{4.6}$$

As one can see from equation 4.6, as subgrid kinetic energy increases, the diffusion term has less influence and the energy has to be produced from the production term through subgrid dissipation. When the amount of subgrid energy for the "One equation" model reaches the same magnitude as the "Smagorinsky" they show similar behaviour, since now the dissipation term is in the same order of magnitude. When C_ϵ is not a constant, but is dependent on the local flow like in the "Dynamic one equation" model, it seems to yield more logical subgrid dissipation. Now equation (4.6) is also influenced by " C_ϵ ". This variable being close to zero in the initial state, results in a low subgrid dissipation and stays that way until the transition to turbulence starts. There it starts to grow as it should, because the energy cascade of turbulence states that energy is transferred down to the small scales. Although the "WALE" model produces eddy viscosity through a algebraic equation using the strain rate and rotation rate, it agrees very well with the "Dynamic one equation" model.

4.3.4 Numerical error

It is important to keep the numerical error sufficiently small in CFD. Conservation of energy is one way to investigate such an error. The only source of dissipation in this case should be either molecular or subgrid dissipation. Therefore if a summation of these two terms is different from the loss of the total filtered kinetic energy, it should be due to numerical issues. This principle is also explained by equation(3.25). Hence figures 4.16 and 4.18 should conform to figure 4.14a. For the projection method, the summation of energy dissipated by the two former figures lead to the numerical value of 0.2473 of the initial energy. The total kinetic energy loss was 0.2568 from figure 4.14. This means that the numerical dissipation is only 3.72% by the projection method, which is in the same order as from previous work [14]. The same procedure yields a total dissipation of 0.2491 and a total kinetic loss of 0.2756 by "pisoFoam", which yields a 9.63% error. It further confirms the result from the Taylor Green vortices case, where the "pisoFoam" solver has significantly more numerical dissipation than the projection method. It seems that the numerical diffusion hinders the "pisoFoam" in developing as much turbulence as the projection method. The lack of subgrid dissipation and molecular dissipation by "pisoFoam" compared to the projection method, support this statement. In the next section about Reynold stress, a consequence of lacking subgrid dissipation is discussed.

4.3.5 Reynold stresses

The Reynold stresses shown in figure 4.19 are exclusively the subgrid stresses. It is normalized by the velocity of upper free stream. The projection method gave higher values for the Reynolds stresses for both solvers. The reason is that the projection method has more energy in the subgrid domain, making the shear stress tensor larger for the projection method cases. A ratio about 3/4 was observed at $t = 70$ for the "Dynamic one equation" model. The resolved scale perturbations were the same for both solvers, therefore the figure shows only the subgrid Reynold stress. This may come from the numerical dissipation by "pisoFoam". Energy that were supposed to be transferred to the small scales, dissipated instead artificially. Thus the Reynolds stresses are larger for the projection method. When it comes to the spreading of the shear layer, they seem to agree very well, that is again no surprise because of the momentum thickness were fairly similar in the solvers too.

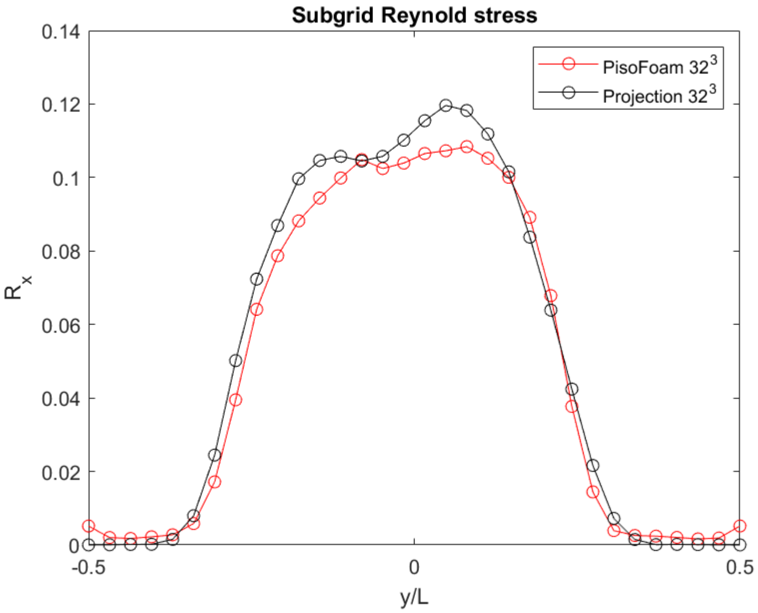


Fig. 4.19: The subgrid Reynold stresses in streamwise (R_{11}) direction at t=70 with "Dynamic one equation" model

5. CONCLUSION

The projection method was successfully reconstructed and matched the results found in the lid driven cavity case with the standard transient solver "pisoFoam" in OpenFOAM®. Both solvers agreed with results from previous simulations of the lid driven cavity [2]. Further on, an agreement of the hypothesis of the projection method being less numerically dissipating is confirmed in the inviscid Taylor Green vortex case. It showed a clear difference, where the projection method preserved 99.98% of the initial energy, whereas the "pisoFoam" solver had only 97.220% left.

A similar outcome was found for the temporal mixing layer case. Two parallel streams interact and create a shear layer, which was studied with periodic boundaries. The projection method had a total error of 3.72% when "pisoFoam" had a 9.67% further supporting the dissipation hypothesis. For subgrid modeling the "Dynamic one equation" and the "WALE" model were found to have the best performance, whereas an overestimated subgrid dissipation were found to be destructive for both the "Smagorinsky" and "One equation" model. As for turbulence behaviour, the projection method performed well and seemed to display the characteristics. For example, known vortex structures were captured in the simulations. Also, the energy spectrum showed tendency to the 5/3 law, which means it captured the typical energy cascade.

As for computational efficiency, the solvers seem to be comparable. The projection method were faster in the lid driven cavity flow, but slower in the temporal mixing layer which is a more complex case. Therefore, whether to use "pisoFoam" or the projcetion method, is a question about time and precision. This means the cost of accuracy is, as expected, measured in the amount of time. However, 25% more computational time should be reasonable, especially having the significant difference in accuracy in mind. However, the same time step was used, which can be favorable for the explicit solver as discussed in the theory part. Further work should asses this, and investigate the accuracy/time ratio of both solvers. Expanding with more cases is also recommended, to investigate the case dependency and further clarify its strengths and weaknesses.

REFERENCES

[1] H. K. Versteeg and W. Malalasekera, *An introduction to computational fluid dynamics - the finite volume method.* Addison-Wesley-Longman, 2007.

[2] V. Vuorinen, J.-P. Keskinen, C. Duwig, and B. Boersma, "On the implementation of low-dissipative runge–kutta projection methods for time dependent flows using openfoam®," *Computers & Fluids*, vol. 93, pp. 153–163, 2014.

[3] W. Kieffer, S. Moujaes, and N. Armbya, "Cfd study of section characteristics of formula mazda race car wings," *Mathematical and Computer Modelling*, vol. 43, no. 11-12, pp. 1275–1287, 2006.

[4] P. Tucker, S. Menon, C. Merkle, J. Oefelein, and V. Yang, "An approach to improved credibility of cfd calculations for rocket injector design," in *43rd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, p. 5572, 2007.

[5] C. Rhie and W. L. Chow, "Numerical study of the turbulent flow past an airfoil with trailing edge separation," *AIAA journal*, vol. 21, no. 11, pp. 1525–1532, 1983.

[6] M. Raschi, F. Mut, G. Byrne, C. M. Putman, S. Tateshima, F. Viñuela, T. Tanoue, K. Tanishita, and J. R. Cebral, "Cfd and piv analysis of hemodynamics in a growing intracranial aneurysm," *International journal for numerical methods in biomedical engineering*, vol. 28, no. 2, pp. 214–228, 2012.

[7] R. I. Issa, "Solution of the implicitly discretised fluid flow equations by operator-splitting," *Journal of computational physics*, vol. 62, no. 1, pp. 40–65, 1986.

[8] J. C. Butcher, *Numerical methods for ordinary differential equations.* John Wiley & Sons, 2016.

[9] B. Jayashankara and V. Ganesan, "Effect of fuel injection timing and intake pressure on the performance of a di diesel engine–a parametric study using cfd," *Energy Conversion and Management*, vol. 51, no. 10, pp. 1835–1848, 2010.

[10] A. Crespo, J. Hernandez, and S. Frandsen, "Survey of modelling methods for wind turbine wakes and wind farms," *Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology*, vol. 2, no. 1, pp. 1–24, 1999.

[11] R. Courant, K. Friedrichs, and H. Lewy, "On the partial difference equations of mathematical physics," *IBM journal of Research and Development*, vol. 11, no. 2, pp. 215–234, 1967.

[12] S. B. Pope and S. B. Pope, *Turbulent flows.* Cambridge university press, 2000.

[13] J. Smagorinsky, "General circulation experiments with the primitive equations: I. the basic experiment," *Monthly weather review*, vol. 91, no. 3, pp. 99–164, 1963.

[14] B. Vreman, B. Geurts, and H. Kuerten, "Large-eddy simulation of the turbulent mixing layer," *Journal of Fluid Mechanics*, vol. 339, pp. 357–390, 1997.

[15] A. Yoshizawa and K. Horiuti, "A statistically-derived subgrid-scale kinetic energy model for the large-eddy simulation of turbulent flows," *Journal of the Physical Society of Japan*, vol. 54, no. 8, pp. 2834–2839, 1985.

[16] W.-W. Kim and S. Menon, "A new dynamic one-equation subgrid-scale model for large eddy simulations," in *33rd Aerospace Sciences Meeting and Exhibit*, p. 356, 1995.

[17] F. Nicoud and F. Ducros, "Subgrid-scale stress modelling based on the square of the velocity gradient tensor," *Flow, turbulence and Combustion*, vol. 62, no. 3, pp. 183–200, 1999.

[18] M. M. Rogers and R. D. Moser, "Direct simulation of a self-similar turbulent mixing layer," *Physics of Fluids*, vol. 6, no. 2, pp. 903–923, 1994.

[19] G. L. Brown and A. Roshko, “On density effects and large structure in turbulent mixing layers,” *Journal of Fluid Mechanics*, vol. 64, no. 4, pp. 775–816, 1974.

[20] F. Champagne, Y. Pao, and I. J. Wygnanski, “On the two-dimensional mixing region,” *Journal of Fluid Mechanics*, vol. 74, no. 2, pp. 209–250, 1976.

[21] G. I. Taylor and A. E. Green, “Mechanism of the production of small eddies from large ones,” *Proceedings of the Royal Society of London. Series A-Mathematical and Physical Sciences*, vol. 158, no. 895, pp. 499–521, 1937.

[22] N. Sandham and W. Reynolds, “Three-dimensional simulations of large eddies in the compressible mixing layer,” *Journal of Fluid Mechanics*, vol. 224, pp. 133–158, 1991.

[23] Y. Wang, M. Tanahashi, and T. Miyauchi, “Coherent fine scale eddies in turbulence transition of spatially-developing mixing layer,” *International Journal of Heat and Fluid Flow*, vol. 28, no. 6, pp. 1280–1290, 2007.

[24] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang, *Spectral methods: evolution to complex geometries and applications to fluid dynamics.* Springer Science & Business Media, 2007.

[25] F. Moukalled, L. Mangani, M. Darwish, *et al.*, “The finite volume method in computational fluid dynamics,” *An Advanced Introduction with OpenFOAM and Matlab*, pp. 3–8, 2016.

[26] Q. Zhou, F. He, and M. Shen, “Direct numerical simulation of a spatially developing compressible plane mixing layer: flow structures and mean flow properties,” *Journal of Fluid Mechanics*, vol. 711, pp. 437–468, 2012.

Appendices

Appendix A

PROJECTION METHOD SOLVER RK4

```
 /*---------------------------------------------*\
  ========= |
  \\ / F ield | OpenFOAM: The Open Source CFD Toolbox
   \\ / O peration |
    \\ / A nd | Copyright (C) 2011-2016 OpenFOAM Foundation
     \\/ M anipulation |
-------------------------------------------------
 while (runTime.run())
    {

       #include "CourantNo.H"




       Info<< "Time = " << runTime.timeName() << nl << endl;



       Uold = U; Urk= U;

//------------------------------------------------------
phi = (fvc::interpolate(U) & mesh.Sf());
dU = dt*(fvc::laplacian(turbulence->nuEff(), U) - fvc::div(phi,U));

Urk = Urk +dU/6;
U = Uold + 0.5*dU;
U.correctBoundaryConditions();
solve(pEqn == fvc::div(U)/runTime.deltaT());
U = U - (fvc::grad(p) * runTime.deltaT());
U.correctBoundaryConditions();

//------------------------------------------------------
```

```
phi = (fvc::interpolate(U) & mesh.Sf());
dU = dt*(fvc::laplacian(turbulence->nuEff(), U) - fvc::div(phi,U));

Urk= Urk +dU/3;
U = Uold + 0.5*dU;
U.correctBoundaryConditions();
solve(pEqn == fvc::div(U)/runTime.deltaT());
U = U - (fvc::grad(p) * runTime.deltaT());
U.correctBoundaryConditions();

//-----------------------------------------------------
phi = (fvc::interpolate(U) & mesh.Sf());
dU = dt*(fvc::laplacian(turbulence->nuEff(), U) - fvc::div(phi,U));



Urk = Urk+dU/3;
U = Uold + dU;
U.correctBoundaryConditions();
solve(pEqn == fvc::div(U)/runTime.deltaT());
U = U - (fvc::grad(p) * runTime.deltaT());
U.correctBoundaryConditions();

//-----------------------------------------------------
phi = (fvc::interpolate(U) & mesh.Sf());
dU = dt*(fvc::laplacian(turbulence->nuEff(), U) - fvc::div(phi,U));



Urk = Urk +dU/6;
U = Urk;
U.correctBoundaryConditions();
solve(pEqn == fvc::div(U)/runTime.deltaT());
U = U - (fvc::grad(p) * runTime.deltaT());
U.correctBoundaryConditions();


phi = (fvc::interpolate(U) & mesh.Sf());
turbulence->correct();




       Info<< "ExecutionTime = " << runTime.elapsedCpuTime() << " s"
           << " ClockTime = " << runTime.elapsedClockTime() << " s"
           << nl << endl;

    runTime.write();


    }

    Info<< "End" << endl;

    return 0;
}


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

Appendix B

MODIFYING PISOFOAM AND THE PROJECTION METHOD SOLVER FOR THE TAYLOR GREEN VORTEX

```


forAll(U,cellI)
  {
     U[cellI].x() = Foam::sin(U.mesh().C()[cellI].x())*
     Foam::cos(U.mesh().C()[cellI].y());

     U[cellI].y() = -Foam::cos(U.mesh().C()[cellI].x())
     *Foam::sin(U.mesh().C()[cellI].y());

     p[cellI] = 0.25*(Foam::cos(2*U.mesh().C()[cellI].x())
     +Foam::cos(2*U.mesh().C()[cellI].y()));

     initialEnergy = initialEnergy + 0.5*sqr(U[cellI].x()) + 0.5*
         sqr(U[cellI].y());

  }
```

Appendix C

RISK ASSESMENT

NTNU	Risk assessment	Prepared by	Number	Date
		HSE section	HMSRV2603E	04.02.2011
		Approved by		Replaces
HSE/KS		The Rector		01.12.2006

Unit: *EPT* **Date: 04.02.2019**

Line manager: Terese Løvås

Participants in the identification process (including their function)**:**

Short description of the main activity/main process: Master project for student Sondre Relling: Development of a low-dissipation solver for large-eddy simulation based on OpenFOAM

Activity from the identification process form	**Potential undesirable incident/strain**	**Likelihood:**	**Consequence:**			**Risk Value (human)**	**Comments/status Suggested measures**
		Likelihood (1-5)	Human (A-E)	Environment (A-E)	Economy/ material (A-E)		
Writing, programming, simulating on a computer	None	1	A	A	A	A	

Likelihood, e.g.:

1. *Minimal*
2. *Low*
3. *Medium*
4. *High*
5. *Very high*

Consequence, e.g.:

- *A. Safe*
- *B. Relatively safe*
- *C. Dangerous*
- *D. Critical*
- *E. Very critical*

Risk value (each one to be estimated separately):
Human = Likelihood x Human Consequence
Environmental = Likelihood x Environmental consequence
Financial/material = Likelihood x Consequence for Economy/materiel

NTNU	Risk assessment	Prepared by	Number	Date
		HSE section	HMSRV2603E	04.02.2011
		Approved by		Replaces
HSE/KS		The Rector		01.12.2006

Potential undesirable incident/strain

Identify possible incidents and conditions that may lead to situations that pose a hazard to people, the environment and any materiel/equipment involved.

Criteria for the assessment of likelihood and consequence in relation to fieldwork

Each activity is assessed according to a worst-case scenario. Likelihood and consequence are to be assessed separately for each potential undesirable incident. Before starting on the quantification, the participants should agree what they understand by the assessment criteria:

Likelihood

Minimal **1**	**Low** **2**	**Medium** **3**	**High** **4**	**Very high** **5**
Once every 50 years or less	Once every 10 years or less	Once a year or less	Once a month or less	Once a week

Consequence

Grading	**Human**	**Environment**	**Financial/material**
E **Very critical**	May produce fatality/ies	Very prolonged, non-reversible damage	Shutdown of work >1 year.
D **Critical**	Permanent injury, may produce serious serious health damage/sickness	Prolonged damage. Long recovery time.	Shutdown of work 0.5-1 year.
C **Dangerous**	Serious personal injury	Minor damage. Long recovery time	Shutdown of work < 1 month
B **Relatively safe**	Injury that requires medical treatment	Minor damage. Short recovery time	Shutdown of work < 1week
A **Safe**	Injury that requires first aid	Insignificant damage. Short recovery time	Shutdown of work < 1day

The unit makes its own decision as to whether opting to fill in or not consequences for economy/materiel, for example if the unit is going to use particularly valuable equipment. It is up to the individual unit to choose the assessment criteria for this column.

Risk = Likelihood x Consequence

Please calculate the risk value for “Human”, “Environment” and, if chosen, “Economy/materiel”, separately.

About the column ”Comments/status, suggested preventative and corrective measures”:

Measures can impact on both likelihood and consequences. Prioritise measures that can prevent the incident from occurring; in other words, likelihood-reducing measures are to be prioritised above greater emergency preparedness, i.e. consequence-reducing measures.

MATRIX FOR RISK ASSESSMENTS at NTNU

CONSEQUENCE						
	Extremely serious	🟨 E1	🟥 E2	🟥 E3	🟥 E4	🟥 E5
	Serious	🟨 D1	🟨 D2	🟥 D3	🟥 D4	🟥 D5
	Moderate	🟨 C1	🟨 C2	🟨 C3	🟥 C4	🟥 C5
	Minor	🟩 B1	🟩 B2	🟨 B3	🟨 B4	🟥 B5
	Not significant	🟩 A1	🟩 A2	🟩 A3	🟨 A4	🟨 A5
		Very low	Low	Medium	High	Very high
		LIKELIHOOD				

Principle for acceptance criteria. Explanation of the colours used in the risk matrix.

Colour		Description
Red	🟥	Unacceptable risk. Measures must be taken to reduce the risk.
Yellow	🟨	Assessment range. Measures must be considered.
Green	🟩	Acceptable risk Measures can be considered based on other considerations.