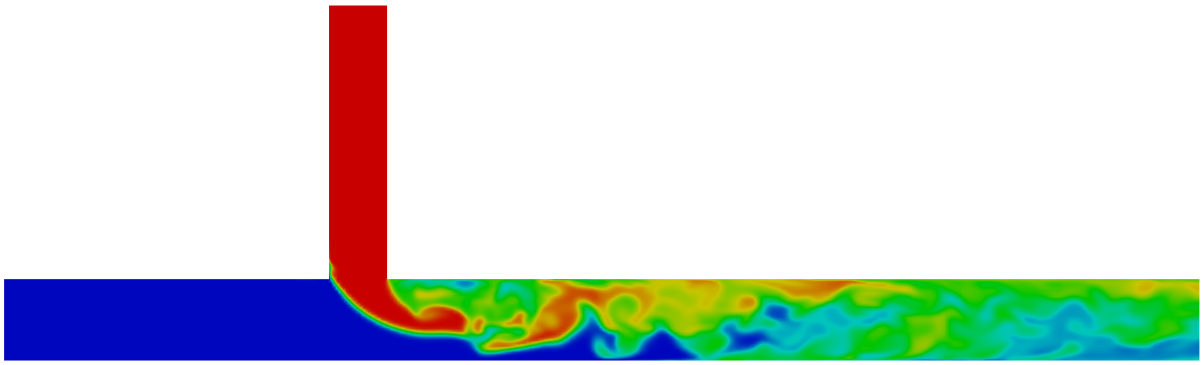
CHALMERS
UNIVERSITY OF TECHNOLOGY



Development of a Test Suite for Verification & Validation of OpenFOAM

Master's thesis in Nuclear Science and Technology

SIMON PERSSON

Department of Applied Mechanics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's Thesis 2017:59

Development of a Test Suite for Verification & Validation of OpenFOAM

Simon Persson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Applied Mechanics
Division of Fluid Dynamics
Chalmers University of Technology
Gothenburg, Sweden 2017

Development of a test suite for Verification & Validation of OpenFOAM
Simon Persson

Supervisor: Nicolas Edh, Forsmarks Kraftgrupp AB
Examiner: Håkan Nilsson, Department of Applied Mechanics

Master's Thesis 2017:59
ISSN 1652-8557
Department of Applied Mechanics
Division of Fluid Dynamics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 (0)31 772 1000

Cover: Temperature distribution for thermal mixing within a t-junction.

Department of Applied Mechanics
Typeset in LaTeX
Gothenburg, Sweden 2017

Development of a Test Suite for Verification & Validation of OpenFOAM
SIMON PERSSON
Department of Applied Mechanics
Division of Fluid Dynamics Chalmers University of Technology

Abstract

Computational fluid dynamics is an important tool which can be used to simulate various properties of a flow. CFD simulations are used within the nuclear power industry to aid in the evaluation of thermal loads within a given system. In order to verify and validate some aspects of the open source solver OpenFOAM a test suite was developed. The main goal was to build a series of automated tests while at the same time enable future additions of new tests. The verification process focused on the evaluation of certain numerical schemes used for both incompressible and compressible solvers in OpenFOAM. In order to evaluate the results of each case two different test criteria where used. The difference between simulated results and exact solutions to a specific problem was used to calculate a discretization error for the whole computational domain. The discretization error can then be used together with a refinement of the mesh to calculate an observed order of accuracy. In order to obtain exact analytic equations to the governing equations the method of manufactured solution was implemented. A solution for the equations was manufactured and a resulting source term was implemented in the simulation. The results from the simulation should then correspond to the manufactured solution. It was found that the results for the order of accuracy could be used as a test criteria when evaluating numerical schemes and solvers. A validation test case was also implemented into the test suite. The simulation was based upon an experiment previously used as a benchmark for various CFD simulations. It was found that a coarse mesh with a limited amount of cells could be used for evaluation of thermal mixing. In order to determine if a test is accurate enough a validation metric evaluated from the difference between the results from simulation and the experimental data. The metric was defined from the discretization error also used in the verification cases. The error was evaluated at the points where experimental data was available.

Keywords: Verification, Validation, OpenFOAM, CFD, MMS, Manufactured Solution.

Nomenclature

δ_{ij}	Kroenecker delta
ϵ	Error
$\hat{\phi}$	Discretized solution
$\hat{p}$	Observed order of accuracy
μ	Dynamic viscosity
$\overline{\phi}$	Time averaged ϕ
ϕ'	Fluctuating ϕ
ϕ	General variable
ρ	Density
τ_{ij}	Stress tensor
ε	Viscous dissipation
c_p	Specific heat capacity
e	Internal energy
g	Gravitational acceleration
h	Enthalpy
K	Kinetic energy
k	Turbulent kinetic energy
Pr	Prandtl number
q	Heat flux
R_s	Specific gas constant
S	Volumetric source
t	Time
TE	Truncation error
U	Magnitude of velocity
u	Velocity

Contents

Preface

This study intended to develop a set of test cases which could be used for verification and validation of certain numerical schemes and solvers used in OpenFOAM. The work has been performed between January 2017 to June 2017 at Forsmark Kraftgrupp AB, Sweden. I would like to thank Forsmark Kraftgrupp AB for the opportunity to work on this project as a part of my master's thesis. The experimental data for the validation test cases was provided by Vattenfall Research & Development, Älvkarleby. Without this data the validation test case could not have been included. Regarding future uses of the data an agreement must be signed between the user and Vattenfall Research & Development. Finally I would also like to give special thanks to my supervisor at Forsmark, Nicolas Edh, for his involvement and co-operation during the project.

Gothenburg, June 2017

SIMON PERSSON

Chapter 1

Introduction

This section will briefly cover the general topics presented in this thesis. The emphasis is put on computational fluid dynamics and its applications within some application of the nuclear power industry. The section will also cover some of the fundamental properties of a nuclear reactor and how those properties can affect simulation performed with computational fluid dynamics.

1.1 Computational Fluid Dynamics

Computational Fluid Dynamics (CFD) is a process where fluid flow and heat transfer are modeled from the transport equations that describe the motion of fluids. CFD is used in various industrial applications where aerodynamics, hydrodynamics and thermal transport of fluids are a part of the process [1].

Since the numerical models solved within CFD-calculations introduce approximations it is important to verify that the calculations are performed properly and that the simulations are validated. Verification is important in order to ensure that the software is operated according to its specifications. Correct implementation of the model and application of various mathematical tools within the software are processes which need to be verified. The validation procedure is focused on comparing the results from the simulation with data from real applications. It is not necessarily the process to prove that a model is correct in all aspects, rather if it is accurate enough to represent and provide information about the system that is being modeled. Experimental data or other already validated simulations are two examples of data that can be used to validate a simulation.

The CFD process generally involves a few steps in order to estimate the propagation of parameters within a fluid flow problem. The first step is to specify the domain and construct a geometry that represents the system and define the physical properties of the fluid flow. The domain is then divided into several small volumes which will make up a mesh of the domain. The solution of the problem is performed by numerical solution of the governing transport equations in each cell. This data can then be

used to describe the properties of the flow in the specified system. There are various types of techniques that can be applied but the most commonly used technique in CFD is based on the finite volume method [1]. This method is based on solving the transport equation in a defined finite control volume. The governing transport equation is integrated over the control volume, and the resulting equations are then discretised so that a system of algebraic equations is obtained. These equations can then be solved with numerical iterative methods from specified initial conditions.

1.2 Applications in Nuclear Power Industry

The main principle of a nuclear power plant is to convert thermal energy, released during fission of an atomic nucleus, to electrical energy which can be delivered to the electrical grid. The thermal energy is transferred from the reactor fuel to a coolant medium, water is most commonly used. The water is heated in a high pressure environment to increase the boiling temperature and to generate steam with high thermal energy. The steam is then transported to steam turbines that converts the thermal energy into electrical energy. Due to the high pressure and large temperature gradients combined with the long operating time and ionizing radiation the material properties of a component in the system may change. In order to ensure safe operation of a power plant the material transients must be analyzed with the help of physical models and computational simulations. Computation fluid dynamics (CFD) is one of the computational tools available to help evaluate the material transients within critical systems in the power plant.

Due to the presence of ionizing radiation that is produced in the fuel during operation the nuclear power industry has a very high demand on procedures and operational safety. The Swedish Radiation Safety Authority (SSM) has specified regulations regarding the use and application of simulations used to evaluate the operational safety of a power plant. Due to the approximations used to rewrite the equations and to perform the iterative calculations the performance of the simulations must be evaluated. By verifying and validating the results one can estimate if the simulations have been performed according to specifications of the software and that they are an accurate representation of the real system.

1.2.1 OpenFOAM

The computational tool to be evaluated for application within nuclear energy applications is called OpenFOAM. OpenFOAM (Open source Field Operation And Manipulation) is an open source CFD development tool used to solve problems within continuum mechanics [2]. OpenFOAM has a wide variety of application, however this thesis will only deal with the solvers and applications used for CFD simulations. More specifically, some of the solvers and utilities that can be used for evaluation of thermal transients within the nuclear industry is of interest.

1.3 Objective of this thesis

The main goal of the thesis is to formulate a number of test cases that can be used to evaluate simulations done with the open source CFD tool OpenFOAM. A number of different cases will be constructed and evaluated with an automatic test suite. The verification process should evaluate if the implementation of each simulation performs according to the specifications of the software. Emphasis will be put on implementation of each test case, analysis of how one can perform a verification of the mathematical model used in a simulation and create an automatic test loop that can be used for present and future releases of OpenFOAM.

OpenFOAM will also be validated against experimental data from experiments previously used as benchmark for various CFD simulations. The emphasis will be on temperature transients and flow structured which may lead to thermal fatigue in various components. The test will be based on experiments performed by Vattenfall Research and Development AB. Much of the work will focus on how to properly evaluate the simulations with the experimental data and what parameters are needed to set a criteria for the test. The work will be limited to a number of cases that are prepared and simulated with tools available in OpenFOAM. The main focus is to evaluate some of the solvers and numerical schemes used for simulations performed at Forsmarks Kraftgrupp AB.

Chapter 2

Theory

This chapter will present a theoretical background of the equations that are solved during a CFD simulation. The governing equations within OpenFOAM are based on the conservation equations for mass, momentum and enthalpy. A method used to verify the mathematical model of the simulations will also be stated. Finally some of the metrics used for the verification and validation will be explained.

2.1 Governing equations in CFD

The CFD simulations performed in this thesis are based upon solving the governing equations for fluid flow. These equations are often defined as the conservation of mass, momentum and energy. In order to close the system of equations additional relationships are needed, an equation of state is used to relate pressure and density while a thermophysical model is used to link the internal energy, or in this work enthalpy, to temperature.

2.1.1 Conservation of Mass

The equation that defines the conservation of mass (also called the continuity equation) can be derived from the rate of change for mass inside the system and the mass flow through each surface element of a control volume [1]. The resulting equation can be written with Einstein notation as [1]

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i) = 0, \tag{2.1}$$

where ρ is the density of the fluid and $\vec{u}$ is the velocity of the flow. Note that this equation describes the conservation of mass for a compressible fluid. If the fluid is

incompressible the density will be constant and the expression is simplified into

$$\frac{\partial}{\partial x_i} u_i = 0. \tag{2.2}$$

2.1.2 Conservation of Momentum

The conservation of momentum for a control volume is defined as [3]

$$\frac{\partial}{\partial t} u_i + \frac{\partial}{\partial x_j}(\rho u_j u_i) = -\frac{\partial}{\partial x_i}(\delta_{ij} p) + \frac{\partial}{\partial x_j}\tau_{ij} - \rho b_i, \tag{2.3}$$

where τ_{ij} is the stress tensor and b_i represents the body forces applied to the control volume. The stress tensor for Newtonian fluids can be written as [3]

$$\tau_{ij} = 2\mu D_{ij} - \frac{2}{3}\mu\delta_{ij}\frac{\partial}{\partial x_i} u_i, \tag{2.4}$$

where D_{ij} is the deformation vector defined as

$$D_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right). \tag{2.5}$$

The constant μ represents the dynamic viscosity.

2.1.3 Conservation of Energy

The conservation of energy implies that the rate of change of energy inside the system must be equal to the sum of heat added to the system and total work exerted by the fluid. The contribution to the energy equation will come from the internal (thermal) energy, kinetic energy and external body forces. The total energy inside the system can be written as $e_{tot} = e + K$ where e is the internal energy and K is the kinetic energy defined as $K = \frac{1}{2}u_i u_i$. From the general transport equation [4] one can derive the governing equation for internal energy

$$\frac{\partial}{\partial t}(\rho e) + \frac{\partial}{\partial x_j}(\rho e u_j) = \frac{\partial}{\partial x_i} q_i + S_q \tag{2.6}$$

where q is the heat flux and S_q is a heat source.

The kinetic energy equation can also be derived from the general transport equation

$$\frac{\partial}{\partial t}(\rho K) + \frac{\partial}{\partial x_j}(\rho u_j K) = \frac{\partial}{\partial x_j}(u_j p) + \frac{\partial}{\partial x_j}(u_i \tau_{ij}) - \rho u_i b_i \tag{2.7}$$

and if we add the internal energy with kinetic energy the conservation equation for total energy is obtained

$$\frac{\partial}{\partial t}(\rho e) + \frac{\partial}{\partial x_j}(\rho e u_j) + \frac{\partial}{\partial t}(\rho K) + \frac{\partial}{\partial x_j}(\rho u_j K) = \frac{\partial}{\partial x_j}(u_j p) + \frac{\partial}{\partial x_i} q_i + \frac{\partial}{\partial x_j}(u_i \tau_{ij}) + S_q - \rho u_i b_i. \tag{2.8}$$

It can often be beneficial to write the conservation of energy in the form of enthalpy. The enthalpy of the system can be written as the sum of total energy in the system and kinematic pressure

$$h = e_{tot} + \frac{p}{\rho}. \tag{2.9}$$

The terms containing internal energy in equation (2.6) can then be rewritten as

$$\begin{aligned} \frac{\partial}{\partial t}(\rho e) &= \frac{\partial}{\partial t}\left[\rho\left(h - \frac{p}{\rho}\right)\right] = \frac{\partial}{\partial t}(\rho h) - \frac{\partial}{\partial t}p, \\ \frac{\partial}{\partial x_j}(\rho e u_j) &= \frac{\partial}{\partial x_j}\left[u_j\rho\left(h - \frac{p}{\rho}\right)\right] = \frac{\partial}{\partial x_j}(\rho h u_j) - \frac{\partial}{\partial x_j}(u_j p), \end{aligned} \tag{2.10}$$

leading to the governing equation for enthalpy

$$\frac{\partial}{\partial t}(\rho h) + \frac{\partial}{\partial x_j}(\rho h u_j) + \frac{\partial}{\partial t}(\rho K) + \frac{\partial}{\partial x_j}(\rho u_j K) - \frac{\partial}{\partial t}p = \frac{\partial}{\partial x_i}q_i + \frac{\partial}{\partial x_j}(u_i\tau_{ij}) + S_q - \rho u_i b_i. \tag{2.11}$$

2.1.4 Incompressible Navier-Stokes Equations

For many applications of fluid flow the incompressible Navier-Stokes equations can be used. An incompressible flow can be defined as a flow where the divergence of the velocity is zero [1]. The Navier-Stokes equations for incompressible fluid can be derived from the conservation of mass (2.1) and momentum (2.3)

$$\begin{aligned} &\frac{\partial}{\partial x_i}u = 0 \\ &\frac{\partial}{\partial t}u_i + u_j\frac{\partial}{\partial x_j}u_i = -\frac{\partial}{\partial x_i}(\delta_{ij}p) + \frac{\partial}{\partial x_j \partial x_j}u_i - \rho g_i, \end{aligned} \tag{2.12}$$

where the stress tensor τ_{ij} has been rewritten since the viscosity is constant and the effect from the bulk viscosity is zero [1]. The term associated with body forces has been approximated to g_i which represents the external acceleration applied on the fluid due to gravitational forces.

2.1.5 Compressible Navier-Stokes

The Navier-Stokes equations for compressible flow that are implemented in OpenFOAM can be defined from the governing equations for mass, momentum and enthalpy which are written above (2.1), (2.3), (2.11). If we neglect body sources except

the gravitational acceleration the Navier Stokes equations will be

$$\begin{aligned}
&\frac{\partial}{\partial t}\rho + \frac{\partial}{\partial x_i}(\rho u_i) = 0 \\
&\frac{\partial}{\partial t}u_i + \frac{\partial}{\partial x_j}(\rho u_j u_i) = -\frac{\partial}{\partial x_i}(\delta_{ij} p) + \frac{\partial}{\partial x_j}\tau_{ij} - \rho g_i \\
&\frac{\partial}{\partial t}(\rho h) + \frac{\partial}{\partial x_j}(\rho h u_j) + \frac{\partial}{\partial t}(\rho K) + \frac{\partial}{\partial x_j}(\rho u_j K) - \frac{\partial}{\partial t}p = \frac{\partial}{\partial x_i}q_i + \frac{\partial}{\partial x_j}(u_i \tau_{ij}) + S_q - \rho u_i g_i
\end{aligned} \tag{2.13}$$

where the stress tensor is defined as

$$\begin{aligned}
\tau_{ij} &= 2\mu D_{ij} - \frac{2}{3}\mu\delta_{ij}\frac{\partial}{\partial x_i}u_i \\
D_{ij} &= \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right).
\end{aligned} \tag{2.14}$$

2.1.6 Turbulence modelling

Turbulence can be defined as chaotic and unpredictable changes in the flow pattern of moving fluids [5]. In order to try and describe a turbulent flow the governing variables are often divided into one time- (RANS) or spatial (LES) averaged part $\overline{\phi}$ and one fluctuating part ϕ' according to

$$\phi = \overline{\phi} + \phi'. \tag{2.15}$$

If we insert this decomposition of the velocity into the NS equations for incompressible flow (2.12) and perform an average w.r.t space or time for each term we obtain [1]

$$\begin{aligned}
&\frac{\partial}{\partial x_i}\overline{u} = 0 \\
&\frac{\partial}{\partial t}\overline{u}_i + \overline{u}_j\frac{\partial}{\partial x_j}u_i = -\frac{1}{\rho}\frac{\partial}{\partial x_i}(\delta_{ij}p) + \frac{\partial}{\partial x_j \partial x_j}\overline{u}_i + \frac{1}{\rho}\frac{\partial}{\partial x_j}(\rho u_i' u_j') - \rho g_i,
\end{aligned} \tag{2.16}$$

where a new term $u_i' u_j'$ representing the Reynolds stresses is present. This term will represent the correlation between the fluctuating velocities within the turbulent flow [5]. One way to resolve this term is by applying a model which will try to predict the fluctuating velocities within the flow. There are a wide variety of different models that can be applied [5]. The models which will be used for this work are based on two different methods, Reynolds Averaged Navier-Stokes (RANS) and Large-Eddy Simulations (LES).

The RANS models are based on a time average of the fluctuating velocity terms (2.16). The model used within this work is the standard $k - \epsilon$ model. This model uses the turbulent kinetic energy k and the rate of viscous dissipation ϵ to resolve the Reynolds stresses present during turbulent flow [1]. One disadvantage of the

RANS models comes from problems that occur when it tries to model all turbulent eddies of the flow with the same model. The characteristics of large eddies are very different from the very small eddies. Small eddies most often behave in an isotropic manner while the larger eddies will behave very anisotropicially due the interaction with other properties of the flow. This is one of the reasons one might use the more computational heavy LES-model. Large-eddy simulations puts an emphasis on the large eddies and their interaction with the surrounding fluid while the small eddies are modeled by a more simple sub-grid-scale model. In order to separate the interaction of small and large eddies a spatial filter function is applied to the unsteady NS equations (2.16). The filter function used within this thesis is based on the cubic root of the cell volume [5].

2.1.7 Thermophysical models

There is a wide variety of thermophysical models available for simulations within OpenFOAM. The definition of the thermo physical model is a combination of transport-, thermodynamic- and equation of state models which are used to describe the physical properties of the fluid and close the system of governing equations.

The transport model is used to determine fluid properties such as dynamic viscosity μ, thermal conductivity κ and thermal diffusivity α. For this work the quantities have been defined by two different models. The most simple model requires constant values for the dynamic viscosity and Prandtl number [6] which can be used to calculate all relevant properties since the Pr number is defined as

$$Pr = \frac{c_p \mu}{\kappa}. \tag{2.17}$$

For a simulation where large temperature fluctuations are present the properties of the flow might be temperature dependent. A model where the dynamic viscosity and thermal conductivity is defined from a polynomial function of temperature can then be used

$$\mu = \sum_{i=0}^{N-1} a_i T^i, \tag{2.18}$$

where the polynomial is of order N.

The thermodynamic model is used to determine the specific heat capacity c_p (c_v for internal energy) which is later used to calculate the temperature from the enthalpy within the system. Just as for the transport model the value of c_p can be provided as a constant or as a polynomial function of order N

$$c_p = \sum_{i=0}^{N-1} a_i T^i. \tag{2.19}$$

The equation of state is used to determine the density of the fluid. Several models are available [6] but only the models based on the ideal gas law and temperature

dependent polynomial are used for this thesis. The *perfectGas* model will relate density to the pressure and temperature according to

$$\rho = \frac{p}{R_s T}, \tag{2.20}$$

where R_s is the specific gas constant. For an incompressible fluid the density can be defined from a temperature dependent polynomial just as the specific heat, see Equation (2.19).

2.2 Verification

The term software verification can be defined in many different ways, generally dependent on what type of software that is to be verified. When verification is used within scientific computing i.e. modelling or computer simulation the focus on verification is to evaluate the correctness and accuracy of a mathematical model [7]. An example of a very general term of the verification procedure for scientific software can be stated as: "Solve the equations right." [8]. The goal is often to ensure accurate numerical solutions to the various mathematical models used within the scientific software. Therefore the focus of the verification procedure should be to identify, quantify and reduce the error in the computer code and the numerical simulation [7]. The verification of scientific computer software can further be divided into two different categories, code verification and solution (or calculation) verification [7].

Solution verification is defined as the process to determine the correctness of input- and output data and numerical accuracy of the end result [7]. This numerical accuracy will include iterative and numerical errors that arises when solving system of equations on a digital computer.

The code verification evaluates if the code is a good representation of the mathematical model that is used to solve the governing equations. This model includes the discretisation of governing equations and the implementation of various initial and boundary conditions that is used to solve the equations. In order to evaluate a code the solution must be compared to some criteria in order to determine the accuracy of the mathematical model. These various criteria can be divided into a few sub-categories with various degrees of stringency [7]. From the most simple to most rigorous these tests are; simple-test, code-to-code comparison, discretisation error and order of accuracy-tests.

Simple tests includes scenarios where an exact solution is not necessary. These include symmetry tests where a symmetrical problem is modeled from symmetrical boundary and initial conditions. The final result should be a symmetrical solution for the whole geometry. This means that the model can be evaluated from just a simple geometry and boundary conditions. Another simple test for verification is to check the conservation of various quantities in the mathematical model. If we take a look at the conservation equation (2.1) one can compare the total flow of

mass into a domain with the total flow going out of it. Another test which does not require an exact solution to the problem is code-to-code verification. By taking an already verified solution to a specific problem and compare it to the model that is to be evaluated. This test assumes that the mathematical models are the same for both cases and that the code that is used as a benchmark has been verified in some manner. The problem with this type of verification is that the exact same models and algorithms must be used in order to produce the same results [7].

The tests that evaluates discretization error and order of accuracy will require an exact solution to be available in order to perform the verification. This can be achieved by having an analytic solution to a particular problem that is used for verification of the model. If an analytic solution is available it is possible to compare the results from the simulation with the exact solution to the problem. However it may be very difficult to find an exact solution for some types of problems. Examples of this can be found for CFD simulations where complex geometries, multiple non-linear coupled and sub-models are present.

2.2.1 Method of Manufactured Solutions

As mentioned earlier many of the tests used for verification requires an exact solution to the mathematical model that is to be solved and for problems where complex partial differential equations are included an exact solution can be very difficult to obtain. The method of manufactured solutions (MMS) can be used to construct an exact analytic solution from the governing equations of the mathematical model. As the name suggests a solution is manufactured and the equations are solved backwards from this solution by adding a source term to the equations. Since the goal of the verification procedure is to verify the mathematical correctness of a model an exact solution with realistic physical properties is not required [7]. However one must take care so that conditions used for the model to be valid must be met. One such example could be the requirement of a divergence free velocity field ($\nabla \cdot \vec{u} = 0$) for an incompressible model. Another potential problem may arise within an application where the transfer of heat can be evaluated. If a manufactured solution for temperature is defined it is important that the value is positive within the whole domain if the unit Kelvin is used.

The procedure to create a manufactured solution starts with the governing equations for the mathematical model. For partial differential equations the derivatives can be represented by a linear operator $L()$. Say that we want to create a solution for the variable ϕ, the governing equation can now be represented as

$$L(\phi) = 0. \tag{2.21}$$

A manufactured solution ϕ_m is chosen, however this solution will not solve the system of equations since $L(\phi_m) \neq 0$. We can now implement a source term S_ϕ which is defined as the result of the manufactured solution $L(\phi_m) = S_\phi$. By adding this source to the governing equations used in the mathematical model the equation

to be solved will be

$$L(\phi) = S_\phi, \tag{2.22}$$

which should return the simulated results of the actual manufactured solution ϕ_m. The solution obtained from the simulation ϕ can now be compared to the exact solution that was constructed ϕ_m.

Another feature of the manufactured solutions is that a certain criteria for the boundary conditions can be applied to the solution. The boundary can be described by a spatial function $F(\vec{r}) = C$ where C is the constant value of a physical quantity at he boundary. By multiplying the spatially dependent terms of the manufactured solution with a the difference $(C - F(\vec{r}))^m$ one can set the boundary condition to fulfill a desired criteria [9].

2.2.2 Truncation error

The truncation error is defined as the difference between an exact integral or differential equation and the implemented discretised equation used in a computer simulation. Discretization of an equation is the process of transforming a continuous mathematical expression into a discrete representation for spatial and transient variables which can be solved numerically [3]. The discretization is not an exact representation of the continuous expression which means an error has been introduced into the solution. The truncation error can be evaluated by applying a Taylor series expansion on each variable and insert them into the discretized equations. As an example we can produce a simple discretization of the derivative of ϕ. From the expansion of $\phi(x)$ we can rearrange the terms into an expression for the truncation error

$$\begin{aligned}
\phi(x) &= \phi(x_0) + (x - x_0)\left.\frac{\partial \phi}{\partial x}\right|_{x_0} + \frac{(x - x_0)^2}{2}\left.\frac{\partial^2 \phi}{\partial x^2}\right|_{x_0} + \mathcal{O}[(x - x_0)^3] \\
\phi(x) - \phi(x_0) &= (x - x_0)\left.\frac{\partial \phi}{\partial x}\right|_{x_0} + \frac{(x - x_0)^2}{2}\left.\frac{\partial^2 \phi}{\partial x^2}\right|_{x_0} + \mathcal{O}[(x - x_0)^3] \\
\frac{\phi(x) - \phi(x_0)}{x - x_0} &= \left.\frac{\partial \phi}{\partial x}\right|_{x_0} + \frac{(x - x_0)}{2}\left.\frac{\partial^2 \phi}{\partial x^2}\right|_{x_0} + \mathcal{O}[(x - x_0)^2],
\end{aligned} \tag{2.23}$$

where the left hand side represents the discretised solution and the second- and third term on the right hand side is the truncation error. By defining $L()$ as a vector containing each operation made on the mathematical model, we can write a general expression for truncation as

$$L_r(\phi) = L(\phi) + TE_r(\phi) \tag{2.24}$$

where TE is the truncation error and r is the spatial and temporal refinement parameter of the system.

2.2.3 Discretization Error

The discretisation error can be derived from the numerical solution of a system if the exact solution to the problem is known. It is a scalar that can be evaluated locally or for the entire domain [3]. If an exact solution ϕ is known and the result from a numerical simulation $\hat{\phi}$ has been obtained the continuous disretization error is defined as

$$\|\hat{\phi} - \phi\|_1 = \frac{1}{\Omega}\int_\Omega |\hat{\phi} - \phi| d\omega, \tag{2.25}$$

where the L1-norm of the error has been evaluated. For finite-volume simulations there is no continuous solution to the problem which means a discrete error must be evaluated

$$\|\hat{\phi} - \phi\|_1 = \frac{1}{\Omega}\sum_{n=1}^{N} a_n |\hat{\phi}_n - \phi_n|, \tag{2.26}$$

where n represents a cell in the domain, N is the total number of cells and a_n in the size of each cell. For a uniform mesh the size of each cell will be identical which means the expression can be simplified into

$$\|\hat{\phi} - \phi\|_1 = \frac{1}{N}\sum_{n=1}^{N} |\hat{\phi}_n - \phi_n|. \tag{2.27}$$

Another representation of the error is the root mean square (RMS) error which is simply obtained from the L2 norm of the error. For a discrete solution with a uniform grid this can be written as

$$\|\hat{\phi} - \phi\|_2 = \left(\frac{1}{N}\sum_{n=1}^{N} |\hat{\phi}_n - \phi_n|^2\right)^{1/2}. \tag{2.28}$$

Finally one can define the absolute maximum value of the error by taking the infinity norm

$$\|\hat{\phi} - \phi\|_\infty = max(|\hat{\phi}_n - \phi_n|). \tag{2.29}$$

From equation (2.28) it is clear that discretisation error has a direct relation to the size of the mesh. A finer mesh with more cells will reduce the error of the simulated solution. However the size and quality of the mesh is often a limiting factor when using CFD since a larger mesh will contribute to a longer computational time. Therefore the error cannot be made insignificant without drastically increasing the amount of memory and computational time needed to solve a particular problem [1].

There are some disadvantages that must be considered when evaluating a mathematical model from the discetization error. The first problem is that an exact solution to the problem must be available. As mentioned earlier there are numerous problems where an exact solution to a problem can be very difficult to obtain. One example is the Navier-Stokes equations for compressible flow defined in equation (2.1), (2.3) and (2.11). Another disadvantage comes from the fact that the error must undergo an evaluation to determine if it is sufficiently small. There is no obvious answer to

how small the error is supposed to be for an accurate model. A subjective analysis based on the working parameters, numerical values and application of the model must be considered in order to evaluate if a given error is sufficiently small.

2.2.4 Order of accuracy

The order of accuracy criteria for code verification combines a convergence test of the numerical simulation and the reduction of the discretisation error as the mesh is refined. One can divide the test into a theoretical accuracy called formal order of accuracy and the observed order of accuracy obtained from tests on the mathematical model. For an accurate model the discretization error should decrease at the same rate as the formal order of accuracy.

The formal order of accuracy is the theoretical rate at which the simulated solution converges towards the exact solution of the model. As shown earlier there is a direct connection between the size of the numerical grid and the discretisation error. If the mesh discretization parameter r is reduced the discretization error of the solution should also decrease (2.28). By inserting the solution to the discrete equation, $\hat{\phi}$, in the expression for truncation error (2.24) we get the expression

$$L(\hat{\phi}) + TE_r(\hat{\phi}) = 0, \tag{2.30}$$

since the term $L_r(\hat{\phi}) = 0$ [7]. From here one can subtract the solution for the exact mathematical model $L(\phi)$ from the expression and rewrite it once again to obtain

$$L(\hat{\phi}) - L(\phi) + TE_r(\hat{\phi}) = 0. \tag{2.31}$$

If L is a linear operator this can be written as [7]

$$L(\hat{\phi} - \phi) = -TE_r(\hat{\phi}). \tag{2.32}$$

Note that the right hand side of the expression is the previously defined truncation error of the discretized solution. This means that we can use the truncation error of a numerical scheme to calculate the formal order of accuracy.

The observed order of accuracy is the actual order of accuracy for the mathematical model that is to be verified. The discretized equation ϕ_r can be expanded in terms of the mesh spacing (size of each cell) h to obtain [7]

$$\phi_h = \phi_{h=0} + h\frac{\partial \phi}{\partial h}\bigg|_{h=0} + \frac{h^2}{2}\frac{\partial^2 \phi}{\partial h^2}\bigg|_{h=0} + ... + \mathcal{O}(h^n), \tag{2.33}$$

if we let $h \to 0$. For a numerical scheme that is p:th order accurate the terms of order $p-1$ will be equal to zero, see definition of truncation error (2.24). The expression can therefore be written as

$$\phi_h = \phi_{h=0} + \frac{h^p}{p!}\frac{\partial^p \phi}{\partial h^p}\bigg|_{h=0} + \mathcal{O}(h^{p+1}), \tag{2.34}$$

and since we let the mesh parameter $h \to 0$ the term $\phi_{h=0}$ can be approximated to the exact solution to the problem ϕ. This means the equations can be rewritten once again to obtain

$$\epsilon_h = f_p h^p + \mathcal{O}(h^{p+1}), \tag{2.35}$$

where ϵ_h is the difference between the discretized- and exact solution for a scheme with order of accuracy p. The coefficient f_p is only dependent of spatial or temporal parameters $f_p(x, y, z, t)$ [3].

Since we assumed the parameter h is small the higher order terms than p can also be neglected. This implies that for two different mesh spacing h and rh the error can be written as

$$\begin{aligned} \epsilon_h &= f_p h^{\hat{p}} \\ \epsilon_{rh} &= f_p (rh)^{\hat{p}}, \end{aligned} \tag{2.36}$$

where $\hat{p}$ is the formal order of accuracy, r is the refinement factor and p is the formal order of accuracy of the numerical scheme. Combining these into a single equation yields and expression for the observed order of accuracy

$$\hat{p} = \frac{ln\left(\frac{||\epsilon_{rh}||}{||\epsilon_h||}\right)}{ln(r)}, \tag{2.37}$$

where the norm of the discretization errors has been used to obtain an observed order of accuracy for the whole domain. Note that this expression is only accurate if the lowest order terms for both discretization error and truncation error are dominant for both expressions [7]. This is called the asymptotic range and the order of accuracy will only be equal to the formal order of accuracy within this asymptotic region.

2.3 Validation

Validation is used to estimate the accuracy of a computational model when it is compared to a real physical application. This is different from the verification procedure where the focus was to evaluate the accuracy of the mathematical operations used in the model. The most common method for validation is to compare simulated results with data taken from an experiment[7]. This means that either an experiment must be designed for the validation of a certain numerical model or an existing experiment for some application is used as a design basis for the simulation.

When comparing the model that is to be validated against the experimental results there are several considerations to be made. First one must consider what application the model to be validated will be used for. For many research and scientific applications the validation is used as a tool to evaluate theoretical models with the help of experimental data. For engineering applications the goal of the validation is often more focused on the accuracy of the model and whether it can be used to gain information about a certain application [7].

Other considerations to be made during assessment is how the data from simulation and experiments can be compared. Transient simulations of turbulent flow will have very large fluctuations of the result which is dependent on time. One solution to this problem is to calculate a mean value for each quantity which is used to describe the properties of the system. If the mean value is taken over a long enough time interval the fluctuating parts of the solution can be estimated as zero [5].

2.3.1 Validation Metric

The accuracy of a model can be estimated by a validation metric which compares the results from simulation and experiments for various input parameters. A metric can be defined in a number of different ways. There are several considerations that can be taken when evaluating what metric to use. Some of these are the overall goal of the validation, availability of experimental data, uncertainties and number of experimental or simulated results [7]. For this work the goal was to use data from one experiment and compare with the results from a simulation. Due to the transient nature of the results and the chaotic flow pattern present within turbulent flow the results from both experiment and simulation must be comparable in some form. The validation for this thesis will use a metric that compare estimated mean values of available quantities from both experiment and simulation.

The estimated error of the simulation can be defined as the difference between the simulated mean value $\overline{\phi}$ and experimental measurement $\overline{\phi}_e$

$$\tilde{\epsilon} = \overline{\phi} - \overline{\phi}_e. \tag{2.38}$$

The mean values are taken for each point of measurement available from the experimental data. The result from the simulation will be interpolated to the measurement point used in the experiment from the closest available cell points. This means that the data from the experiment will not be modified for the validation of the simulation. From the estimated error between simulation and experiment a total error for all available data points can be estimated by taking the norm of the difference. Both the $L2$- and infinity norm (2.28) (2.29) will be calculated for each spatial region where measurements was taken during the experiment, see Figure 4.1.

Chapter 3

Verification of Numerical Schemes and Solvers in OpenFOAM

The beginning of this chapter will give a brief overview of the test suite which is used to execute each test. The focus of the test suite is to run a number of test cases and report the results to the user. Each case is prepared and ready to be executed individually, in groups according to type of test or as one test where all cases are executed. In order to keep the runtime within a reasonable limit the test cases are limited to a few number of variations with different schemes and mesh resolution.

The first set of tests have been designed to evaluate the discretization of some of the mathematical operators used by the solvers in OpenFOAM. These cases will evaluate how well the numerical schemes can discretize some of the differential operators used when performing CFD simulations.

Due to the difficulty of obtaining exact analytic solutions to the governing equations some verification cases have been evaluated with the MMS. A few solvers used for incompressible and compressible flow were evaluated by manufacturing a solution to the governing equations and calculate a source term which is implemented into the solver. Since the result of the simulation should correspond to the manufactured solutions the discretization error and order of accuracy can be evaluated.

3.1 Automatic test harness

The main feature of the software used for the verification and validation tests is to execute each simulation and report the results. Each test case will be initiated by a script which starts the pre process, simulation and post-process utilities needed for each test. Pre processing utilities will be generating the mesh and prepare initial and boundary conditions for each case. The necessary settings for the actual simulation is already prepared for each test case. All of the solvers used for the tests

will be standard solvers and the settings are chosen from recommendations within the OpenFOAM user guide [6] or available tutorial cases. After the simulation is performed the utilities available in OpenFOAM will be used for most of the post processing of the data.

In order for a test to be successful a test criteria must be defined which will check if the simulation has passed or failed the test. This criteria will check if the simulation was able to execute and finish all of the specified utilities for each case. The data obtained from the applications used from the post processing tools in OpenFOAM will be compared to an expected value, e.g. max discretization error and order of accuracy evaluation. In order to calculate the order of accuracy a very small script is used to compare the resulting discretization error from two different simulations. A special script was also made specifically for interpolation of some of the results for the validation case of a t-junction. In order to compare the simulation with the experimental data the difference in velocity must be evaluated at a point specified by the experimental data.

3.1.1 CMake & CTest

The software which is used to initiate the tests is called CTest. It is an open source tool mainly used for the execution of fast and simple unit tests which is included into the system software managing tool CMake. The actual execution of each test will be performed with simple scripts present within each of the OpenFOAM cases. Each test will consist of a pre processing step where boundary conditions are set and the mesh is prepared. The next step is to perform the numerical simulation and report whether the simulation was successful or failed for some reason. The final step for each case is to check the calculated values for discretization error or order of accuracy and compare them to the previous test run.

In order to present the results another tool from CMake is used, the web based server tool CDash. The results are uploaded to a server and the result from each test can be evaluated. The result from each test, pass or fail, is reported and the result is easy to identify. In order to compare the results between two test runs the important evaluation data is stored after each test has been completed. This data can then be used for the future simulation to evaluate if the results between two tests have changed since last time the test was performed.

3.1.2 OpenFOAM file structure

All numerical simulations in OpenFOAM will have a similar file structure. There are three main directories which are used when running an application; constant, system and time directories.

The constant directory contains files for all physical properties used in the simulation

and all information regarding the computational mesh. The physical properties to be defined will vary depending on the solver that is used to run a simulation, more information can be found in the user guide [6]. The computational mesh can be generated from various applications contained within OpenFOAM or from other external applications.

Files contained in the system directory are used to set parameters used by OpenFOAM while running the simulation. Apart from various files used for optional functions, mesh generation information and post processing utilities the system directory contains the required files *controlDict*, *fvSchemes* and *fvSolutions*. The *controlDict* file is used to set running parameters such as simulation time, time step and data output parameters. It is also used to specify what utilities are to be used for data output and post process. Selection of solvers used to solve the discretized equations and the criteria for tolerances are defined in *fvSolution*. Here the algorithm control for various pressure correction algorithms can also be defined. Finally the numerical discretization schemes used in the simulation will be set in the *fvSchemes* file.

The final type of directory in the OpenFOAM structure is the time directories. For a given time of the simulations the previously defined fields will be written in a corresponding time directory, the interval between each directory is defined in the *controlDict*. The initial conditions for the simulation is defined in a time directory corresponding to the start time of the simulation. Initial conditions are required before a simulation can be executed.

3.2 Test cases

There are a vast number of tests that can be set up for evaluation of a numerical simulation. The design of a test case will depend on what application the software is used for. The aim of this work was to evaluate and set up a test structure which can be further built upon by adding test cases or modify existing ones. A few different types of tests have been designed to test some of the solvers that are typically used for nuclear applications. The tests will focus on velocity and temperature distribution for an incompressible liquid solvers where special thermophysical models for high pressure environments can be used.

3.3 Test of Various Differential Operators

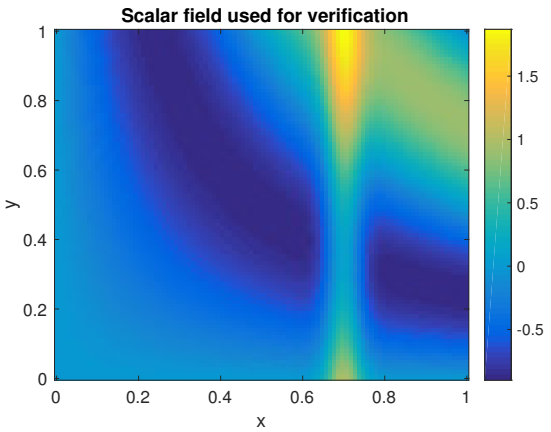
There are a vast variety of smaller tests that can be defined for any numerical calculation OpenFOAM performs while running a simulation. These tests can be very easy to set up by constructing a custom solver which specifically evaluates the desired operation of the code. In order to test some of the more common discretization schemes an application which defines a specific field on which a mathematical opera-

tion can be performed. The operators that are evaluated for this thesis are gradient, divergence and laplacian.

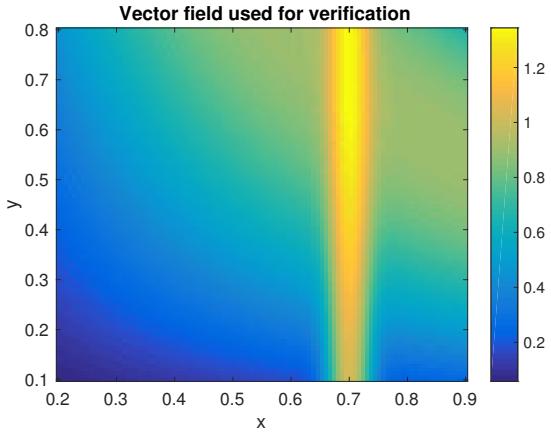
To achieve this a field has to be specified on which to apply each operator. Two important factors to consider are continuity and the derivatives of the field. Since the goal of the test is to evaluate differential operators it is important to define a field which is differentiable. A differential field will ensure that each differential operation will be defined for all points of the whole domain. By defining a dimensionless field which consists of various exponential and trigonometric terms these criteria will be fulfilled. A scalar field and a vector field is defined from two simple expressions

$$\begin{aligned} f(x,y) &= a_0 \exp\left(-a_1 \frac{(x-a_2)^2}{L}\right) + b_0 \sin\left(-b_1 \pi \frac{xy^2}{L}\right) \\ \vec{g}(x,y,z) &= \left(a_0 \exp\left[-a_1 \frac{(x-a_2)^2}{L}\right], b_0 \sin\left[-b_1 \pi \frac{xy^2}{L}\right], 0\right) \end{aligned} \tag{3.1}$$

where the values for constants a, b_x and L can be found in Table 3.1. The coefficients for the fields where chosen so that the gradients at some positions are fairly large, see Figures 3.1a 3.1b. This was performed since many of the discretization schemes perform some sort of linear interpolation to discretize the solution of a differential operation. For a very rough mesh this might cause substantial errors in the numerical calculation.



(a) Scalar field used to calculate the order of accuracy for numerical schemes in OpenFOAM.



(b) magnitude of the two dimensional vector field used for the verification of the discretization of divergence.

In order to implement this test an application was written where the field defined in equation (3.1) was defined over in a very simple geometry. The computational domain is two dimensional with equal sides and one cell size thick. After the field is defined for the whole domain the mathematical operation is executed for all cells in the domain. The result is compared to the analytic solution calculated directly from the equations defined in (3.1). The differential operations that are performed is the divergence of a vector field $\nabla \cdot \vec{g}$, gradient of the scalar field ∇f and laplacian of the scalar field $\nabla^2 f$.

In order to evaluate the order of accuracy of the numerical schemes used to discretize the solutions the operations are performed for a range of varying mesh sizes starting from a size of 20×20 up to 320×320. The discretization error will be evaluated from the $L2$-norm (2.28) and the order of accuracy is calculated with the previously defined equation (2.37).

Table 3.1: Coefficients used for verification of differential operators.

	$\mathbf{a_0}$	$\mathbf{a_1}$	$\mathbf{a_2}$	$\mathbf{b_0}$	$\mathbf{b_1}$	$\mathbf{L}$
Scalar field	1	15	0.7	0.9	1	1
Vector field	1	15	0.7	0.9	0.5	0.7

3.4 Evaluation with Method of Manufactured Solution

When designing a test based on the method of manufactured solution some properties of the applied solution must be taken into consideration. The function that describes the solution should be analytic and all of the derivatives should be smooth [7]. Care must also be taken when performing the derivatives present in the governing equations so that no terms in the equations disappear. By constructing a solution based on exponential and trigonometric functions all of these considerations will be fulfilled.

Another important aspect of the method is to make sure all of the physical quantities can handle restriction from the model that is to be verified. One such example that need to be considered for solutions designed for OpenFOAM is valid for temperature. The standard solvers in OpenFOAM uses the unit Kelvin (K) for temperature data, therefore it is important that the solution strictly larger than zero for all points within the domain.

The benefit of MMS means that any of the governing equations used in the mathematical model of the simulation can be evaluated. The first case to be evaluated is the incompressible Navier Stokes equations. There are a variety of solvers for incompressible flow available in OpenFOAM for both steady state and transient analysis of the flow. Two solvers are selected for evaluation with MMS; *simpleFoam* which is a steady state turbulent solver based on the PISO algorithm for pressure correction [1] and *pimpleFoam* which is a transient solver for turbulent flow based on the combined PISO-SIMPLE algorithm.

Another test is performed on a compressible solver where there are no demands on the solutions to be incompressible, see Section 2.1.4. Since the transfer of heat within the fluid for liquid flow in high pressure environments is of interest for the applications at Forsmark a transient solver for heat transfer is selected; *buoyantPimpleFoam* which is based on the *pimpleFoam* algorithm for buoyant flow [6].

3.4.1 MMS for Incompressible flow

The solvers for incompressible flows demand that the solution that is used to calculate the sources for the governing equations are incompressible. For an incompressible flow the divergence of the velocity field must be equal to zero

$$\nabla \cdot \vec{u} = 0, \tag{3.2}$$

see Equation (2.12). The computation will be performed for a simple two-dimensional domain since the goal of the verification is to evaluate the mathematical model. The solutions that are used for the governing equations are

$$\begin{aligned} u(x,y) &= u_0 + u_{xy} x \sin\left(\frac{a\pi xy}{L^2}\right) \\ v(x,y) &= v_0 + v_{xy} x \sin\left(\frac{b\pi xy}{L^2}\right), \end{aligned} \tag{3.3}$$

where the values for the coefficients can be found in Table 3.2. In order for the flow to be defined as incompressible the coefficients a and b must be equal while u_{xy} and v_{xy} must be of equal size but with different signs.

In order to evaluate the pressure correction of the solver the system of equations will be closed by submitting a solution for the pressure. For an incompressible solver it is possible to choose a reference pressure as the initial condition. For *simpleFoam* and *pimpleFoam* this would normally be set to zero, however for the terms in the governing equations related to pressure to be non-zero a trigonometric function has been constructed

$$p(x,y) = p_0 + p_y \cos\left(\frac{c\pi y}{L^2}\right). \tag{3.4}$$

The trigonometric term in the pressure equation will also be multiplied by $(xy(x_0 - x)(y_0 - y))^m$ where the values for x_0 and y_0 are determined by the coordinates for the two boundaries where $x, y \neq 0$. This ensures that the gradient normal to the boundary is zero in addition to a Neumann boundary condition at all boundaries of the domain if $m = 2$. The fields for the magnitude of the velocity and the pressure can be seen in figure 3.2a 3.2b.

By inserting the manufactured solutions (3.3) and (3.4) into the momentum equation for incompressible flow (2.12) with an added source term

$$\frac{\partial \vec{U}}{\partial t} + (\vec{U} \cdot \nabla)\vec{U} = -\nabla p + \nu \nabla^2 \vec{U} + S_U, \tag{3.5}$$

the source term S_U for the manufactured solution can be calculated. Note that the terms for external body forces has been neglected. If we look at each individual

component of the velocity each source term can be written as

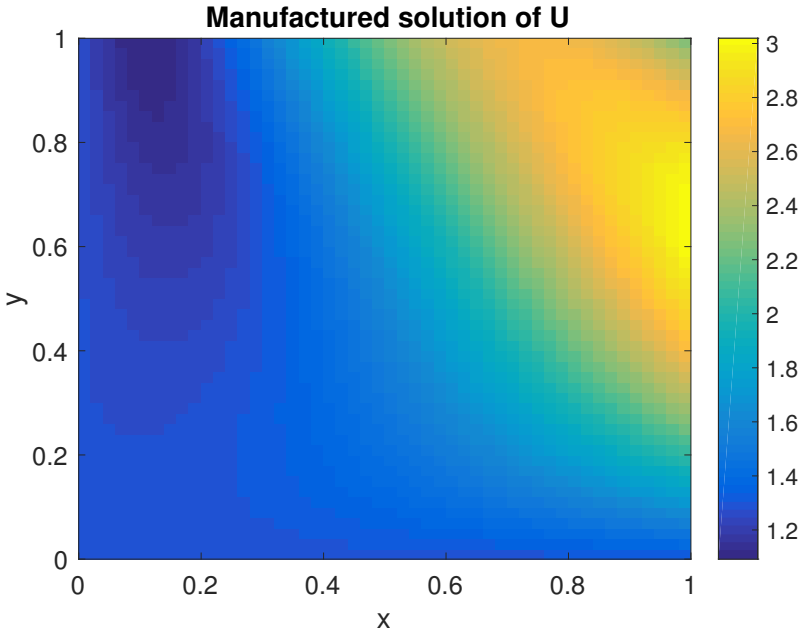
$$\begin{aligned} S_u &= \frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + \frac{\partial p}{\partial x} - \nu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \\ S_v &= \frac{\partial v}{\partial t} + v\frac{\partial u}{\partial x} + v\frac{\partial v}{\partial y} + \frac{\partial p}{\partial y} - \nu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right). \end{aligned} \tag{3.6}$$

Since these expressions are fairly long it might be beneficial to use a symbolic software to solve each system of equations. For this thesis the open source software sageMath has been used.

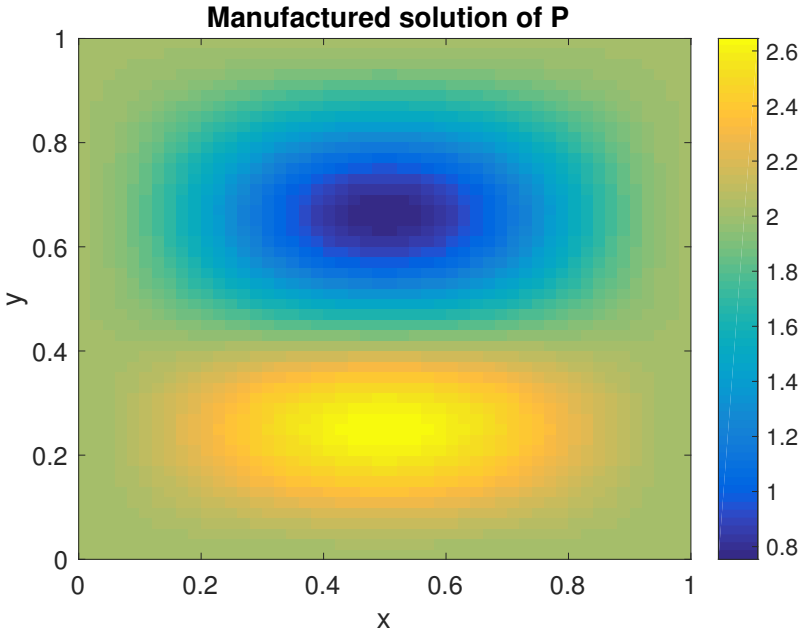
In order to ensure that all terms are of similar sizes the value of the viscosity ν will be set to a very high value. Since only the mathematical model is to be evaluated there are some benefits to be gained by making sure all terms are of equal size [10]. If the physical value of ν is selected the diffusive terms of the manufactured solution will be very small, a factor of 10^{-3} smaller than the convective terms. The value of ν has been set to 1 and can also be seen in table 3.2.

Table 3.2: Value of each coefficient used for manufactured solutions in compressible solvers

Constant	**Value**
a	0.8
b	0.8
c	0.7
u_0	1
u_{xy}	2
v_0	0.6
v_{xy}	-2
p_0	2
p_y	50
ν	1



(a) Velocity Field



(b) Pressure field

Figure 3.2: Manufactured solutions used for the incompressible solvers

3.4.2 MMS for Compressible flow

The setup of MMS for compressible flow will be very similar to the procedure used for the incompressible solvers. Since heat transfer within the fluid will be present for the compressible solvers a solution must be manufactured for the mass, momentum and energy equations. Since OpenFOAM uses temperature as the initial condition for the thermal conditions of the fluid the solution for the energy equation will be

based on a temperature field. Manufactured solutions to the governing equations are constructed with trigonometric terms according to

$$\begin{aligned}
\rho(x,y) &= \rho_0 + \rho_x \cos\left(\frac{a_x \pi x}{L}\right) + \rho_y \sin\left(\frac{a_y \pi y}{L}\right) \\
u(x,y) &= u_0 + u_x \cos\left(\frac{b_x \pi x}{L}\right) + u_y \sin\left(\frac{b_y \pi y}{L}\right) + u_{xy} \sin\left(\frac{b_{xy} \pi xy}{L^2}\right) \\
v(x,y) &= v_0 + v_x \cos\left(\frac{c_x \pi x}{L}\right) + v_y \sin\left(\frac{c_y \pi y}{L}\right) + v_{xy} \sin\left(\frac{c_{xy} \pi xy}{L^2}\right) \\
T(x,y) &= T_0 + T_x \cos\left(\frac{d_x \pi x}{L}\right) + T_y \sin\left(\frac{d_y \pi y}{L}\right) + T_{xy} \sin\left(\frac{d_{xy} \pi xy}{L^2}\right),
\end{aligned} \tag{3.7}$$

where the value of each constant parameter can be found in Table 3.3. The field for each manufactured solution can be seen in Figures 3.3, 3.4 and 3.5. As can be seen in these figures the boundary condition for density and temperature has been set to a Neumann boundary condition. The same procedure as earlier has also been used to ensure that the gradient normal to the boundary is zero. The reason for this is to ensure that the boundary condition for pressure, which is calculated from ρ and T, could be defined as *zeroGradient* in the boundary file.

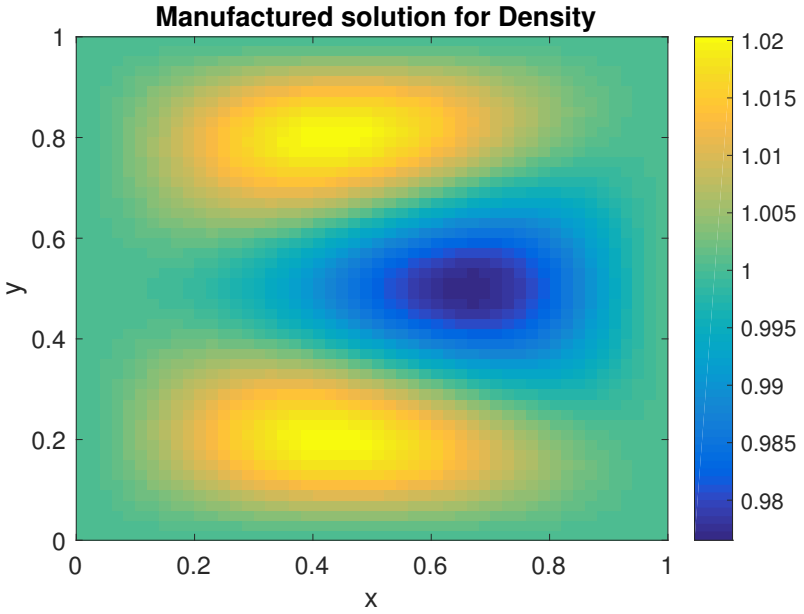


Figure 3.3: Manufactured solution for density used for verification of buoyant solver.

In order to close the system of equations a relationship for pressure must be implemented. If the thermophysical model based on the ideal gas law is selected the pressure can be derived from the ideal gas law. It is possible to write a relationship between the pressure and density according to

$$\rho = \frac{p}{R_s T}, \tag{3.8}$$

where R_s is the specific gas constant.

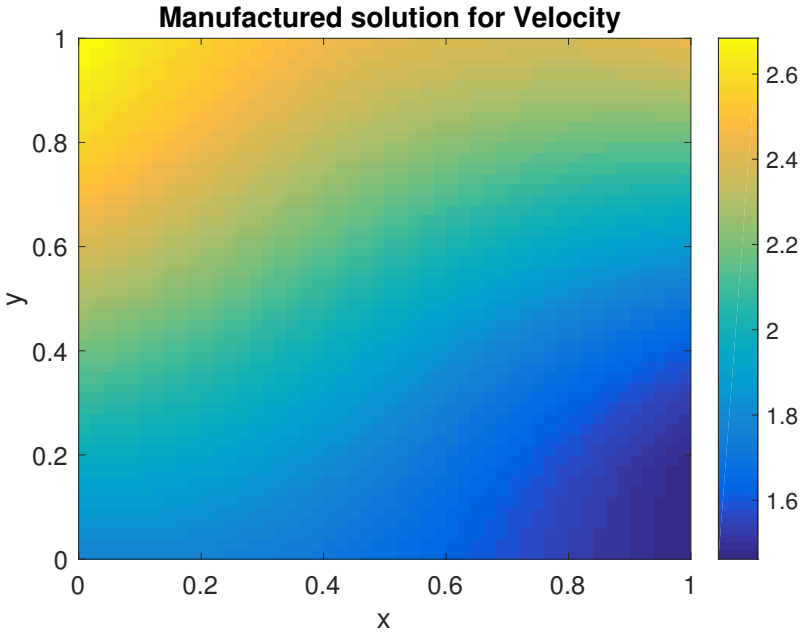


Figure 3.4: Velocity field used for MMS of *buoyantPimpleFOAM*

A relationship between the temperature and enthalpy must also be established in order to implement a source term for the energy equation. In order to evaluate the temperature from the enthalpy OpenFOAM solves the thermodynamic equation

$$c_p = \frac{dh}{dT}, \tag{3.9}$$

with the Newton-Raphson method. This means that the temperature for a given cell can be evaluated from the specific heat capacity and enthalpy in that cell. Due to this we can simply write the manufactured solution for temperature as enthalpy with the expression

$$h = c_p T. \tag{3.10}$$

One must also consider the buoyant pressure term used in the pressure correction algorithm for buoyant solvers. This pressure term is defined as

$$p_{rgh} = p - \rho \vec{g} \cdot \vec{r}, \tag{3.11}$$

where $\vec{g}$ is the gravitational acceleration vector and $\vec{r}$ is the position vector. By inserting this into the momentum equation (2.3) one can rearrange the terms for pressure gradient and external force applied from the gravitational acceleration

$$-\nabla p + \rho \vec{g} = -\nabla p_{rgh} - (\vec{g} \cdot \vec{r}) \nabla \rho. \tag{3.12}$$

If we insert the manufactured solutions (3.7) into the governing equations used in OpenFOAM and close the system of equations with (3.8), (3.10) and (3.11) the sources for each solution S_ρ, S_u, S_v and S_h can be determined just as for the incompressible case. Due to the size of the resulting expressions a software with a symbolic equation solver is recommended.

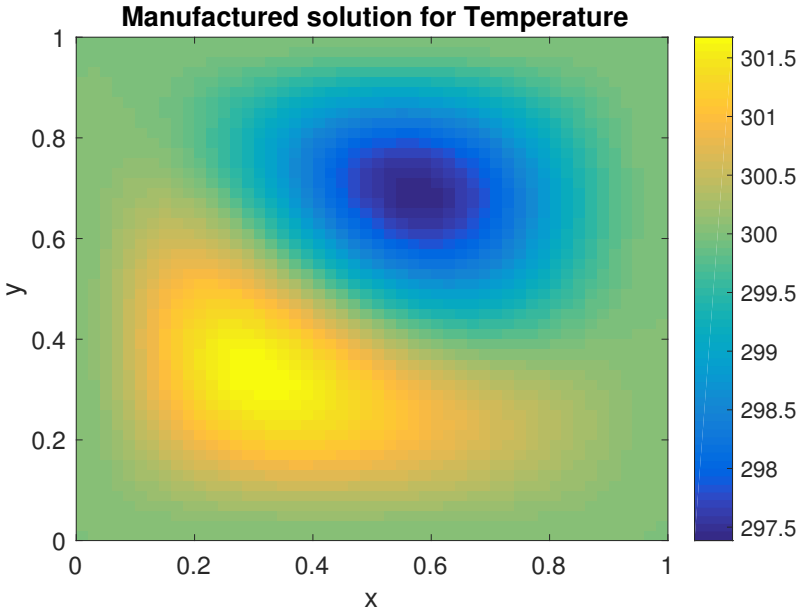


Figure 3.5: Manufactured solution of temperature used for order of accuracy assessment of numerical schemes.

For the selection of physical parameters linked to the properties of the fluid they will not be set to real physical values. In order for each source terms to to be of similar sizes the values used in the thermophysical model for dynamic viscosity μ, Prandtl number Pr and thermal diffusivity α will be set to unity. All data for physical and thermophysical constants can be found in table 3.4

Table 3.3: Values for each of the coefficients used for the manufactured solutions of compressible flow.

Index, i	$\mathbf{a_i}$	$\mathbf{b_i}$	$\mathbf{c_i}$	$\mathbf{d_i}$	$\rho_\mathbf{i}$	$\mathbf{u_i}$	$\mathbf{v_i}$	$\mathbf{T_i}$
0	-	-	-	-	1	1.5	1	300
x	0.3	0.9	0.7	0.8	2	0.5	0.5	-100
y	1	0.5	0.8	0.5	-2	1.2	-1	200
xy	0.3	0.8	0.8	0.9	3	-1	0.5	-100

Table 3.4: Value of physical constants used for MMS.

Constant	**Value**
R_s	287.10
g_y	-9.81
c_p	1010.1
μ	1
Pr	1
α	1

3.4.3 Implementation of sources with fvOptions

In order to implement the sources in OpenFOAM the already existing utility *fvOptions* is used. Within this utility it is possible to define a coded source where the equations for the sources can be directly implemented. The code will define all constant and look up the spatial coordinates which in turn is used to determine the value of the source in each cell.

As a final note it is important for the sources to be properly scaled and linearised. When discretizing each source the spatial parameters for each cell will be included in the equation [1], leading to multiplication with the volume of each cell. Therefore each source will be implemented according to

$$S_{\phi,i} = -S(\vec{r})V_i, \tag{3.13}$$

where the index i represents the cell index. Also note the negative sign which comes from the implementation of the source terms in the OpenFOAM solver. Examples of sources implemented in *fvOptions* can be found in the source code files of OpenFOAM [11].

3.5 Order of accuracy for Differential Operators

One of the tests defined in chapter 3.3 was designed to assess some of the derivative operators in OpenFOAM. Two fields with quite large variations in amplitude over a two dimensional domain were constructed (3.1). In order to evaluate the discretization of the differential operators a small application was created. The application calculated the field, evaluated the theoretical values in each cell, calculated a discretized solution and evaluated the discretization error over the whole domain (2.28). The test was performed for the gradient, divergence and laplacian of the appropriate field which can be seen in section 3.3. The results for the divergence of the vector field $\vec{g}$ can be seen in Figure 3.6. The numerical scheme used for the divergence was *Gauss linear* which uses Gauss integration with linear interpolation to discretize the divergence of the field [6]. It is expected that a large variation over a short spatial distance will cause an error when using linear interpolation. The order of accuracy for the divergence can be seen in Figure 3.7. We can see that for all but the first refinement of the mesh the change in discretization error follows the formal order of accuracy for the numerical scheme which is equal to 2.

The results from the discretization using the gradient scheme *Gauss Linear* can be seen in Figure 3.8. The majority of the contribution to the error over the whole domain defined as the discretization error will be present where the gradient varies over a short distance. Once again the order of accuracy follows the expected results as the mesh is refined, see Figure 3.7.

Finally the results from the discretized laplacian of the scalar field f can be seen

in Figure 3.9. The error for the laplacian is quite large since the values of the field will have a very large variation in amplitude over a short distance. The numerical scheme used for the discretization was once again based on linear interpolation. The order of accuracy results can be seen in Figure 3.7. The accuracy for the laplacian is somewhat lower than the expected values compared to the results from divergence and gradient. However the observed order of accuracy is again very close to the expected values once a few refinements have been performed.

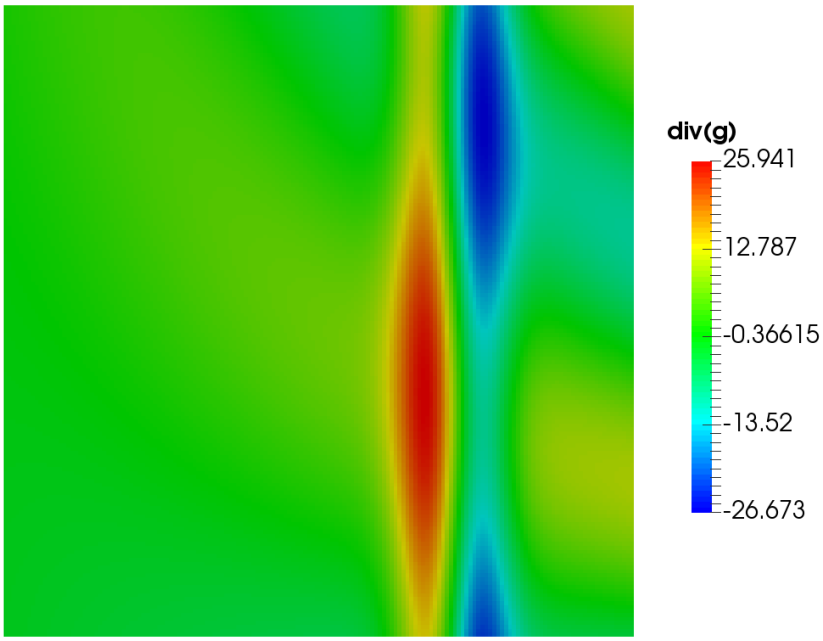


Figure 3.6: Divergence of the vector field $\vec{g}$.

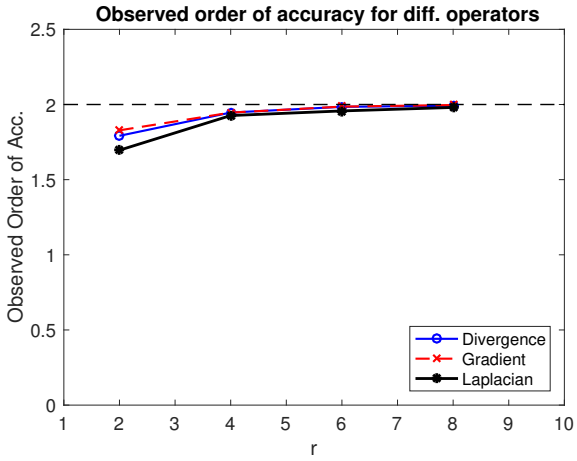


Figure 3.7: Observed order of accuracy for the discretization of various differential operators. The formal order of accuracy for the numerical schemes is equal to 2.

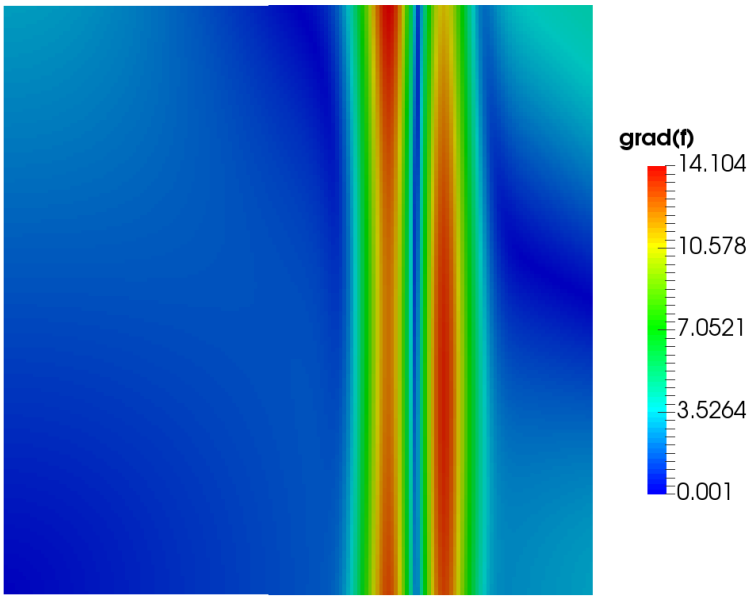


Figure 3.8: Gradient of the scalar field f defined for the test of discretization schemes for gradient operator.

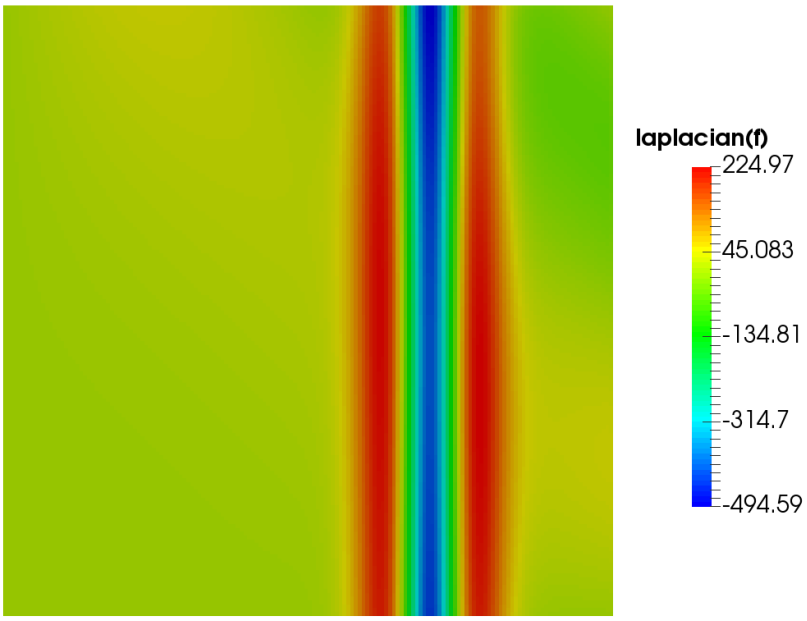


Figure 3.9: Discretized solution for the laplacian of field f.

From the results of the discretization error for each differential operator a good criteria for the pass and fail of a test should be defined for a refinement where the observed order of accuracy corresponds to the formal order of accuracy. The

criteria for pass and fail of the test will check the order of accuracy for a refinement in two steps. If the result is within a certain distance from the formal order of accuracy the test will pass, this limit has been defined as $\pm 10\%$ of the formal order of accuracy. Note that each of the simulations have only been evaluated in a 2-dimensional geometry. An evaluation in three dimensions might be an addition that could be added to the test harness in the future, it might also be beneficial to add tests of more mathematical operators. The test can easily be expanded into 3-dimensions by simply adding a number of cells in the z-direction of the mesh and by supplying a component for the vector field $\vec{g}$.

3.6 Numerical Simulations with MMS

The MMS has been used to verify the mathematical model for three different solvers and a few common numerical schemes [6]. Two incompressible solvers where evaluated; *simpleFoam* and *pimpleFoam* along with a solver used for the transfer of heat within the fluid, *buoyantPimpleFoam*.

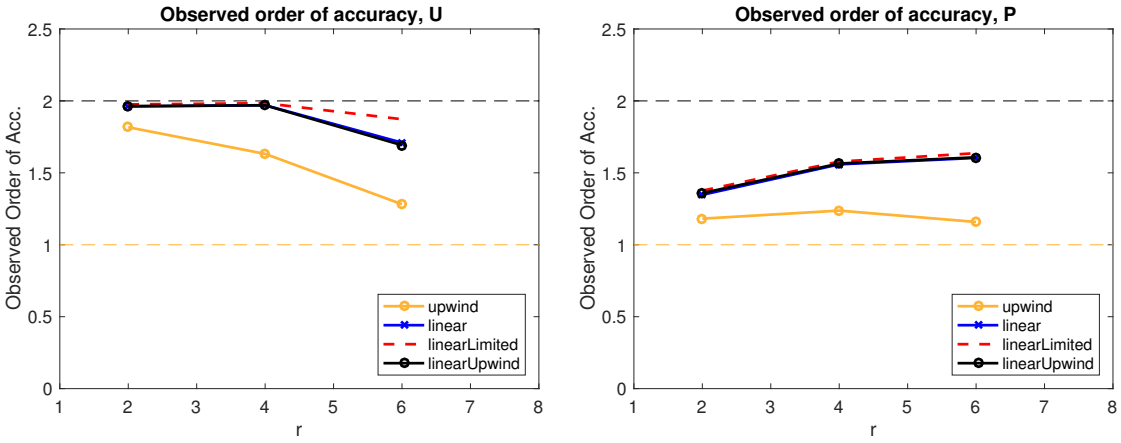
3.6.1 MMS and simpleFoam

In order to calculate the discretization error and order of accuracy for simulations based on the Navier-Stokes equations (2.12) an analytic solution to the governing equations must be available. The MMS discussed in section 3.4 is used to add a source term to the equations which will lead to a known solution. The manufactured solution used in the simulation can be seen in Equation (3.3). Note that the solutions used to derive the sources had to be that of an incompressible flow. If the flow is not incompressible the continuity of the mass flux inside the domain would not be conserved. The difference in mass flux going into the domain would not be equal to the outgoing flux.

The results for one of the simulations performed for an incompressible flow can be seen in Figures 3.11 where both the calculated error for velocity and pressure can be seen for the linear upwind scheme. Observed order of accuracy for mesh sizes ranging from 10×10 up to 160×160 cells was used for some of the more commonly used schemes. The accuracy is calculated from the discretization error taken for the whole domain, see Section 2.2.4. The schemes that where varied during the order of accuracy test was the divergence schemes for velocity. As expected the error is reduced for each refinement of the mesh, however one must take into consideration the runtime for each of the results. Due to the increased number of cells both the time of each iteration and the number of iterations is increased each time the mesh is refined. The order of accuracy results varies between the velocity components and the pressure. For the coarse meshes the observed order of accuracy is very close to the formal order of accuracy for all second order accurate schemes used during the simulation, the first order accurate upwind scheme seems to reach the formal order

of accuracy for the very fine meshes. Note that the L2-norm was used to provided a value for the observed order of accuracy for the velocity and pressure.

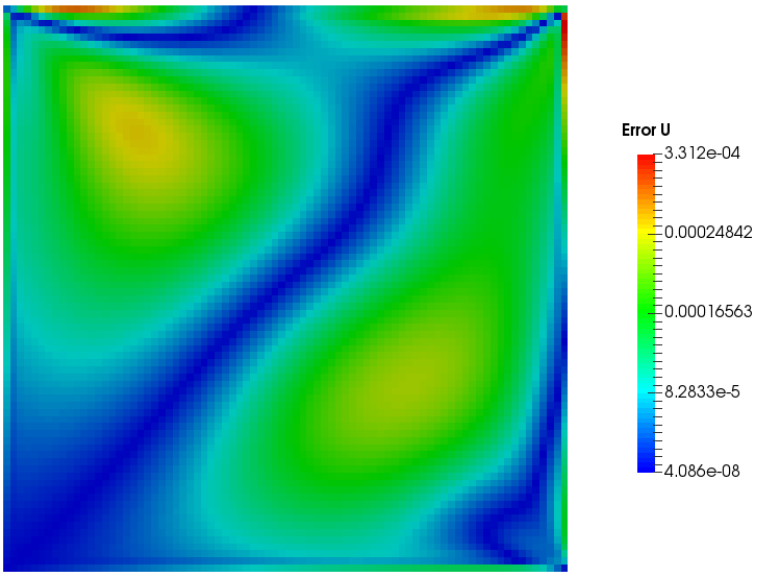
The order of accuracy for pressure is somewhat lower than the order of accuracy for velocity. The order of the change tends to be higher for a larger mesh for all types of divergence schemes. The discretization error for pressure is mainly present at two small regions close to the boundary for all types of divergence schemes. Usually the pressure is defined at the outlet of a domain during a simulation. Some problems might have been introduced when the boundaries for pressure was all set to *zeroGradient* in order to simplify the equations for the velocity.



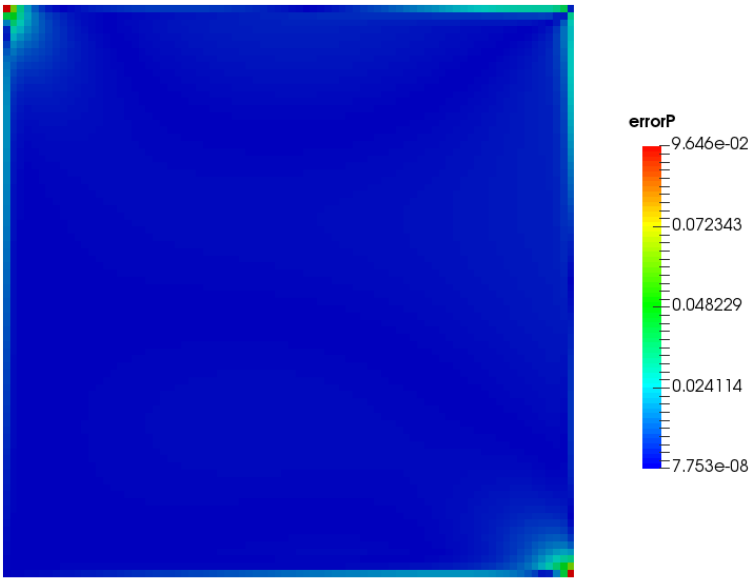
(a) Order of accuracy, velocity.

(b) Order of accuracy, pressure.

Figure 3.10: Comparison between observed order of accuracy and formal order of accuracy for various numerical schemes. The formal order of accuracy for linear schemes are equal to 2, while the accuracy for upwind should be equal to 1.



(a) Error for magnitude of velocity U.



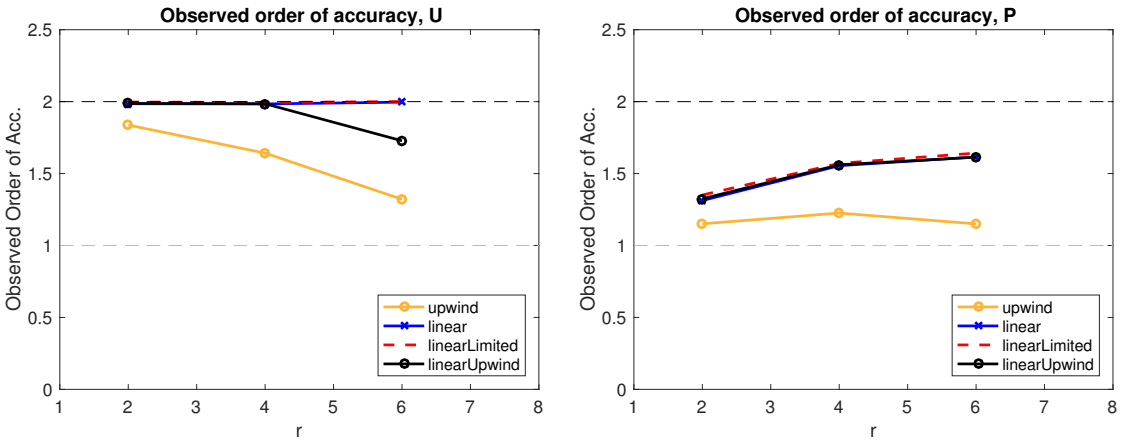
(b) Error for pressure p.

Figure 3.11: Results for verification of *simpleFoam*-solver. The error is used to calculate a discretization error which can be used to calculate order of accuracy.

3.6.2 MMS and pimpleFOAM

The merged PISO-SIMPLE pressure algorithm will also be evaluated for an incompressible flow. The same solutions to the equations will be used as for the *simpleFOAM* case, see 3.4. The *pimpleFOAM* solver is a transient solver which means special care must be taken into consideration when choosing a time step. In order to have a stable simulation the Courant number (Co) should not be higher than 0.5 [6].

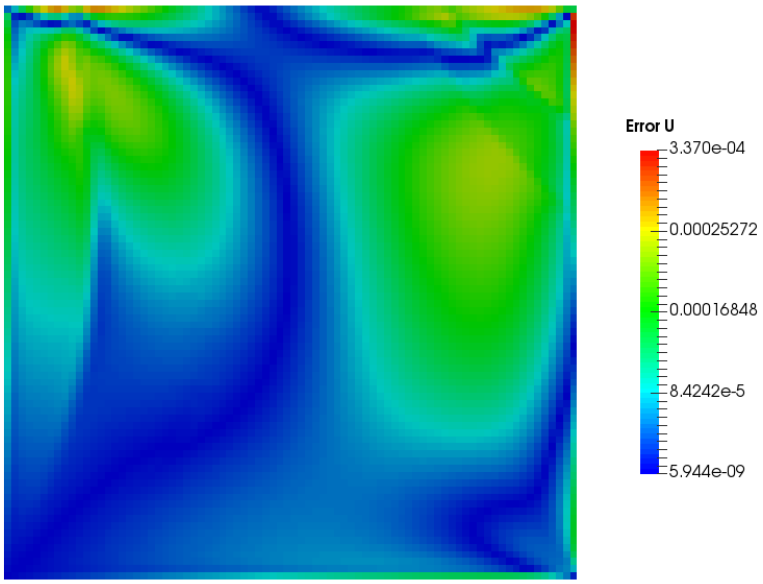
The results including the difference between manufactured solution and simulation are very close to the case for *simpleFOAM*. However the error distribution inside the domain has slightly better results for the lower left part of the domain, see Figure 3.13. The discretization error is once again used to calculate the order of accuracy for both the velocity and pressure. The observed order of accuracy can be viewed in Figure 3.12. Observed order of accuracy for the values of velocity is once again fairly close to the expected theoretical values of the second order accurate schemes. The accuracy for pressure is below the formal order except for the first order accurate linear upwind scheme. A further refinement of meshes for the second order schemes could be necessary in order for the accuracy of pressure to be closer to formal values.



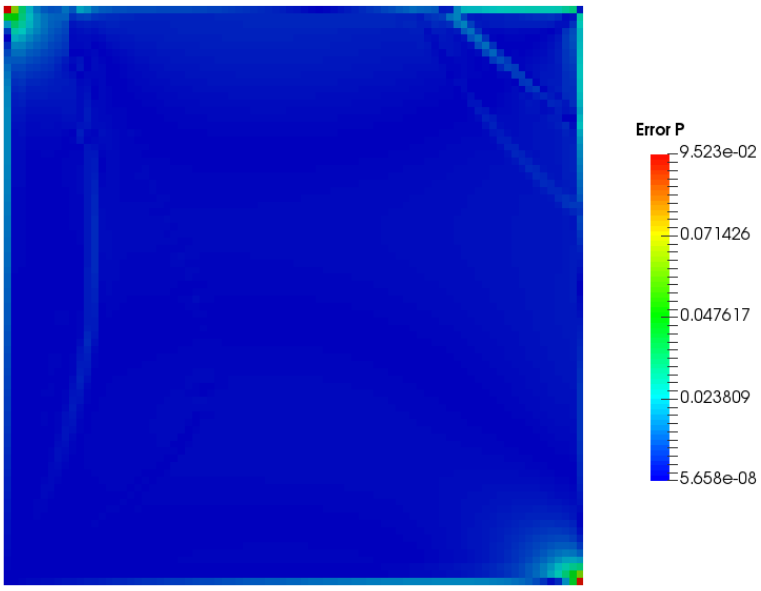
(a) Order of accuracy, velocity.

(b) Order of accuracy, pressure

Figure 3.12: Comparison between observed order of accuracy and formal order of accuracy for the transient solver *pimpleFoam*. The formal order of accuracy is for linear schemes is equal to 2, accuracy for upwind scheme is 1.



(a) Error for magnitude of velocity.



(b) Error for pressure field.

Figure 3.13: Results for verification of *pimpleFoam*-solver. The error is the difference between simulated result and manufactured solution

3.6.3 MMS and buoyantPimpleFoam

The compressible solver which also simulates the transfer of heat inside the system for buoyant flow was also evaluated with the method of manufactured solution. Since a compressible flow is to be simulated a variation in density will be manufactured together with a solution for the velocity and temperature. The solutions that are implemented in order to calculate the source terms and the relationship between pressure, temperature and enthalpy can be seen in section 3.4.2. The field for enthalpy will have the same shape as the result for temperature since the calculation of the temperature is based on the enthalpy values in each cell.

The fields for density, velocity and temperature obtained from the simulation are compared to the manufactured solutions. The resulting error for density can be seen in Figure 3.14. The difference between simulation and manufactured solution for velocity can be found in Figure 3.15, finally the result for specific enthalpy (or temperature) can be found in Figure 3.16. Due to some problems with the implementation only the order of accuracy for velocity and enthalpy will be calculated, which can be found in Figure 3.17. The order of accuracy for velocity will follow the formal order of accuracy for the chosen numerical scheme, upwind biased linear interpolation. Order of accuracy for specific enthalpy is lower than the expected value of 2 for the limited linear scheme. The discretization error for density seems to be independent of the mesh refinement.

The main problem with the compressible case seems to be the implementation of the equation of state. The density is calculated from the pressure and temperature which seem to give the same error and is not dependent on the mesh size. For future work; one solution to this problem might be to implement another thermophysical model and equation of state only based on temperature values. The system of equations would then be closed with a manufactured solution for pressure.

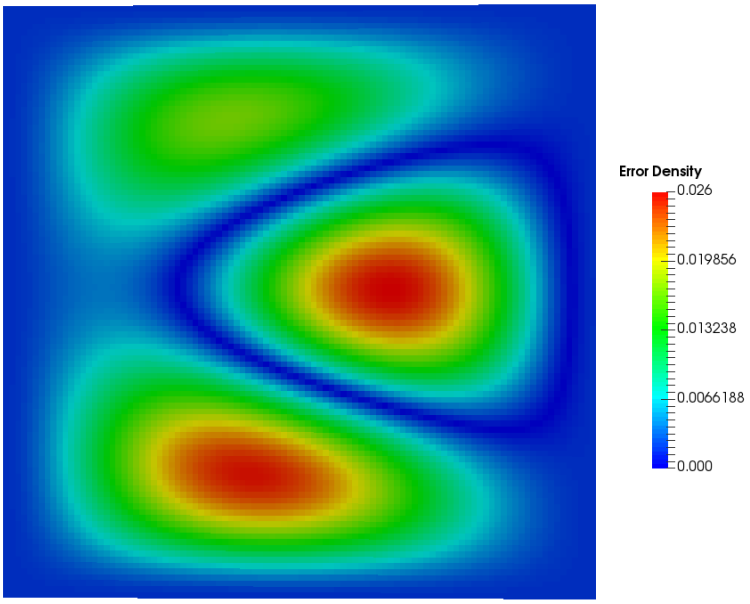


Figure 3.14: Error for density.

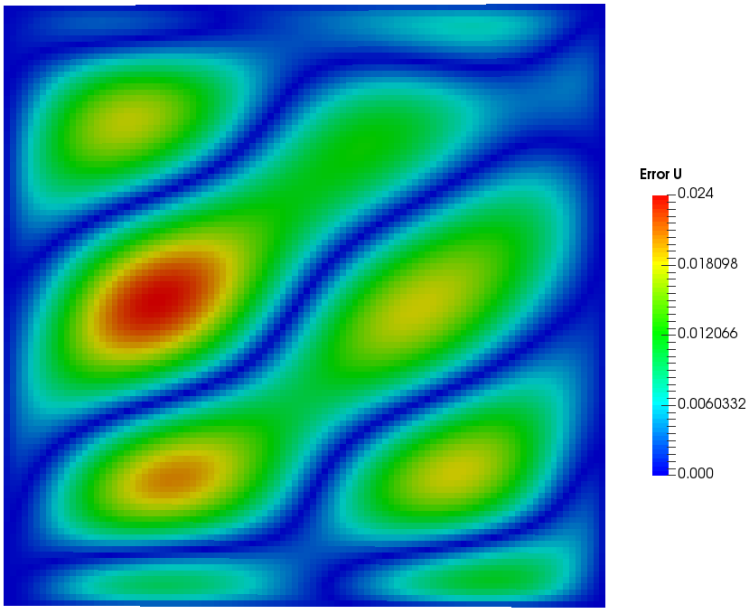


Figure 3.15: Error for magnitude of velocity field.

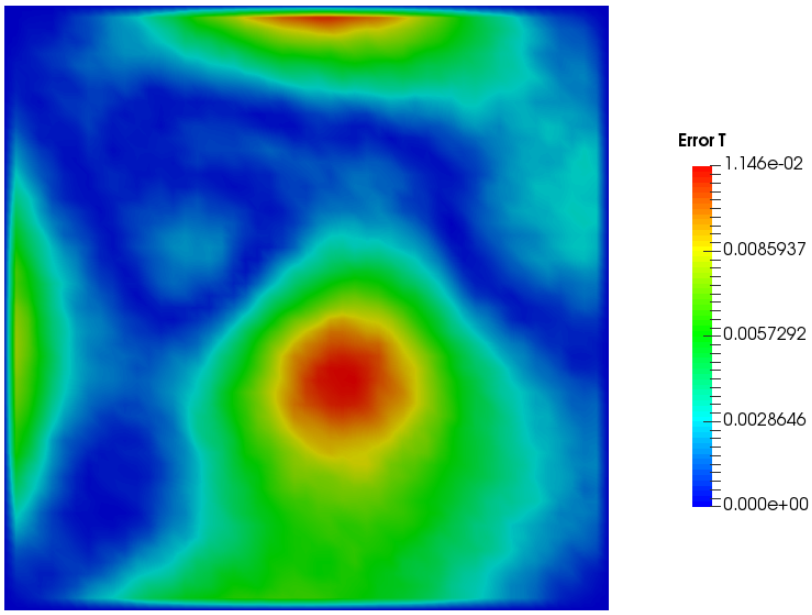


Figure 3.16: Error for temperature.

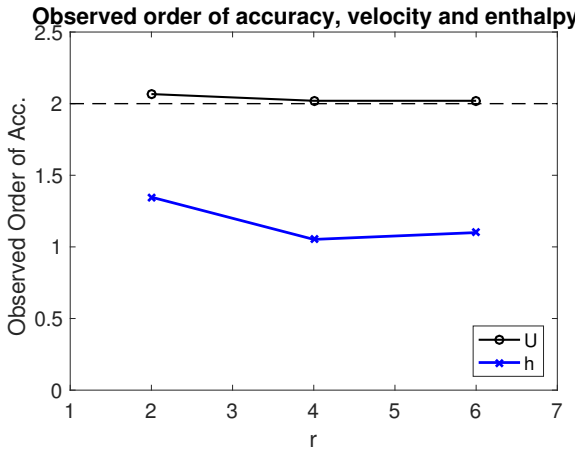


Figure 3.17: Observed order of accuracy for the second order numerical schemes used to calculate velocity and temperature with MMS for compressible NS-equations.

Chapter 4

Validation

The case for validation has been designed from an experiment used to benchmark various CFD simulations in the past. The experiment was performed at Vattenfall Research and Development and has later been used as a blind benchmark by OECD/NEA [12]. The case has been split into two different simulations where the size of the computational domain has been varied. Due to the amount of computational time needed to evaluate a case with a large number of cells the meshes used for validation will be limited to a few hundred thousand cells.

4.1 T-junction

The setup is based on a t-junction that connects two incoming flows and mixes the fluid into single outlet. The incoming fluid will be liquid water with different temperatures, one cold inlet and one hot inlet. Due to the difference in temperature of the incoming fluids thermal mixing will be present after the junction. An overview of the setup can be seen in Figure 4.1. The figure also shows the distance from the junction to each point where the temperature is being measured. At each location from $x = 2D$ up to $x = 8D$ four points where used to measured the temperature at a distance of 1 mm from the wall. The four points are located at the top of the pipe (0°) right wall when facing the flow direction (90°), bottom wall (180°) and left wall (270°). The locations for data acquisition of velocity will also be located at various points after the fluids have mixed. The velocity profiles for both the horizontal- and vertical directions will be saved at a distance of $1.6D$, $2.6D$, $3.6D$ and $4.6D$ from the junction. In order to generate output files for both temperature and velocity data at the specified locations the OpenFOAM utility for probes will be used. The probes store the value of selected fields at a specified time interval. All of the data can then be used during the post process to generate mean values used to calculate the validation metric.

The computational mesh was generated by *cfMesh* which is a mesh generating tool

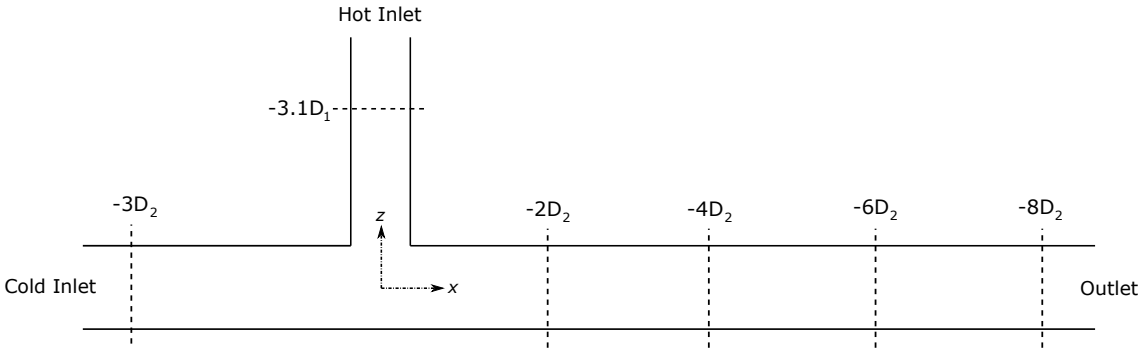


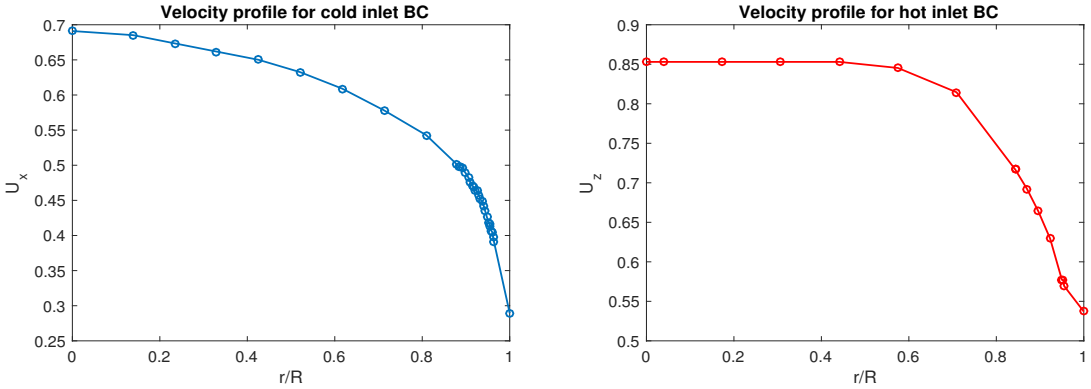
Figure 4.1: Overview of the t-junction used as a validation case. The figure also shows the location of measurements for incoming velocity profiles and temperature measurements.

implemented in OpenFOAM. The geometry surface files was generated with the open source tool SALOME [13], the geometry files are then imported into OpenFOAM and used to generate the mesh. For simplicity the mesh was generated with the Cartesian mesh application available within OpenFOAM. In order to achieve a reasonable runtime for a validation simulation the size of the meshes to be evaluated are $\sim 200k$ and $\sim 400k$ cells. From the concluding report of the blind benchmark [12] of the same experiment it was clear that a rather coarse mesh can give results accurate enough for a validation test case such as this.

4.1.1 Boundary Conditions

The incoming flow for both the hot- and cold inlet is a fully developed turbulent flow. A fully developed turbulent flow is defined so that the mean velocity profile will have the same amplitude as you move downstream of the flow [1]. To achieve this the experiment used a very long channel with no obstacles before the junction. This process would demand a very large domain if the same procedure is used for the simulation, which is not suitable due to the very long computational time. In order to achieve a fully developed turbulent flow a special boundary condition is used to add a random noise to the velocity which represents the fluctuating part of the turbulent flow. The velocity profiles from the experiment is interpolated onto the circular boundary patch by modifying the *turbulentInlet* boundary condition. The modified boundary condition will read the velocity data as a function of r from the case directory and interpolate the data onto any point on the boundary. The only input needed for the interpolation is the center position of the boundary, velocity data as a function of radius and fluctuation parameters used to simulate turbulence within the flow. The experimental velocity profiles for both hot- and cold inlet can be seen in Figure 4.2. These profiles have been measured perpendicular to the flow direction for both incoming flows at a distance of $-3D_2$ and $-3.1D_1$ along the y- and z-axis, see 4.1. The data for all boundary conditions can be found in the summary of benchmarks performed at the t-junction [12]. The boundary conditions for temperature can be found in Table 4.1. This table also lists the diameter of each

pipe and the volumetric flow rates used to calculate the bulk velocity which is used to scale the velocity profiles.



(a) Cold inlet. **(b)** Hot inlet.

Figure 4.2: Velocity profiles used to generate the fully developed turbulent flow at both inlets.

Table 4.1: Inlet data for validation case.

Inlet	Temp. [C°]	Vol. flow rate [l/s]	Bulk velocity [m/s]	Pipe Diameter [m]
coldInlet	19	9	0.59	0.14
hotInlet	36	6	0.77	0.10

4.1.2 Discretization Schemes

The discretization schemes that are used for each of the terms in the governing equations can be found in Table 4.2. The chosen schemes were based on the recommendations found in the OpenFOAM guide and tutorial cases for LES simulations [6], only second order accurate schemes where chosen.

Table 4.2: Numerical schemes used for the simulation of a t-junction.

Term	Numerical Scheme
ddtScheme	backward
grad	linear
div(phi,U)	linearUpwind
div(phi,k)	limitedLinear
div(phi,h)	limitedLinear
div(phi,K)	limitedLinear
div((rho*nuEff)*dev2(T(grad(U))))	linear
laplacian	linear corrected
interpolation	linear

4.1.3 Turbulence model

In order to resolve both large and small eddies in the turbulent flow a LES turbulence model is used. The specific model used in the dynamic one-equation eddy-viscosity model *dynamicKEqn* [1]. The specific coefficients for the spatial filter function will be left as the standard inputs. The filter parameters are represented by the *deltaModel* which is set by *cubeRootVolDelta* which is a simple model that uses the cell volume as filter parameter [11].

4.1.4 Thermophysical model

The thermophysical model that is used is based on temperature dependent properties of the fluid. Properties related to the transport of the fluid will be determined from the *polynomial*-model. This model will determine the values of dynamic viscosity μ and thermal conductivity κ from a polynomial function of the temperature, see Section 2.1.7. The polynomial function has been calculated from thermo physical data provided by the ASME steam tables [14]. The thermodynamic model used for this case is *hPolynomial*, this is also based on a temperature dependent polynomial. The heat capacity c_p which is used to calculate the temperature from the energy equation (2.11) is defined from (2.19). Finally the equation of state that is used by the model will also be defined by a polynomial function of temperature, *icoPolynomial*. A polynomial function for $\rho(T)$ is defined just as for the previous quantities.

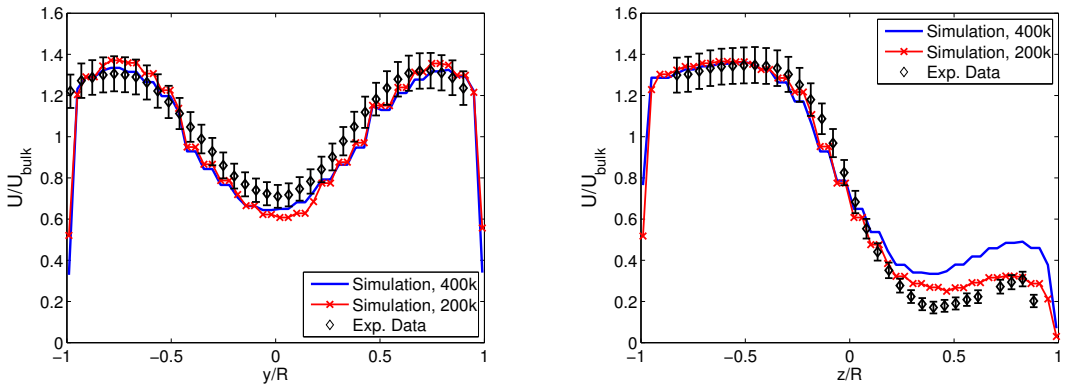
4.2 Validation of thermal mixing in t-junction

The results from the simulation of flow velocity will be represented with velocity profiles taken at a certain distance from the junction. The magnitude of the velocity will be divided with the bulk velocity U_{bulk} of the main channel. Each result is calculated from a mean value gathered from velocity measurements with a frequency of 1000 Hz. The temperature data is gathered for a few given points close to the wall defined in Section 4.1. The time between each measurement is taken at the same interval as for the velocity measurements. The results for temperature are presented with a dimensionless unit T^* defined according to $T^* = \frac{T-T_{hot}}{T_{hot}-T_{cold}}$ where T_{hot} is the temperature of the hot inlet and T_{cold} in the temperature of the cold inlet. After the flow has been fully developed the simulation was left to run for 25 s in order to reduce the impact of fluctuating terms.

The velocity results for both horizontal- and vertical direction of the cross section from the simulation can be seen in figures 4.3 to 4.6. The results are shown for two different meshes with $200k$ and $400k$ cells, this ensured a rather quick runtime for each validation test. The overall result for the simulation lies fairly close to the

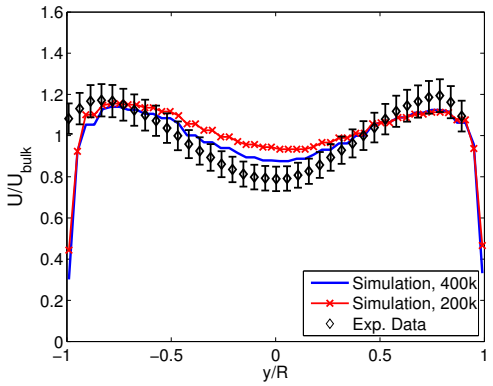
experimental data, especially for velocity profiles located further downstream from the junction. A large part of the results lie within the estimated uncertainty of the experimental values taken from [12]. The major outliers are an overestimated velocity at the top of the pipe close to the junction, see Figure 4.3b and 4.4b. Another noteworthy difference is the small dip in velocity close to the center of the pipe for the horizontal profiles. It seems that a more refined mesh is needed to capture this effect, however this was also the case for the benchmark results with a low number of cells presented in [12].

The validation metric used for the test criteria was defined as the estimated error between simulated values and experimental data at the location for experimental measurements. The simulated result will be interpolated to the coordinates of the experimental data and an estimated error for the whole profile will be evaluated with the $L2$-norm (2.28). The estimated error will only be calculated at two positions, the closest to the junction and the one furthest away from the junction. Only two positions were chosen in order to limit the number of results to be handled, however the option to include all results should be easy to implement since all data is stored within the exact same format with the post-process utilities of OpenFOAM. The criteria for a passed test of velocity parameters can be set to percent error based on the bulk velocity for the main pipe.

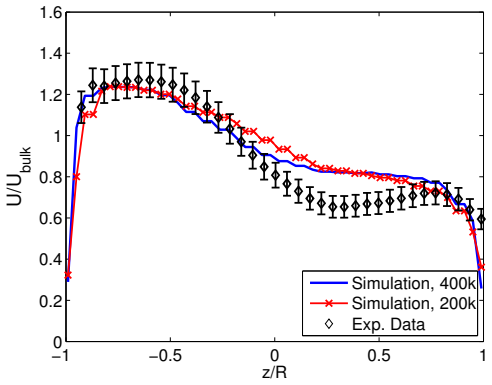


(a) Time averaged velocity, x=1.6D z=0. **(b)** Time averaged velocity, x=1.6D y=0.

Figure 4.3: Velocity profiles measured at a distance of $x = 1.6D$ from the junction.

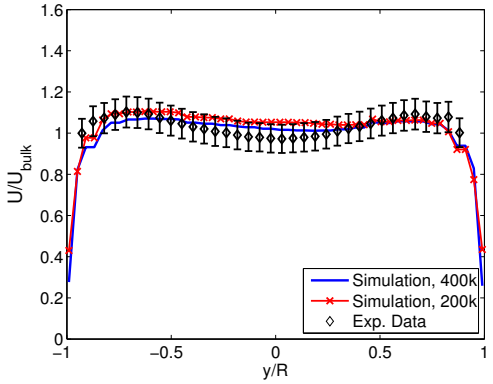


(a) Time averaged velocity, x=1.6D z=0.

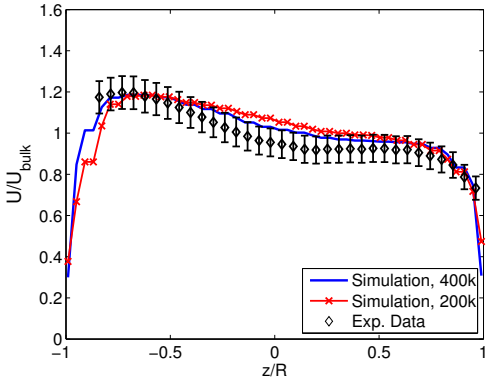


(b) Time averaged velocity, x=1.6D y=0.

Figure 4.4: Velocity profiles measured at a distance of $x = 2.6D$ from the junction.

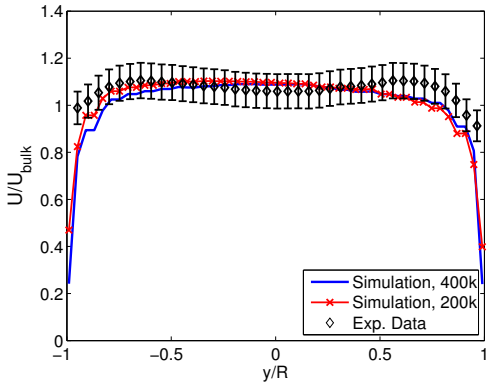


(a) Time averaged velocity, x=3.6D z=0.

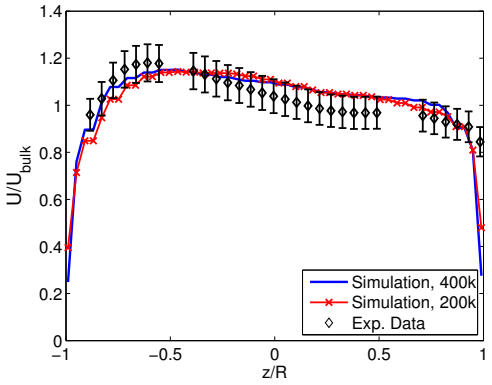


(b) Time averaged velocity, x=3.6D y=0.

Figure 4.5: Velocity profiles measured at a distance of $x = 3.6D$ from the junction.



(a) Time averaged velocity, x=4.6D z=0.



(b) Time averaged velocity, x=4.6D y=0.

Figure 4.6: Velocity profiles measured at a distance of $x = 4.6D$ from the junction.

The overall shape of the temperature distribution at the end of the simulation can

be seen in Figure 4.7. We can see that there are fairly large zones where the temperature fluctuates by up to 10 K. Results for temperature measurements close to the wall at various locations can be found in Figures 4.8a-4.8d. The values for temperature follow the same trend as the experimental data except for the temperature measurements located at the top of the pipe. This might be explained by the increased velocity which was measured for the top of the pipe close to the junction. After the fluids have mixed the temperature may vary by quite a large margin between two positions located close to each other. An increased velocity should interfere with the temperature results since these fluctuating zones may be moved slightly, which would change the temperature results that are only gathered at a few selected points inside the pipe.

The test criteria used for temperature validation will be based on the same principle as the data for velocity. The metric will be based on the difference between the simulated results and the experimental data. Not all points had data available from the experiments so these points will have to be omitted from the results. Once again a single value for the whole domain is calculated from the norm in order to achieve a very simple criteria for pass/fail.

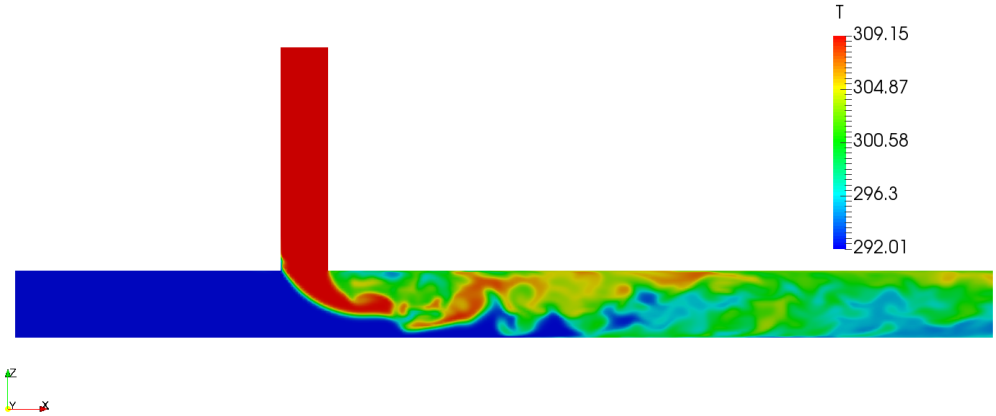
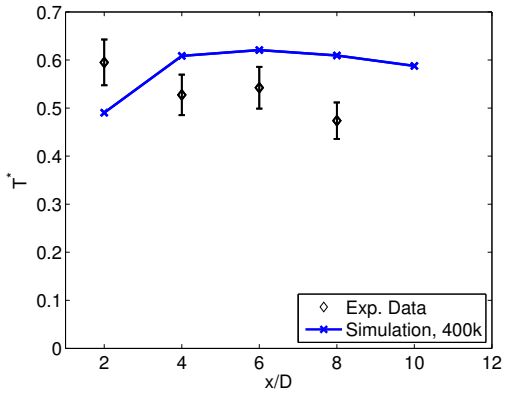
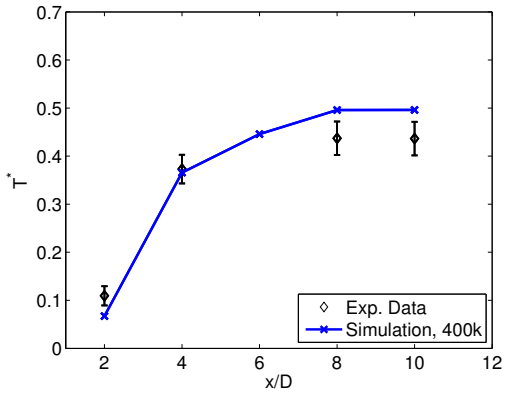


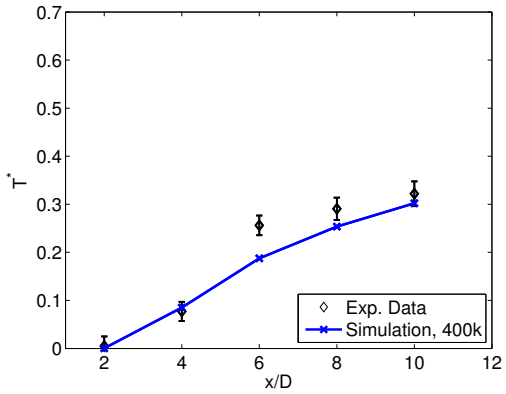
Figure 4.7: Visualization of the temperature distribution when the flow is fully developed.



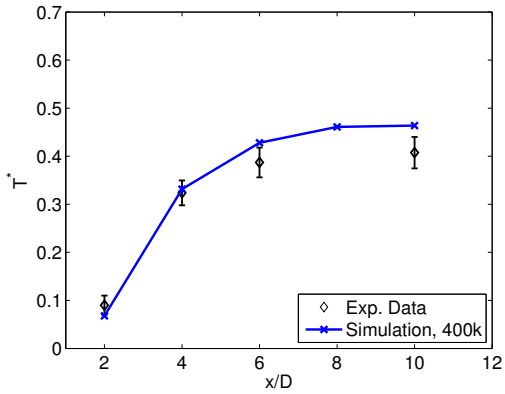
(a) Time averaged T, z=D/2.



(b) Time averaged T, y=D/2.



(c) Time averaged T, z=-D/2.



(d) Time averaged T, y=-D/2.

Figure 4.8: Mean temperature 1 mm from the wall at specified locations after the t-junction.

Chapter 5

Conclusions

In order to ensure safe long term operation of a nuclear power plant the degradation and change of material properties due to thermal transients must be evaluated. The main process of the power plant is to generate thermal energy which then can be converted into electrical energy. This may introduce large temperature fluctuations which can generate thermal stress in various materials. Computational fluid dynamics is one computational tool that can be used to estimate the fluid properties and transient behavior of the flow, the result can then be used as a basis for thermal transient evaluations. OpenFOAM is an open source CFD software that can be used to simulate various properties of a flow.

A test suite for verification and validation of OpenFOAM was developed in order to test some of the mathematical models and compare simulated results to existing experimental data. The goal was to design a number of tests which can be used to evaluate present and future releases of OpenFOAM. Another main future which was emphasized was the ability to add new test cases or edit existing ones for future work. Each test will be performed automatically after it has been initialized and the results are presented to the user after each test has finished.

The verification procedure was designed to evaluate some aspects of the mathematical models used when performing a numerical calculation. To solve the governing equations various numerical schemes are used to create discretized equations which are solved for a specified control volume. In order to evaluate the performance of each test a discretization error is evaluated over the whole computational domain. Since it is difficult to obtain exact analytic expressions from the governing equations, which is needed in order to evaluate the discretization error, the method of manufactured solutions is used. An exact solution to the problem is obtained and can be used to calculate a source term which is implemented in OpenFOAM, the simulated result should then be consistent with the manufactured solution. The method of manufactured solution is used to evaluate two incompressible solvers, *simpleFoam* and *pimpleFoam*, and one compressible solver, *buoyantPimpleFoam*.

One problem when using the discretization error is that a subjective analysis based

on the numerical values and application of the model must be done. It can instead be used to calculate the observed order of accuracy for a given simulation by subsequent refinement of the computational mesh. The observed order of accuracy should be consistent with the theoretical order of accuracy of the numerical scheme used to solve the specified problem. Both discretization error and order of accuracy can be used as a criteria for evaluating the verification tests. The order of accuracy may be beneficial for verification tests since no further evaluation is needed if the result is close to the formal order of accuracy of the numerical scheme. However care must be taken to ensure that the accuracy is evaluated in the asymptotic region where a refinement of the mesh represents a correct decrease in discretization error.

The validation test is based on an experiment of thermal mixing within a t-junction. The experiment has been used to benchmark various CFD simulation in the past. The main goal of the validation test was to perform a larger simulation based on the LES-turbulence model, but still try to keep the computational time within a reasonable limit. The validation was performed for two different meshes, the results indicate that even a small number of cells, $200k$, could be used as a suitable test case. Due to the transient nature of the turbulent flow, the simulated data had to be gathered until a mean values for velocity and temperature could be compared to the experimental data. The validation metric used as a test criteria is based on the same principle as for the verification tests. A discretization error is calculated for the entire domain, this is then used to evaluate if the model is accurate enough for a pass or fail.

5.1 Future Work

Future work that could be implemented is mostly related to the verification procedure. When evaluating the order of accuracy the tests could be expanded into three dimensions, the limitation of two dimensions should not be needed if the simulations are rather simple and fast to execute. It would also be beneficial to put more work into the cases where the method of manufactured solution is used, especially for compressible solvers. It took quite some time to implement the manufactured solution, this limited the amount of time used to evaluate the results and improve the implementation. Another question might be the impact on other types of meshes for the method of manufactured solution cases. What is the result for unstructured meshes where all cells have different volumes.

Some work may also have to be performed on the entire test suite. Future changes of OpenFOAM might change how the input files and various settings are defined. This would require some work in order to ensure each test is able to perform the simulation.

Bibliography

[1] H.K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics : the Finite Volume Method*. Harlow, USA: Pearson Prentice Hall, 2007.

[2] CFD Direct Ltd. *About OpenFOAM*. 2017. URL: `http://cfd.direct/openfoam/about/` (visited on 01/30/2017).

[3] Joel H. Ferziger and Milovan Perić. *Computational Methods for Fluid Dynamics*. Berlin, Germany: Springer-Verlag, 1999.

[4] CFD Direct Ltd. *Energy Equation in OpenFOAM*. 2016. URL: `https://cfd.direct/openfoam/energy-equation/` (visited on 03/12/2017).

[5] Lars davidsson. *An introduction to turbulence models*. Gothenburg, Sweden: Chalmers university of Technology, 2003.

[6] CFD Direct Ltd. *OpenFOAM User Guide*. 2016. URL: `http://cfd.direct/openfoam/user-guide/` (visited on 01/27/2017).

[7] William L. Oberkampf and Christopher J. Roy. *Verification & Validation in Scientific Computing*. Cambridge, United Kingdom: Cambridge University Press, 2010.

[8] Patrick J. Roache. *Verification and Validation in Computational Science and Engineering*. Albuquerque, N.M.: Hermosa publishers, 1998.

[9] Kambiz Salari & Patrick Knupp. "Code Verification by the Method of Manufactured Solution". In: (2000).

[10] Aniruddha Choudharya & Christopher J. Roya & Edward A. Lukeb & Subrahmanya P. Veluric. "Code verification of boundary conditions for compressible and incompressible computational fluid dynamics codes". In: (2016).

[11] OpenFOAM. *OpenFOAM Foundation development repository*. 2017. URL: `https://github.com/OpenFOAM/OpenFOAM-dev` (visited on 04/28/2017).

[12] B. L. Smith & J. H. Mahaffy & K. Angele & J. Westin. "Report of the OECD/NEA-Vattenfall T-Junction Benchmark exercise". In: (2011).

[13] Salome. *SALOME 8.2.0 Documentation*. 2017. URL: `http://www.salome-platform.org/user-section/documentation/current-release` (visited on 03/15/2017).

[14] Allan H. Harvey & William T. Parry & James C. Bellows & John S. Gallagher & Richard D. Harwood. *ASME International Steam Tables for Industrial Use*. New York, NY: ASME Press, 2009.