ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

DOTTORATO DI RICERCA IN

MECCANICA E SCIENZE AVANZATE DELL'INGEGNERIA

Ciclo 37

Settore Concorsuale: 09/C2 - FISICA TECNICA E INGEGNERIA NUCLEARE

Settore Scientifico Disciplinare: ING-IND/19 - IMPIANTI NUCLEARI

DEVELOPMENT OF A CFD TOOL FOR TURBULENT NATURAL CONVECTION AND HEAT TRANSFER SIMULATIONS OF LIQUID METALS

Presentata da: Lucia Sirotti

Coordinatore Dottorato

Lorenzo Donati

Supervisore

Sandro Manservisi

Esame finale anno 2025

Borsa di dottorato del Programma Operativo Nazionale Ricerca e Innovazione 2014-2020 (CCI 2014IT16M2OP005), risorse FSE REACT-EU, Azione IV.4 “Dottorati e contratti di ricerca su tematiche dell’innovazione” e Azione IV.5 “Dottorati su tematiche Green.

Codice CUP: J35F21003210006

Contents

Abstract

This dissertation investigates numerical techniques for studying turbulent natural convection and turbulent heat transfer systems involving liquid metals. Specifically, two strategies are explored. The development of stand-alone solvers to account for all the occurring phenomena, and a code coupling strategy, where two or more numerical codes are integrated to exploit the different code peculiarities. In this work, the latter approach is realized by coupling the in-house finite element code FEMuS with the finite volume code OpenFOAM, using the open-source MED library for data exchange.

Turbulent natural convection is studied in a Differentially Heated Cavity configuration. An anisotropic four-parameter turbulence model is implemented in the FEMuS code and validated against the DNS benchmark for liquid metal-filled cavities. To extend this analysis, the coupling application is first validated in the laminar natural convection regime and then applied to the turbulent case. The volume data transfer algorithm is used to leverage the more accurate thermal turbulence model of the FEMuS code with the extensively validated dynamic solver of OpenFOAM.

Turbulent heat transfer is investigated in a liquid metal heat exchanger configuration. A boundary data transfer algorithm is validated using a Conjugate Heat Transfer problem, where thermal coupling occurs between the fluid and solid domains. This technique is then applied to a finned pipe heat exchanger, with FEMuS simulating the turbulent liquid metal flow and OpenFOAM modeling heat conduction in the solid structure.

Introduction

Liquid metals represent a cutting-edge innovation in the energy sector, particularly as heat transport fluids. Due to their ability to remain liquid over wide temperature ranges and their high thermal conductivity, liquid metals represent a valid alternative to traditional heat transfer fluids. These unique properties make liquid metals suitable for demanding applications with high thermal loads. In such systems, liquid metals could increase their operating temperature and thus the plant's efficiency. Over the years, liquid metals have been investigated for use in Concentrated Solar Power (CSP) plants [1, 2] and Generation IV nuclear reactors [3, 4]. In recent decades, Lead-cooled Fast Reactors (LFRs) and Sodium-cooled Fast Reactors (SFRs) have emerged as some of the most promising technologies for the next generation of nuclear reactors. In Europe, research and development for these reactors have reached an advanced stage, culminating in the conceptualization and design of a large-scale demonstrator, the Advanced Lead Fast Reactor European Demonstrator (ALFRED) [5, 6, 7, 8]. Regarding solar energy, initiatives such as the NEXTower project have also driven the use of liquid metals for CSP systems operating at high temperatures [9, 10, 11]. These power plants are classified as multiscale and multiphysics systems, as they involve physical processes across various spatial scales and interactions of physical phenomena in a strongly coupled framework. The study of multiscale and multiphysics problem has experienced significant advancement thanks to Computational Fluid Dynamics (CFD) tools. However, simulating these complex phenomena requires multiple physics and domains, which still presents significant

challenges for computational tools. Over the years, two primary strategies have been developed to address these challenges: the *monolithic approach*, which involves creating a unified numerical code to model all relevant phenomena, and the code *coupling approach*, which combines existing validated codes to leverage their strengths and specific capabilities. For this purpose, both strategies are explored in this Thesis. The in-house finite element code, FEMuS, and the well-established finite volume code, OpenFOAM, are used both as stand-alone solvers and as subsystems within a coupling framework. The coupling application is developed using the open-source MED library to numerically integrate the FEMuS and OpenFOAM codes. The implemented coupling application supports two types of data transfer. The volume field transfer involves the exchange of numerical data representing physical quantities distributed across the computational domain. The boundary field transfer handles the interaction between physical domains at their interfaces.

Despite their promising role in demanding power plants, the turbulent behavior of liquid metals continues to pose significant challenges for both experimental and numerical investigations. Turbulence modeling in practical engineering systems is commonly addressed using the Reynolds-Averaged Navier-Stokes (RANS) approach, derived by applying a time-averaging operator to the Navier-Stokes equations. This averaging process introduces new turbulent unknowns, namely, the Reynolds stress tensor and the turbulent heat flux, resulting in a non-closed system of equations. Over the years, this turbulence closure problem has driven the development of various modeling techniques. Conventional first-order turbulence models rely on isotropic eddy viscosity and turbulent thermal diffusivity concepts. However, these approximations have been observed to be inadequate for liquid metals, which require more advanced modeling approaches. Among them, the Explicit Algebraic Stress Model (EASM) and the Explicit Algebraic Heat Flux Model (EAHFM) belong to a class of models between the first and second order. They are classified as anisotropic models, as they provide algebraic expressions for each component of the Reynolds stress tensor and turbulent heat flux.

The aim of this Thesis is to investigate both monolithic and coupling strategies for modeling and simulating relevant turbulence phenomena involving liquid metals, such as natural convection and heat transfer. As regard the turbulent natural convection, in the stand-alone code approach, an anisotropic turbulence model is implemented in the FEMuS code to account for the buoyancy effects of liquid metals' natural convection. The EASM and

EAHFM models are validated using a Differentially Heated Cavity (DHC) configuration, where the temperature difference between the cavity's side walls drives the natural convection motion. This configuration is also studied using the coupling code technique. Specifically, the volume data transfer algorithm is used to exchange velocity-related fields from OpenFOAM to FEMuS code and temperature fields from FEMuS to OpenFOAM. The goal is to integrate the OpenFOAM-validated dynamic solver capabilities with the more advanced thermal turbulence model in FEMuS, which is specific for liquid metals.

As part of the National Operational Program (PON) for Research and Innovation, this Ph.D. program features a collaborative project with Nier Ingegneria S.p.A.. The collaboration focuses on conducting CFD analysis of a Lead-Lithium heat exchanger designed by the company. This system is simulated using the implemented boundary data transfer algorithm within the coupling code framework. This approach simulates the Conjugate Heat Transfer (CHT) problem between the internal turbulent flow and the external tube. In this setup, the liquid metal is thermally coupled with a solid domain in an EUROFER pipe covered with copper fins. The anisotropic turbulence model of the FEMuS code simulates turbulent forced flow. At the same time, OpenFOAM models the temperature distribution in the solid regions. Interfaces between the solid and fluid domains are managed through the interface coupling application that ensures accurate interaction between the two regions.

This Thesis is organized as follows. In Chapter 1, the Reynolds-Averaged Navier-Stokes (RANS) system of equations is derived from the governing equations for incompressible laminar flow. Various strategies are presented to tackle the turbulence closure problem, alongside the most popular first-order turbulence models. Then, the Explicit Algebraic Models are derived to account for the buoyancy effects. Chapter 2 focuses on the description of the coupling code strategy. It details the numerical platform framework where the coupling application is implemented. Then, it describes the developed C++ modules that manage the integration between FEMuS and OpenFOAM, along with the numerical algorithm employed. Finally, the volume data transfer algorithm is validated using the Differentially Heated Cavity problem in the laminar natural convection regime. The boundary data transfer is validated through a Conjugate Heat Transfer problem. In Chapter 3, the turbulent natural convection problem is studied. Both mono-

lithic codes, FEMuS and OpenFOAM, and their respective turbulent models simulate the natural convection of liquid metals in a DHC configuration. The results are compared with literature references. Then, the volume data transfer method of the coupling application is employed to study the same DHC configuration, and the results are presented. Chapter 4 details the collaborative project conducted with Nier Ingegneria S.p.A. It describes the liquid metal-air heat exchanger and outlines the design constraints. Turbulent flow simulations are performed using both FEMuS and OpenFOAM codes, with results compared to benchmark data from the literature. Following this, the temperature distribution in the solid regions is computed using OpenFOAM. Finally, the boundary data transfer algorithm is employed to simulate the complete heat exchanger system.

Chapter 1

Turbulence Modeling for Liquid Metals

The knowledge of turbulence flow is important not only in many fields of engineering but more generally as the basis for many physical phenomena that we experience daily, from combustion processes to the motion of fluids within conduits, from the flow around moving vehicles to the behavior of blood in vessels. Even though it is a widely occurring phenomenon, turbulence is not completely understood. For this reason, and mainly because of its role in engineering applications, it is one of the most studied problems in the scientific community.

Over the last few decades, significant progress has been made in this field, driven by advancements in experimental technologies and, most importantly, the evolution of computational simulation techniques. In particular, Computational Fluid Dynamics (CFD) has enabled a quantitative and detailed analysis of turbulent flows by numerically solving the Navier-Stokes equations.

Many computational approaches for simulating turbulent flows are based on the Reynolds-averaged Navier-Stokes equations (RANS). This approach focuses on solving the Navier-Stokes equations by averaging statistical fluctuations and providing information on the mean properties of the flow. The time-averaging process results in a RANS system of equations with several

unknowns exceeding the number of equations, requiring the introduction of turbulent closure models.

Over the years, several turbulence models have been developed to address this closure problem, relying on the Boussinesq hypothesis and Reynolds' analogy. The Boussinesq hypothesis is commonly used in commercial codes because of its simple implementation, but it presents significant limitations, particularly when simulating complex and anisotropic flows. Moreover, most common approaches provide accurate results only for conventional fluids. The error introduced by standard models increases in non-conventional fluids, such as liquid metals, characterized by low-Prandtl number values. Therefore, more sophisticated models have been developed to account for the flow anisotropy and the peculiarity of turbulence in low-Prandtl number fluids.

This chapter focuses on deriving the system of equations implemented in FEMuS code for simulating the turbulent flow of low Prandtl number fluids. This chapter starts with a brief introduction to the governing equations for laminar flows and proceeds to the derivation of the RANS system. Strategies for addressing the closure problem for dynamical and thermal equations are presented. The Boussinesq hypothesis and the corresponding turbulent models are introduced. Then, more accurate anisotropic turbulence models are presented based on the Explicit Algebraic Stress Model (EASM) and the Explicit Algebraic Heat-Flux Model (EAHFM). These formulations have been derived in this chapter to account for the buoyancy effects occurring in turbulent natural convection regimes. Finally, after reviewing the relevant models from the literature, the full turbulence model implemented and employed for the simulations in this thesis is presented in a comprehensive summary.

1.1 Fluid Mechanics Equations

In fluid dynamics, the behavior of a fluid system, whether a gas or a liquid, can be effectively described by treating it as a continuous medium, even if it is composed of atoms and molecules. From a physical perspective, a continuous medium is a space filled with matter where every part can still be regarded as a continuum of matter. This assumption implies that the physical phenomena of interest in fluids can be analyzed on a spatial scale much larger than the distance between individual molecules. This approach allows a given volume of fluid to be represented as consisting of infinitesimal elements, small enough to provide a detailed system description but large enough to contain a

significant number of particles, typically on the order of Avogadro's number [12]. Under this assumption, the motion of the fluid is described as the movement of infinitesimal elements, each with a spatial extent $d\Omega$, which evolve along specific trajectories. At any given time t, a generic fluid element follows its trajectory and occupies a unique space position (x, y, z). The infinitesimal elements' trajectories cannot overlap in space, nor can a single element occupy the same position at different times. Thus, the vector field $\mathbf{u}(x, y, z, t)$ can be defined by describing the velocity of a generic infinitesimal fluid element. The characterization of the thermodynamic state of the fluid requires the additional assumption of *local equilibrium*, which states that each element of the fluid can be considered a thermodynamic system in stable equilibrium at any t. Given this assumption, it is possible to define the scalar fields of temperature $T(x, y, z, t)$, pressure $P(x, y, z, t)$, and density $\rho(x, y, z, t)$.

1.1.1 Conservation Equation

At a fixed time t, consider a generic fluid system or a volume of material $\Omega(t)$, representing a region of space that contains a portion of the fluid mass. This volume is bounded by a closed regular surface $\partial\Omega(t)$, and moves and deforms with the fluid over time. According to the Lagrangian approach, $\Omega(t)$ contains the same fluid particles throughout its time evolution. On the other hand, in the Eulerian approach, a chosen arbitrarily control volume Ω_0 is considered. This control volume represents a fixed region where the liquid particles contained in the control volume can vary over time. The relationship between the two approaches is provided by Reynolds' transport theorem. Therefore, defined a generic extensive property Ψ in $\Omega(t)$ and its corresponding intensive property ψ as

$$\Psi(t) = \int_{\Omega(t)} \rho(\mathbf{x}, t)\psi(\mathbf{x}, t)d\Omega, \tag{1.1}$$

where $\mathbf{x} = (x, y, z)$, the Reynolds' transport theorem is expressed as

$$\frac{d\Psi}{dt} = \int_{\Omega_0} \frac{\partial}{\partial t}(\rho\psi)d\Omega_0 + \int_{\partial\Omega_0} (\rho\psi\mathbf{u}) \cdot \mathbf{n}d\partial\Omega_0. \tag{1.2}$$

The integral conservation equation for ψ states that the extensive property change rate, throughout the volume, equals the net flux balance through

the outer surface $\partial\Omega_0$, plus the net contribution of volumetric generation. Therefore, we obtain

$$\frac{d\Psi}{dt} = -\int_{\partial\Omega_0} \mathbf{J}_\psi \cdot \mathbf{n}\, d\partial\Omega_0 + \int_{\Omega_0} \rho S\, d\Omega_0, \tag{1.3}$$

where $\mathbf{J}_\psi$ is the flux density, and S the generation term per unit of mass and time. Combining equation (1.2) with equation (1.3), we obtain the integral form of the general conservation equation

$$\int_{\Omega_0} \frac{\partial(\rho\psi)}{\partial t}\, d\Omega_0 + \int_{\Omega_0} \nabla\cdot(\rho\psi\mathbf{u})\, d\Omega_0 = -\int_{\Omega_0} \nabla\cdot\mathbf{J}_\psi\, d\Omega_0 + \int_{\Omega_0} \rho S\, d\Omega_0. \tag{1.4}$$

Since the control volume is chosen arbitrarily, equation (1.4) must also hold for any Ω_0, and thus, its differential form must also be valid. Therefore, we can obtain the following expressions for the conservation equation

$$\frac{\partial(\rho\psi)}{\partial t} + \nabla\cdot(\rho\psi\mathbf{u}) = -\nabla\cdot\mathbf{J}_\psi + \rho S, \tag{1.5}$$

or its representation using the Einstein summation notation

$$\frac{\partial}{\partial t}(\rho\psi) + \frac{\partial}{\partial x_i}(\rho\psi u_i) = -\frac{\partial}{\partial x_i} J_{\psi_i} + \rho S. \tag{1.6}$$

The conservation equation expressed in the differential form (1.6) can be used by substituting the extensive property Ψ with mass, momentum, or energy to derive their respective conservation equations.

Mass Conservation Equation

The Eulerian differential form of the mass conservation equation, or continuity equation, can be easily deduced by substituting mass as the extensive property Ψ. Since the production terms, both volumetric and surface, are null, the balance equation (1.6) becomes

$$\frac{\partial\rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i) = 0, \tag{1.7}$$

or in the following form

$$\frac{\partial\rho}{\partial t} + u_i\frac{\partial\rho}{\partial x_i} + \rho\frac{\partial u_i}{\partial x_i} = 0\,. \tag{1.8}$$

The sum of the first two terms represents the substantial or convective derivative of the density. This definition leads to the general form of the mass conservation equation as

$$\frac{D\rho}{Dt} + \rho\frac{\partial u_i}{\partial x_i} = 0. \tag{1.9}$$

In the case of a constant fluid density equation (1.9) simplifies to

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{1.10}$$

meaning that for an incompressible fluid, the velocity field is solenoidal.

Momentum Conservation Equation

We can formulate the momentum conservation equation by imposing $\Psi = \rho\mathbf{u}$, where ρ is the density of the volume of material, and $\mathbf{u}$ is the velocity vector. Therefore, the corresponding specific quantity is the velocity $\psi = \mathbf{u}$, representing the momentum per unit mass. The momentum conservation principle is deduced from Newton's second law of dynamics as applied to fluid systems. It states that the substantial derivative of the momentum associated with the material volume $\Omega(t)$ equals the net external forces acting on the volume. The total external forces include body forces (e.g., gravity) and surface forces. Contact forces act on the external surface of the infinitesimal fluid element, while body forces act throughout the entire fluid volume. The former are classified as pressure and viscous forces, expressed as follows

$$J_{ij} = -\sigma_{ij} = p\delta_{ij} - \tau_{ij}, \tag{1.11}$$

where σ_{ij} is the stress tensor, p is the pressure of the system, δ_{ij} is the Kronecker delta, and τ_{ij} is the viscous stress tensor. The first contribution is the isotropic part of the stress tensor σ_{ij}, while the latter represents its deviatoric part.

Body forces include those arising, for example, from gravitational and electromagnetic fields. Neglecting all the fields except for the gravitational one and substituting into equation (1.6), we formulate the Navier-Stokes equation as

$$\frac{\partial(\rho u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_i u_j) = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho g_i. \tag{1.12}$$

From the definition of the substantial derivative, the conservation equation becomes as follows

$$\frac{D(\rho u_i)}{Dt} + \rho\frac{\partial u_j}{\partial x_j}u_i = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho g_i. \tag{1.13}$$

This equation is valid for a continuous system, but describing a specific system requires introducing a constitutive relationship. Given a Newtonian viscous fluid, which follows Newton's experimental law on viscosity, it can be demonstrated that the stress tensor is related to the fluid deformation rate by the following relation

$$\tau_{ij} = \mu S_{ij} - \left(\frac{2}{3}\mu - \lambda\right) tr(\mathbf{S})\delta_{ij}, \tag{1.14}$$

where μ and λ are the dynamic and bulk viscosities, respectively, S_{ij} is the strain rate tensor defined as

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{1.15}$$

and $tr(\mathbf{S})$ is the trace of the strain rate tensor

$$tr(\mathbf{S}) = S_{ii}. \tag{1.16}$$

We can rewrite the Navier-Stokes equation considering the (1.14) as follows

$$\frac{D(\rho u_i)}{Dt} + \rho\frac{\partial u_j}{\partial x_j}u_i = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\mu S_{ij} - \left(\frac{2}{3}\mu - \lambda\right) tr(\mathbf{S})\delta_{ij}\right] + \rho g_i. \tag{1.17}$$

This expression can be simplified under incompressible fluid conditions, where equation (1.10) holds

$$\frac{Du_i}{Dt} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}(\nu S_{ij}) + g_i, \tag{1.18}$$

where $\nu = \frac{\mu}{\rho}$ represents the kinematic viscosity.

Energy Conservation Equation

For the energy conservation equation, the extensive property is the fluid total energy E. The corresponding specific property consists of two contributions: the kinetic energy of the fluid element, $u^2/2$, and its internal energy, ϵ. Thus, the specific energy υ is defined as

$$\upsilon = \epsilon + \frac{u^2}{2}. \tag{1.19}$$

The source terms of the energy conservation equation consider the external forces acting on the surface of the control volume and the heat introduced

or generated within the system. The former accounts for the work of surface forces, $\sigma_{ij}u_j$, and the heat flux through the external surface, q_i. Therefore, J_ψ can be expressed as

$$J_i = q_i - \sigma_{ij}u_j. \tag{1.20}$$

The volumetric contribution consists of terms such as the heat content per unit of time due to gravitational forces and the internal heat generation Q. The latter term accounts for any generated power, such as chemical reactions or radiation absorption. We can write the volumetric power generation as

$$S = \frac{Q}{\rho} + g_i u_i. \tag{1.21}$$

By substituting these quantities into equation (1.10), the energy equation can be derived as follows

$$\frac{D}{Dt}(\rho\upsilon) + \rho\upsilon\frac{\partial u_i}{\partial x_i} = -\frac{\partial q_i}{\partial x_i} - \frac{\partial}{\partial x_i}(pu_i - \tau_{ij}u_j) + \rho g_i u_i + Q. \tag{1.22}$$

The equation (1.22) can be rearranged and simplified to obtain the following internal energy conservation equation

$$\rho\frac{D\epsilon}{Dt} = -\frac{\partial q_i}{\partial x_i} + \tau_{ij}\frac{\partial u_j}{\partial x_i} + Q. \tag{1.23}$$

The second term of the right-hand side of this equation represents the viscous forces of the fluid element deformations, which are inherently non-negative. As a result, in any flow field with nonzero deformation rates, the heat generated due to viscous stresses leads to an irreversible increase in the internal energy. This term is called viscous dissipation function and is named Φ.

To write the energy conservation equation, a constitutive equation is introduced to relate the heat transport term, q_i, to the temperature field, T. For a thermally conductive fluid where heat transfer occurs via molecular conduction, this relationship is described by Fourier law

$$q_i = -k(T)\frac{\partial T}{\partial x_i}, \tag{1.24}$$

where $k(T)$ represents the thermal conductivity. In addition, a conservation equation for the variable T can be obtained by reformulating the first term of the equation (1.23). The internal energy ϵ, under the incompressible fluid hypothesis, depends only on temperature, and the following differential relationship can be written

$$d\epsilon = c(T)dT, \tag{1.25}$$

where $c(T)$ is the fluid specific heat. Substituting the relationships (1.24) and (1.25) the energy conservation equation for the temperature field becomes

$$\rho c \frac{DT}{Dt} = \frac{\partial}{\partial x_i}\left(k\frac{\partial T}{\partial x_i}\right) + \Phi + Q, \tag{1.26}$$

or

$$\frac{DT}{Dt} = \frac{\partial}{\partial x_i}\left(\alpha\frac{\partial T}{\partial x_i}\right) + \frac{\Phi}{\rho c} + \frac{Q}{\rho c}, \tag{1.27}$$

where α is the thermal diffusivity.

Summary of the Governing Laminar Equations

To conclude, in absence of additional simplifying assumptions, the full compressible system of governing equations is

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_i}(\rho u_i) = 0, \tag{1.28}$$

$$\frac{\partial (\rho u_i)}{\partial t} + \frac{\partial}{\partial x_j}(\rho u_i u_j) = -\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ij}}{\partial x_j} + \rho g_i, \tag{1.29}$$

$$\frac{\partial}{\partial t}\left(\rho \upsilon u_i\right) + \frac{\partial}{\partial x_i}\left(\rho \upsilon\right) = -\frac{\partial q_i}{\partial x_i} - \frac{\partial}{\partial x_i}(p u_i - \tau_{ij} u_j) + \rho g_i u_i + Q. \tag{1.30}$$

Based on the considerations of previous sections and the simplifications derived for the case of incompressible fluid, the system of governing equations is defined as follows

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{1.31}$$

$$\frac{Du_i}{Dt} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}(\nu S_{ij}) + g_i, \tag{1.32}$$

$$\frac{DT}{Dt} = \frac{\partial}{\partial x_i}\left(\alpha\frac{\partial T}{\partial x_i}\right) + \frac{\Phi}{\rho c} + \frac{Q}{\rho c}. \tag{1.33}$$

Oberbeck-Boussinesq Approximation

The system of equations (1.28)-(1.30), written for the generic compressible fluid problem, has several unknowns exceeding the number of equations. If we consider non-isothermal flows, the Oberbeck-Boussinesq approximation provides a solution for simplifying the equations and avoiding the need to solve for the complete system. According to this hypothesis, the only variations

in density are those that influence buoyancy forces. Consequently, density can be treated as constant ($\rho = \rho_0 = \text{const}$), except for the gravitational force term in the momentum conservation equation. We can reformulate the system of equations (1.28)-(1.30) as

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{1.34}$$

$$\rho_0 \frac{Du_i}{Dt} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}(\mu S_{ij}) + \rho g_i, \tag{1.35}$$

$$\rho_0 c \frac{DT}{Dt} = \frac{\partial}{\partial x_i}\left(k \frac{\partial T}{\partial x_i}\right) + \Phi + Q. \tag{1.36}$$

The approximation remains valid as long as the temperature undergoes variations smaller than 10–20 K. If this holds, we can consider a reference temperature T_0 and expand ρ in a Taylor series as

$$\rho = \rho(T) = \rho(T_0) + \left.\frac{d\rho}{dT}\right|_{T_0} (T - T_0) + O[(T - T_0)^2], \tag{1.37}$$

where, considering small temperature variations, the second-order term can be neglected. We now introduce the coefficient of isobaric thermal expansion, β, as

$$\beta = -\frac{1}{\rho_0}\left(\frac{\partial \rho}{\partial T}\right)_p, \tag{1.38}$$

and its value at the reference temperature

$$\beta = -\frac{1}{\rho}\left.\frac{d\rho}{dT}\right|_{T_0}, \tag{1.39}$$

where $\rho(T_0) = \rho_0$. Therefore, we can formulate the temperature dependence of the density using the thermal expansion coefficient as

$$\rho(T) = \rho_0 - \rho_0 \beta (T - T_0). \tag{1.40}$$

This expression leads the momentum conservation equation to be rewritten as

$$\rho_0 \frac{Du_i}{Dt} = -\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}(\mu S_{ij}) + g_i \left[\rho_0 - \rho_0 \beta (T - T_0)\right]. \tag{1.41}$$

Finally, defined the piezometric pressure as $P = p - \rho_0 g_i x_i$, equation (1.41) becomes

$$\rho_0 \frac{Du_i}{Dt} = -\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}(\mu S_{ij}) + \rho_0 g_i \beta (T - T_0), \tag{1.42}$$

where $\rho_0 g_i \beta(T - T_0)$ represents the buoyancy force. Given these assumptions and the constitutive equations introduced earlier, the system of equations under the Oberbeck-Boussinesq approximation is

$$\frac{\partial u_i}{\partial x_i} = 0, \tag{1.43}$$

$$\frac{Du_i}{Dt} = -\frac{1}{\rho}\frac{\partial P}{\partial x_i} + \frac{\partial}{\partial x_j}(\nu S_{ij}) + g_i \beta(T - T_0), \tag{1.44}$$

$$\frac{DT}{Dt} = \frac{\partial}{\partial x_i}\left(\alpha \frac{\partial T}{\partial x_i}\right), \tag{1.45}$$

where the viscous energy dissipation and the internal heat generation have been neglected.

1.2 Governing Equations for Turbulent Flows

The study of turbulence dates back centuries. Interest in this field arose at the end of the 19th century when Osborn Reynolds experimentally investigated manifestations of this phenomenon. In particular, he noted the formation of visible perturbations in the fluid motion as the velocity increased, and he named this type of flow a *turbulent* motion. Its behavior is characterized by the dimensionless Reynolds number

$$Re = \frac{UL}{\nu}, \tag{1.46}$$

where U is the fluid's velocity, ν is its kinematic viscosity, and L is the hydraulic diameter of the conduit. It has been observed that the motion turns to turbulent behavior for Reynolds numbers exceeding $2000 - 3000$. For lower values, it remains laminar.

Turbulence arises from instabilities in an initially laminar, often two-dimensional flow, which evolves into complex three-dimensional structures like vortices. As the Reynolds number increases, these disturbances intensify, leading to fully developed turbulence characterized by random fluctuations in velocity and temperature over space and time. Turbulence arises from small variations in initial conditions, boundaries, and material properties, leading to random behavior that limits purely deterministic analysis. Therefore, studying turbulence requires statistical methods, which will be briefly introduced in this chapter.

Over the years, several theories have been developed to rigorously explain turbulence. Turbulence has been interpreted as a superposition of vortices (eddies), spanning various scales and sizes. According to Richardson and Kolmogorov's energy cascade theory [13, 14], kinetic energy is transferred from large to smaller eddies through non-linear interactions. The instability of vortices caused by intense non-linear effects leads to their breakdown into increasingly smaller structures, each characterized by higher local Reynolds numbers, thereby sustaining the cascade. This progression persists until the structures reach scales where the Reynolds number is reduced enough. At these smaller scales, molecular diffusion effects become dominant, and kinetic energy is dissipated into heat through the action of viscosity.

The largest turbulent structures, known as macro-scale, have dimensions determined by the geometric properties of the phenomenon. For these structures, a characteristic length scale L and velocity U are defined. From these, a time scale $T = L/U$ and a Reynolds number $Re = UL/\nu$ can be determined. On the other hand, the scales responsible for dissipating turbulent energy are known as Kolmogorov scales or microscales [15]. Their characteristic length is represented by η, the characteristic time by τ, and the velocity by v. These quantities are defined as follows

$$\eta = \left(\frac{\nu^3}{\varepsilon}\right)^{\frac{1}{4}}, \quad \tau = \left(\frac{\nu}{\varepsilon}\right)^{\frac{1}{2}}, \quad v = (\nu\varepsilon)^{\frac{1}{4}}, \tag{1.47}$$

where ε is the viscous dissipation rate at Kolmogorov scales, representing the turbulent kinetic energy, k, dissipated per unit mass and time. Under stationary conditions, the viscous dissipation rate is expressed as the ratio of the kinetic energy per unit mass at macroscopic scales, E_c, and T as follows

$$\varepsilon \sim \frac{E_c}{T} = \frac{U^3}{2L}. \tag{1.48}$$

From expressions (1.47), a Reynolds number associated with the small scales can be defined as

$$Re_\eta = \frac{\eta v}{\nu}, \tag{1.49}$$

which can be demonstrated through straightforward steps to be equal to unity. The theory of the energy cascade can be explained by introducing a time scale associated with viscous diffusion, denoted by T_v. This time scale is derived from the one-dimensional diffusion equation as

$$\frac{\partial u}{\partial t} = \nu\frac{\partial^2 u}{\partial x^2}, \quad \Rightarrow \quad \frac{U}{T_v} \sim \frac{\nu U}{L^2}, \tag{1.50}$$

that allows to define the time scale, T_v, as

$$T_v \sim \frac{L^2}{\nu} = ReT. \tag{1.51}$$

According to (1.51), at high Reynolds numbers in turbulent flows, the viscous diffusion time scale is much larger than the characteristic time scale of the mean flow. This condition leads to reduced energy dissipation in large-scale motions and explains the transfer of kinetic energy from large scales to small scales. At the Kolmogorov scale, instead, the viscous diffusion becomes relevant and it lastly dissipates the energy. These effect rises when the time scale, T_v, is of the same order of magnitude of the characteristic time τ and at microscales the Reynolds number is unitary, indeed, we have

$$T_v \sim Re\tau, \quad \Rightarrow \quad T_v \sim \tau\,. \tag{1.52}$$

From equations (1.47) and (1.48), the ratio between the microscales and macroscales can be derived as

$$\frac{\eta}{L} = \left(\frac{\nu}{UL}\right)^{\frac{3}{4}} = Re^{-\frac{3}{4}}\,, \tag{1.53}$$

$$\frac{\tau}{T} = \left(\frac{\nu}{UL}\right)^{\frac{1}{2}} = Re^{-\frac{1}{2}}\,, \tag{1.54}$$

$$\frac{v}{U} = \left(\frac{\nu}{UL}\right)^{\frac{1}{4}} = Re^{-\frac{1}{4}}\,. \tag{1.55}$$

As Re increases, the difference in orders of magnitude between microscales and macroscales becomes more important. Based on these considerations, simulating the behavior of turbulent flows across all scales would require a mesh with elements as small as η and a timestep of the order of τ. This method, known as Direct Numerical Simulation (DNS), solves the Navier-Stokes equations for turbulent flows considering the whole range of spatial and temporal scales without any simplifying assumptions. Therefore, the DNS approach would require a grid with a number of elements equal to $N = (L/\eta)^3 = Re^{\frac{9}{4}}$. As the Reynolds number increases, the number of grid elements grows exponentially, leading to high computational demands and very long total computation times.

Since DNS's computational cost is extremely high, even at low Reynolds numbers, alternative modeling approaches are typically employed to describe turbulence. An example of an approximate mathematical model is the Large

Eddy Simulation (LES). This approach applies a filtering operation to directly simulate the larger vortex structures while modeling the smaller ones. As a result, LES requires less grid refinement and has a lower computational cost than DNS. The third approach is based on the Reynolds-averaged Navier-Stokes method. This statistical approach focuses on simulating only the mean motion fields, significantly reducing the required number of grid elements and, consequently, the computational cost.

1.2.1 Derivation of Reynolds-Averaged Navier-Stokes and Energy Equations

Reynolds introduced the Reynolds Averaged Navier-Stokes equations approach in 1895 [16]. This method does not resolve the detailed behavior at the smallest scales and assumes that turbulence is a purely statistical phenomenon across all scales. In his theory, he introduced three different types of averaging, which can be applied to the turbulent flow characteristics. Whether the turbulent flow is statistically stationary, homogeneous, or periodic (meaning that it can be replicated N times), the three averaging methods are the time average, spatial average, and ensemble average, respectively. In this discussion, we assume statistical stationarity and homogeneity of the flow, under which assumption all these averaging methods are equivalent. Therefore, the following analysis is based on the application of the temporal average, defined as

$$\langle \psi(\mathbf{x}, t) \rangle = \frac{1}{T} \int_{t}^{t+T} \psi(\mathbf{x}, t) dt \,, \tag{1.56}$$

where $\psi(\mathbf{x}, t)$ is a random field and T is the time scale. This averaging operator assumes that the quantity $\psi(\mathbf{x}, t)$ may vary over time with fluctuations occurring on a scale much larger than that of the averaging operator. According to the Reynolds decomposition, the instantaneous and fluctuating velocity vector field, $\mathbf{u}(\mathbf{x}, t)$, is composed by the sum of its mean value $\langle \mathbf{u}(\mathbf{x}, t) \rangle$ and a perturbation or turbulent fluctuation, $\mathbf{u}'(\mathbf{x}, t)$, as follows

$$\mathbf{u}(\mathbf{x}, t) = \langle \mathbf{u}(\mathbf{x}, t) \rangle + \mathbf{u}'(\mathbf{x}, t) \,. \tag{1.57}$$

Similarly, the pressure field and the temperature field can be defined as

$$p(\mathbf{x}, t) = \langle p(\mathbf{x}, t) \rangle + p'(\mathbf{x}, t) \,, \tag{1.58}$$

$$T(\mathbf{x},t) = \langle T(\mathbf{x},t)\rangle + T'(\mathbf{x},t)\,. \tag{1.59}$$

Thus, we can obtain the following system of equations known as the Reynolds Averaged Navier-Stokes equations

$$\frac{\partial\langle u_i\rangle}{\partial x_i} = 0\,, \tag{1.60}$$

$$\frac{D\langle u_i\rangle}{Dt} = -\frac{1}{\rho}\frac{\partial\langle p\rangle}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial\langle u_i\rangle}{\partial x_j} + \frac{\partial\langle u_j\rangle}{\partial x_i}\right) - \langle u_i' u_j'\rangle\right] - g_i\beta\langle T\rangle\,, \tag{1.61}$$

$$\frac{D\langle T\rangle}{Dt} = \frac{\partial}{\partial x_i}\left(\alpha\frac{\partial\langle T\rangle}{\partial x_i} - \langle u_i' T'\rangle\right)\,. \tag{1.62}$$

The term $\langle u_i' u_j'\rangle$ is introduced as a result of the averaging operator applied to the convective term in the Navier-Stokes equation. This contribution acts as a source term in the mean velocity evolution equation, and it can be read as a mean momentum flux due to the fluctuating velocity. This contribution can be defined as a stress tensor added to the viscous stress tensor, τ_{ij}. Thus, it is commonly indicated as $\tau_{ij}^r = -\rho\langle u_i' u_j'\rangle$ and it is known as the Reynolds Stress Tensor. Therefore, the equation can be rewritten also in the following form

$$\rho\frac{D\langle u_i\rangle}{Dt} = -\frac{\partial\langle p\rangle}{\partial x_i} + \rho\frac{\partial\tau_{ij}^{eff}}{\partial x_j} - \rho g_i\beta\langle T\rangle\,, \tag{1.63}$$

where $\tau_{ij}^{eff} = \tau_{ij} + \tau_{ij}^r$ is the effective stress tensor.

On the other hand, the term $\langle u_i' T'\rangle$ in equation (1.62) is the velocity-temperature variance and it represents the flux of the temperature due to the fluctuating velocity field. This contribution is called Turbulent Heat Flux and it can be named as $q_i^r = \rho c_p\langle u_i' T'\rangle$. Thus, equation (1.62) can be formulated as

$$\rho c\frac{D\langle T\rangle}{Dt} = -\frac{\partial q_i^{eff}}{\partial x_i} + Q\,, \tag{1.64}$$

where $q_i^{eff} = q_i + q_i^r$ is the effective heat flux.

1.2.2 Law of the Wall

We now introduce some empirical relations, known as the *law of the wall*, which allow to evaluate the wall behavior of the mean velocity and mean temperature fields. To define these relations we introduce the dimensionless velocity field as

$$u^+ = \frac{\langle u\rangle}{u_\tau}\,, \tag{1.65}$$

where u_τ is the friction velocity. This reference velocity is defined as

$$u_\tau = \sqrt{\frac{\tau_w}{\rho}}\,, \tag{1.66}$$

where τ_w is the viscous stress contribution at the wall. The law of the wall can be generically expressed as

$$u^+ = f(y^+)\,, \tag{1.67}$$

where $y^+ = u_\tau d/\nu$ is the dimensionless distance from the wall. We can distinguish three distinct regions in the the boundary layer, where different laws of the wall hold. Close to the wall, within the region at $y^+ < 5$, the following expression is used

$$u^+ = y^+\,. \tag{1.68}$$

Therefore, the region where equation (1.68) holds is called the *viscous sublayer* or *linear region*. Far from wall, at $y^+ > 50$, the following expression is valid

$$u^+ = \frac{1}{\kappa}\Big[\ln(y^+) + A\Big]\,, \tag{1.69}$$

where A is a constant usually equal to 5.2 and κ is the Von Kármán constant. Equation (1.69) describes the mean velocity in a region known as the *logarithmic region*. Between the linear and logarithmic regions, there is an intermediate zone known as the *buffer layer*, where the velocity profile transitions between the two. The dimensionless velocity can thus be written as

$$u^+ = \begin{cases} y^+ & \text{if } y^+ < 5 \\ \dfrac{1}{\kappa}\Big[\ln(y^+) + A\Big] & \text{if } y^+ > 50. \end{cases} \tag{1.70}$$

Similar correlations can be derived to describe the temperature profile in the two regions of the boundary layer. We define the dimensionless temperature T^+ as

$$T^+ = \frac{(T_w - \langle T \rangle)}{T_\tau}\,, \tag{1.71}$$

where T_w is the wall temperature and T_τ is called the friction temperature and is defined as

$$T_\tau = \frac{q_w}{u_\tau \rho c_p}\,. \tag{1.72}$$

Here, q_w is the wall heat flux. In the linear region, we can introduce the following law of the wall

$$T^+ = Pr y^+\,, \tag{1.73}$$

while the behavior in the logarithmic region can be expressed by the following function

$$T^+ = \frac{1}{\kappa}\ln(y^+) + C\,, \tag{1.74}$$

where the constant C depends on the Prandtl number and the coefficient κ.

1.2.3 The Closure Problem

Due to the introduction of the Reynolds stress tensor and the turbulent heat flux through the averaging process, the system of equations for the mean field (1.60) - (1.62) differs from the original system of equations (1.43) - (1.45). This difference is not negligible, and unlike the original system, the RANS problem is no longer closed. In this system, the tensor $\langle u_i' u_j' \rangle$ and the vector $\langle u_i' T' \rangle$ increase the number of unknowns. In particular, the Reynolds stress tensor for a three-dimensional problem is obtained from the dyadic product of the velocity fluctuations with themselves and is equal to

$$\langle u_i' u_j' \rangle = \begin{bmatrix} \langle u'^2 \rangle & \langle u'v' \rangle & \langle u'w' \rangle \\ \langle v'u' \rangle & \langle v'^2 \rangle & \langle v'w' \rangle \\ \langle w'u' \rangle & \langle w'v' \rangle & \langle w'^2 \rangle \end{bmatrix}. \tag{1.75}$$

The diagonal components, $\langle u_i' u_i' \rangle$, are referred to as normal stresses, while the off-diagonal terms are known as shear stresses. The Reynolds stress tensor is symmetric, thus $\langle u_i' u_j' \rangle = \langle u_j' u_i' \rangle$, and it introduces six additional unknowns to the system of equations (1.60) - (1.62). The turbulent heat flux, instead, is defined as

$$\langle u_i' T' \rangle = (\langle u'T' \rangle, \langle v'T' \rangle, \langle w'T' \rangle)\,, \tag{1.76}$$

and it adds three more unknowns. Overall, the system consists of only five equations and fourteen unknowns, including the pressure field, the three components of the velocity field, the temperature field, the six components of the Reynolds stress tensor, and the three components of the turbulent heat flux. Consequently, without additional information to determine the extra statistical terms, the RANS equations cannot be solved.

Over the years, several strategies have been developed to address the closure problem of turbulence. These strategies can be divided into two main categories. The first one consists of introducing first-order models that rely on the concept of eddy viscosity, referred to as eddy viscosity models, and

eddy thermal diffusivity, known as eddy thermal diffusivity models. The second category includes second-order models known as nonlinear eddy viscosity/diffusivity models or anisotropic models, which add to the RANS system transport equations for each component of the Reynolds stress tensor and the turbulent heat flux.

1.3 Dynamic Turbulence Modeling

The purpose of eddy viscosity models is to introduce a closure expression that computes the Reynolds stress tensor, without solving six separate equations for the six tensor unknowns.

The first approach was introduced by Boussinesq in 1877 [17] and refers to the concept of *turbulent viscosity*. This hypothesis asserts that turbulence produces effects similar to molecular diffusion, and the Reynolds stress tensor is mathematically equivalent to that of viscous stresses. As a result, the eddy viscosity hypothesis assumes a form similar to the constitutive relation between stress and rate of strain introduced for a Newtonian fluid. The equation (1.61) can be reformulated as

$$\frac{D\langle u_i\rangle}{Dt} = -\frac{1}{\rho}\frac{\partial\langle p\rangle}{\partial x_i} + \frac{\partial}{\partial x_j}\left[(\nu+\nu_t)\left(\frac{\partial\langle u_i\rangle}{\partial x_j}+\frac{\partial\langle u_j\rangle}{\partial x_i}\right)\right] - g_i\beta\langle T\rangle\,, \tag{1.77}$$

where ν_t is the turbulent viscosity. The two contributions on the viscosity diffusion are usually known as $\nu_{eff} = \nu + \nu_t$, which is the effective viscosity.

By using Equation (1.77), the closure problem is significantly simplified, reducing the six unknown components of the tensor to a single scalar value, ν_t. Unlike molecular viscosity, ν_t also depends on the flow's state of motion, and in order to solve the closure problem, turbulent models must be introduced. Eddy viscosity models are classified according to the type of equations they introduce into the system. Models that use algebraic equations are known as zero-equation models, while those that add N differential equations to close the problem are referred to as N-equation models.

1.3.1 Zero-Equation Model: Mixing Length

In zero-equation models, turbulent viscosity is specified algebraically. Among the algebraic models, the mixing length model introduced by Ludwig Prandtl in 1925 [18] is worth mentioning. This simple model defines the turbulent

viscosity as a function of the mixing length ℓ_m as

$$\nu_t = \ell_m^2 \frac{d\langle u \rangle}{dy}. \tag{1.78}$$

As shown by (1.78), to become fully determined, this model requires the computation of ℓ_m value. The formula used for computing the mixing length depends on the distance from the wall; in particular, two different relationships have been introduced according to the region of the boundary layer considered. The first one is valid near the wall, in the linear region, and the second is defined within the logarithmic region. Thus, close to the wall the following expression holds

$$\ell_m = \kappa y, \tag{1.79}$$

where κ is the Von Kármán constant. For the logarithmic region, in 1956, Van Driest proposed the following empirical relationship [19]

$$\ell_m = \kappa y \left[1 - \exp\left(-\frac{y u_\tau}{\nu A}\right)\right], \tag{1.80}$$

where $A = 26$ is called the Van Driest constant.

The main limitation of this model is its specificity, as it is explicitly designed for wall flows and is therefore not applicable to other types of flows or to real-world scenarios of interest.

1.3.2 One-Equation Model

The one-equation model, or k-model, assumes that turbulent viscosity is a function of the turbulent kinetic energy $k(\mathbf{x}, t)$. This turbulent variable is defined to be half the trace of the Reynolds stress tensor

$$k = \frac{1}{2}\left(\langle u'^2 \rangle + \langle v'^2 \rangle + \langle w'^2 \rangle\right) = \frac{1}{2}\langle u_i' u_i' \rangle. \tag{1.81}$$

Therefore, in order to formulate the transport equation for the mean turbulent, kinetic energy is necessary to first derive an equation for $\langle u_i' u_j' \rangle$. Given the Reynolds-Averaged Navier-Stokes equation (1.61) and the momentum equation (1.44) for the instantaneous velocity $\mathbf{u}$, we subtract (1.61) from (1.44) and we obtain a transport equation for the fluctuating velocity field

$$\begin{aligned} \frac{\partial u_i'}{\partial t} + \langle u_j \rangle \frac{\partial u_i'}{\partial x_j} + u_j' \frac{\partial \langle u_i \rangle}{\partial x_j} + u_j' \frac{\partial u_i'}{\partial x_j} = -\frac{1}{\rho}\frac{\partial p'}{\partial x_i} + \\ + \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial u_i'}{\partial x_j} + \frac{\partial u_j'}{\partial x_i}\right) + \langle u_i' u_j' \rangle\right] - g_i \beta T'. \end{aligned} \tag{1.82}$$

By multiplying the i-th equation of (1.82) by u'_j and the j-th equation of (1.82) by u'_i and time-averaging, we can formulate the transport equation for $\langle u'_i u'_j \rangle$ as the sum of the two resulting equations

$$\begin{aligned}\frac{\partial \langle u'_i u'_j \rangle}{\partial t} + \langle u_k \rangle \frac{\partial \langle u'_i u'_j \rangle}{\partial x_k} + \langle u'_i u'_k \rangle \frac{\partial \langle u_j \rangle}{\partial x_k} + \langle u'_j u'_k \rangle \frac{\partial \langle u_i \rangle}{\partial x_k} + \\ + \frac{\partial}{\partial x_k} \langle u'_i u'_j u'_k \rangle = -\frac{1}{\rho} \left(\langle u'_i \frac{\partial p'}{\partial x_j} \rangle + \langle u'_j \frac{\partial p'}{\partial x_i} \rangle \right) + \\ + \nu \left(\langle u'_j \frac{\partial^2 u'_i}{\partial x_k^2} \rangle + \langle u'_i \frac{\partial^2 u'_j}{\partial x_k^2} \rangle \right) + \langle u'_j \frac{\partial u'_k u'_i}{\partial x_k} \rangle + \\ + \langle u'_i \frac{\partial u'_k u'_j}{\partial x_k} \rangle - \beta \left(g_j \langle u'_i T' \rangle + g_i \langle u'_j T' \rangle \right).\end{aligned} \tag{1.83}$$

The first two terms on the right-hand side of Equation (1.83) represent the interaction between the pressure gradient and turbulent fluctuations, and the viscous dissipation of these fluctuations, respectively. By reformulating these terms, we obtain the following equation for the Reynolds stress tensor

$$\begin{aligned}\frac{D \langle u'_i u'_j \rangle}{Dt} = -\langle u'_i u'_k \rangle \frac{\partial \langle u_j \rangle}{\partial x_k} - \langle u'_j u'_k \rangle \frac{\partial \langle u_i \rangle}{\partial x_k} - 2\nu \langle \frac{\partial u'_i}{\partial x_k} \frac{\partial u'_j}{\partial x_k} \rangle + \\ + \frac{1}{\rho} \left(\langle p' \frac{\partial u'_j}{\partial x_i} \rangle + \langle p' \frac{\partial u'_i}{\partial x_j} \rangle \right) + \frac{\partial}{\partial x_k} \left(\nu \frac{\partial \langle u'_i u'_j \rangle}{\partial x_k} - \langle u'_i u'_j u'_k \rangle + \right. \\ \left. - \langle \frac{p'}{\rho} (\delta_{ki} u'_j + \delta_{kj} u'_i) \rangle \right) - \beta \left(g_j \langle u'_i T' \rangle + g_i \langle u'_j T' \rangle \right),\end{aligned} \tag{1.84}$$

Given the equation (1.84), if we compute its trace and we divide it by 2, we can derive the transport equation for the turbulent kinetic energy as

$$\begin{aligned}\frac{Dk}{Dt} = -\langle u'_i u'_k \rangle \frac{\partial \langle u_i \rangle}{\partial x_k} - \nu \langle \frac{\partial u'_i}{\partial x_k} \frac{\partial u'_i}{\partial x_k} \rangle + \frac{\partial}{\partial x_k} \left(\nu \frac{\partial k}{\partial x_k} + \right. \\ \left. - \frac{1}{2} \langle u'_i u'_i u'_k \rangle - \frac{\langle p' u'_k \rangle}{\rho} \right) - \beta g_i \langle u'_i T' \rangle,\end{aligned} \tag{1.85}$$

or in its compact form as

$$\frac{Dk}{Dt} = P_k - S_k + D_k + G_k. \tag{1.86}$$

The first contribution on the right-hand side of the equation is a production term associated with the work of turbulent stresses and is responsible

for transferring turbulent energy from the mean motion to the fluctuations. Applying Boussinesq's hypothesis, this term can be rewritten as follows

$$P_k = -\langle u_i' u_k' \rangle \frac{\partial \langle u_i \rangle}{\partial x_k} = 2\nu_t S^2 \,. \tag{1.87}$$

The second term on the right-hand side represents the dissipation rate of turbulent kinetic energy, which originates from the work of turbulent stresses that dissipate turbulent kinetic energy into internal energy. The dissipative contribution is indicated as ε, and it has been modeled in [18] applying dimensionless considerations as a function of the mean turbulent kinetic energy and the mixing length. Thus, the dissipation rate is given by

$$S_k = \nu \langle \frac{\partial u_i'}{\partial x_k} \frac{\partial u_i'}{\partial x_k} \rangle = \varepsilon = C_D \frac{k^{3/2}}{\ell_m}, \tag{1.88}$$

where ℓ_m is the aforementioned mixing length, and C_D is a proportionality coefficient that depends on the type of flow considered. It usually assumes the value of 0.08. The diffusion term at the right-hand side of equation (1.85) is defined as the sum of three contributions and is responsible for redistributing turbulent kinetic energy within the domain. It includes the molecular diffusion $\nu \partial k / \partial x_k$, which is the diffusion of turbulent energy caused by the molecular transport process. The second term, $\langle u_i' u_i' u_k' \rangle$, is the turbulent transport and it is the rate at which turbulent energy is transported through the fluid by turbulent fluctuations. Lastly, the third term is the pressure diffusion, which is another form of turbulent transport resulting from the correlation of pressure and velocity fluctuations. These two last contributions can be modeled as a function of k as

$$-\langle u_i' u_i' u_k' \rangle - \frac{\langle p' u_k' \rangle}{\rho} = \frac{\nu_t}{\sigma_k} \frac{\partial k}{\partial x_k}. \tag{1.89}$$

As a result, the diffusion term can be rewritten as

$$D_k = \frac{\partial}{\partial x_k} \left(\nu \frac{\partial k}{\partial x_k} - \frac{1}{2} \langle u_i' u_i' u_k' \rangle - \frac{\langle p' u_k' \rangle}{\rho} \right) = \frac{\partial}{\partial x_k} \left[\left(\nu + \frac{\nu_t}{\sigma_k} \right) \frac{\partial k}{\partial x_k} \right], \tag{1.90}$$

where the coefficient σ_k is called the effective Prandtl-Schmidt number for diffusion, and it is empirically determined, in particular, for incompressible fluids, it is taken as a constant. We reformulate equation (1.85) by applying these considerations and we obtain the following transport equation for the turbulent kinetic energy

$$\frac{Dk}{Dt} = \frac{\partial}{\partial x_k} \left[\left(\nu + \frac{\nu_t}{\sigma_k} \right) \frac{\partial k}{\partial x_k} \right] + 2\nu_t S^2 - C_D \frac{k^{3/2}}{\ell_m} + G_k. \tag{1.91}$$

Given Equation (1.91), the turbulent kinetic energy can be determined and the turbulent viscosity can be computed using the following expression

$$\nu_t = \ell_m \sqrt{k}. \tag{1.92}$$

The most evident limitation of this model remains its dependence on the evaluation of mixing length, which does not have a universally valid definition.

1.3.3 Two-Equation Models

To address the limited versatility of earlier models and eliminate dependence on the definition of the mixing length, alternative differential models have been developed. Kolmogorov was the first to propose, in 1942 [20], the use of a second differential equation. According to [20], the turbulence model involves defining a second turbulent variables, generically defined as $k^\alpha \ell_m^\beta$, and writing its transport equation similarly to that of k. The introduction of the second variable enables the modeling of turbulent viscosity as a function of k and a characteristic time scale, τ_u, which in turn depends on the introduced variables. Hence, we can formulate the turbulent viscosity as

$$\nu_t = C_\nu k \tau_u \,. \tag{1.93}$$

where C_ν is a constant coefficient determined according to the turbulent model. Both the coefficient and the dynamic time scale will be introduced in the following sections.

$k - \varepsilon$ Model

One of the first two-equations model was proposed by Chou in 1945 [21] and then refined by Jones and Launder in 1972 [22]. It introduces the dissipation rate of turbulent energy, ε, as the second variable. Based on dimensional considerations, the characteristic time scale is defined as

$$\tau_u = \frac{k}{\varepsilon}\,, \tag{1.94}$$

therefore, the turbulent viscosity can be simply derived as

$$\nu_t = C_\mu \frac{k^2}{\varepsilon}. \tag{1.95}$$

where the constant C_μ usually assumes the value of 0.09.

The transport equation for ε can be derived by applying the following mathematical steps to the transport equation for velocity fluctuation (1.82). Firstly, the process involves differentiating the equation (1.82) with respect to x_k. Then, the resulting equation has to be multiplied by the quantity

$$2\nu\frac{\partial u_i'}{\partial x_k}\,. \tag{1.96}$$

Lastly, the time-averaging process and some algebraic manipulations of the equation terms lead to the following transport equation

$$\begin{aligned}\frac{D\varepsilon}{Dt} &= -2\nu\left(\langle\frac{\partial u_i'}{\partial x_k}\frac{\partial u_j'}{\partial x_k}\rangle + \langle\frac{\partial u_k'}{\partial x_j}\frac{\partial u_k'}{\partial x_i}\rangle\right)\frac{\partial\langle u_i\rangle}{\partial x_j} + \\ &- 2\nu\left(\langle\frac{\partial u_i'}{\partial x_k}\frac{\partial u_j'}{\partial x_k}\frac{\partial u_i'}{\partial x_j}\rangle + \langle u_j'\frac{\partial u_i'}{\partial x_k}\rangle\frac{\partial^2 u_i}{\partial x_j\partial x_k} + \nu\langle\frac{\partial^2 u_i'}{\partial x_j\partial x_k}\rangle^2\right) + \\ &+ \frac{\partial}{\partial x_j}\left[\nu\frac{\partial\varepsilon}{\partial x_j} - \nu\langle u_j'\frac{\partial u_i'}{\partial x_k}\frac{\partial u_i'}{\partial x_k}\rangle - \frac{2\nu}{\rho}\langle\frac{\partial u_j'}{\partial x_k}\frac{\partial p'}{\partial x_k}\rangle\right] + \\ &- 2\nu g_i\beta\langle\frac{\partial T'}{\partial x_k}\frac{\partial u_i'}{\partial x_k}\rangle.\end{aligned} \tag{1.97}$$

or in its compact form as

$$\frac{D\varepsilon}{Dt} = P_\varepsilon - S_\varepsilon + D_\varepsilon + G_\varepsilon\,. \tag{1.98}$$

Since the resulting terms of the ε equation introduce other unknowns, some hypothesis have to be introduced [23, 24] in order to obtain a closed form of the equation. The terms on the first line of the right-hand side are the production terms, that can be modeled as

$$P_\varepsilon = -2\nu\left(\langle\frac{\partial u_i'}{\partial x_k}\frac{\partial u_j'}{\partial x_k}\rangle + \langle\frac{\partial u_k'}{\partial x_j}\frac{\partial u_k'}{\partial x_i}\rangle\right)\frac{\partial\langle u_i\rangle}{\partial x_j} = C_{\varepsilon 1}\frac{\varepsilon}{k}\langle u_i'u_k'\rangle\frac{\partial\langle u_i\rangle}{\partial x_k}, \tag{1.99}$$

where $\langle u_i'u_k'\rangle\partial\langle u_i\rangle/\partial x_k$ is the production term in turbulent kinetic energy equation (1.85). Additionally, given the Boussinesq hypothesis, we can rewrite the production term as

$$P_\varepsilon = 2\nu_t C_{\varepsilon 1}\frac{\varepsilon}{k}S^2\,. \tag{1.100}$$

The dissipation term, in the second line of the equation (1.97), includes the contribution of three terms, and it is modeled similarly to the production terms as

$$S_\varepsilon = -C_{\varepsilon 2}\frac{\varepsilon^2}{k}. \tag{1.101}$$

The third line of equation (1.97) represents the diffusion term, which consists of the sum of molecular diffusion and turbulent transport of ε. Given the same gradient-diffusion hypothesis adopted in (1.89) and neglecting the diffusion transport of ε due to pressure fluctuations, the term D_ε can be rewritten as

$$D_\varepsilon = \frac{\partial}{\partial x_k}\left(\nu + \frac{\nu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_k}. \tag{1.102}$$

The buoyancy term can be modeled using the buoyancy production term of k equation as

$$G_\varepsilon = c_b \frac{\varepsilon}{k} G_k \,. \tag{1.103}$$

Finally, the two equations k-ε model commonly referred to as the standard $k-\varepsilon$ model are summed up as follows

$$\frac{Dk}{Dt} = \frac{\partial}{\partial x_k}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_k}\right] + 2\nu_t S^2 - \varepsilon + G_k \,, \tag{1.104}$$

$$\frac{D\varepsilon}{Dt} = \frac{\partial}{\partial x_k}\left[\left(\nu + \frac{\nu_t}{\sigma_\varepsilon}\right)\frac{\partial \varepsilon}{\partial x_k}\right] + C_{\varepsilon 1}\frac{\varepsilon}{k}P_k - C_{\varepsilon 2}\frac{\varepsilon^2}{k} + c_b\frac{\varepsilon}{k}G_k \,, \tag{1.105}$$

where the coefficients in the standard $k-\varepsilon$ model are assumed to be constant and are defined as

$$C_\mu = 0.09,\ C_{\varepsilon 1} = 1.44,\ C_{\varepsilon 2} = 1.92,\ c_b = 1.2,\ \sigma_k = 1.4,\ \sigma_\varepsilon = 1.3. \tag{1.106}$$

The assumption of a constant coefficient is an approximation that may be acceptable for simple flows but can prove quite inaccurate for complex flows. Over the years, other versions of the k-ε model have been developed to assign more detailed expressions to the coefficient in the standard model. Among them, we introduce the Re-Normalisation Group methods known as RNG $k-\varepsilon$ model and the k-ε model introduced by Manservisi and Menghini in [25]. The former has been developed by Yakhot et al. [26, 27] and it provides the following expression for modeling the coefficient of the dissipation term, $C_{\varepsilon 2}$, of equation (1.105)

$$C^*_{\varepsilon 2} = C_{\varepsilon 2} + \frac{C_\mu \eta^3 (1 - \eta/\eta_0)}{1 + \beta\eta^3}, \tag{1.107}$$

where

$$\eta = \frac{Sk}{\varepsilon}\,, \quad S = (2S_{ij}S_{ij})^{1/2}\,. \tag{1.108}$$

The other model constants adopted in [27] are as follows

$$C_\mu = 0.0845, \sigma_k = 0.7194,\ \sigma_\varepsilon = 0.7194,\ C_{\varepsilon 1} = 1.42, \\ C_{\varepsilon 2} = 1.68, \eta_0 = 4.38,\ \beta = 0.012\,. \tag{1.109}$$

Similarly, the model in [25] introduces a new expression for the $C_{\varepsilon 2}$ coefficient as the product of a constant and a damping function

$$C^*_{\varepsilon 2} = C_{\varepsilon 2} f_\varepsilon\,, \tag{1.110}$$

where the function f_ε is defined as

$$f_\varepsilon = \left(1 - \exp\left(-0.3226 R_d\right)\right)^2 \left(1 - 0.3 \exp\left(-0.0237 R_t^2\right)\right)\,. \tag{1.111}$$

The turbulent Reynolds number, R_t, is defined as

$$R_t = \frac{k^2}{\nu\varepsilon}\,, \tag{1.112}$$

while the dimensionless wall distance, R_d, is given by

$$R_d = \frac{\delta(\varepsilon\nu)^{\frac{1}{4}}}{\nu}\,, \tag{1.113}$$

where δ is the distance from the wall. The other constants of the model used by Manservisi and Menghini in [25] are provided below as

$$C_{\varepsilon 1} = 1.5,\ C_{\varepsilon 2} = 1.9,\ \sigma_k = 1.4,\ \sigma_\varepsilon = 1.4. \tag{1.114}$$

$k - \omega$ Model

In 1942, Kolmogorov in [20] proposed the first model based on two differential equations: one for the mean turbulent kinetic energy, and the second for the specific dissipation rate of turbulent kinetic energy. This second variable, indicated as ω, was defined by Kolmogorov as the ratio between the dissipation rate and the turbulent kinetic energy as

$$\omega = \frac{\varepsilon}{C_\mu k}, \tag{1.115}$$

from which the time scale $\tau_u = 1/\omega$ can be derived, while the turbulent viscosity is

$$\nu_t = \frac{k}{\omega}\,. \tag{1.116}$$

The transport equation for ω derived by Kolmogorov is one of the first formulations adopted. It states that

$$\frac{D\omega}{Dt} = -\beta\omega^2 + \frac{\partial}{\partial x_k}\left(\frac{\nu_t}{\sigma_\omega}\frac{\partial\omega}{\partial x_k}\right), \tag{1.117}$$

where the first term at right hand side is a dissipation term, while the second one is a turbulent diffusion term. The transport equation for ω introduced by Kolmogorov served as the foundation for the $k-\omega$-based models developed later.

The main contribution to the development of this model was carried out by Wilcox in 1988 [28]. He proposed a modified version incorporating the production and molecular diffusion terms to the (1.117). In his model, the production term assumes an analogous form as the one in (1.99), and thus it is given by

$$P_\omega = \gamma\frac{\omega}{k}P_k\,, \tag{1.118}$$

similarly, the buoyancy production term is

$$G_\omega = \gamma\frac{\omega}{k}G_k\,. \tag{1.119}$$

The diffusion contribution is defined as the diffusion terms in the previous turbulent transport equation for ω, thus the $k-\omega$ model takes the following form

$$\frac{Dk}{Dt} = \frac{\partial}{\partial x_k}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_k}\right] + 2\nu_t S^2 - \beta^* k\omega + G_k\,, \tag{1.120}$$

$$\frac{D\omega}{Dt} = \frac{\partial}{\partial x_k}\left[\left(\nu + \frac{\nu_t}{\sigma_\omega}\right)\frac{\partial \omega}{\partial x_k}\right] + \gamma\frac{\omega}{k}P_k - \beta\omega^2 + \gamma\frac{\omega}{k}G_k\,, \tag{1.121}$$

where we can notice that the dissipation term in the k equation has been written as a function of the specific dissipation rate simply applying its definition. In the model proposed by Wilcox in 1988, the coefficients are model constants and are defined as

$$\gamma = \frac{5}{9},\ \beta = \frac{3}{40},\ \beta^* = \frac{9}{100},\ \sigma_k = \sigma_\omega = \frac{1}{2}. \tag{1.122}$$

Ten years later, Wilcox proposed a more refined version of the $k-\omega$ model [29]. He defined a variant of the model that extended the treatment to compressible fluids within the boundary layer, involving a redefinition of the closure coefficients as follows

$$\gamma = \frac{13}{25},\ \beta = \frac{9}{25}f_\beta,\ \beta^* = \frac{9}{100}f^*_\beta, \tag{1.123}$$

where

$$f_\beta = \frac{1+70\chi_\omega}{1+80\chi_\omega},\ \chi_\omega = \frac{|\Omega_{ij}\Omega_{jk}S_{ki}|}{(0.09\omega)^3}, \tag{1.124}$$

and

$$f_\beta^* = \begin{cases} 1 & \text{if } \chi_k \leq 0, \\ \frac{1+680\chi_k^2}{1+400\chi_k^2} & \text{if } \chi_k > 0, \end{cases} \quad \chi_k = \frac{1}{\omega^3}\frac{\partial k}{\partial x_k}\frac{\partial \omega}{\partial x_k}, \tag{1.125}$$

and recalling the definition of the tensors Ω_{ij} and S_{ij}

$$\Omega_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} - \frac{\partial u_j}{\partial x_i}\right), \quad S_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right). \tag{1.126}$$

In 2008, the model was further improved with the addition of a term in the ω equation called cross diffusion [30]. This term depends upon gradients of both k and ω, and it is defined as

$$\frac{\sigma_d}{\omega}\frac{\partial k}{\partial x_k}\frac{\partial \omega}{\partial x_k}. \tag{1.127}$$

The coefficient σ_d is given by

$$\sigma_d = \begin{cases} 0 & \text{if } \frac{\partial k}{\partial x_k}\frac{\partial \omega}{\partial x_k} \leq 0, \\ \sigma_{do} & \text{if } \frac{\partial k}{\partial x_k}\frac{\partial \omega}{\partial x_k} > 0, \end{cases} \tag{1.128}$$

and the constant σ_{do} is defined according to the model. The equation (1.121) can be rewritten in the following form

$$\frac{D\omega}{Dt} = \frac{\partial}{\partial x_k}\left[\left(\nu + \frac{\nu_t}{\sigma_\omega}\right)\frac{\partial \omega}{\partial x_k}\right] + \frac{\sigma_d}{\omega}\frac{\partial k}{\partial x_k}\frac{\partial \omega}{\partial x_k} + \gamma\frac{\omega}{k}P_k - \beta\omega^2 + \gamma\frac{\omega}{k}G_k, \tag{1.129}$$

This model also imposes a constraint for the turbulent viscosity, which is expressed as

$$\nu_t = \frac{k}{\tilde{\omega}} \quad \text{with} \quad \tilde{\omega} = \max\left\{\omega, \frac{7}{8}\sqrt{\frac{2S_{ij}S_{ij}}{\beta^*}}\right\}. \tag{1.130}$$

Another approach for deriving the turbulent closure model is introduced in [31]. Here, the ω equation is derived from equations (1.104) and (1.105) by substituting the definition of ω in (1.105) and manipulating the resulting equation with straightforward algebraic steps. The resulting transport equation can be written as follows

$$\begin{aligned} \frac{D\omega}{Dt} = \frac{\partial}{\partial x_k}\left[\left(\nu + \frac{\nu_t}{\sigma_\omega}\right)\frac{\partial k}{\partial x_k}\right] + \frac{2}{k}\left(\nu + \frac{\nu_t}{\sigma_\omega}\right)\frac{\partial k}{\partial x_j}\frac{\partial \omega}{\partial x_j} + \\ + (c_{\varepsilon 1} - 1)\frac{\omega}{k}P_k - c_\mu(c_{\varepsilon 2}f_\varepsilon - 1)\omega^2 + (c_b - 1)\frac{\omega}{k}G_k, \end{aligned} \tag{1.131}$$

where the coefficient are defined as in (1.111) and (1.114). It worth noting that the derivation of the transport equation for ω, starting from the equation for ε, introduces the same cross-diffusion contribution as in the Wilcox model.

SST k-ω model

In this paragraph, another well-established method is presented. This model is the Shear Stress Transport (SST) $k-\omega$ model introduced by Menter et al. in 1993 [32, 33]. It is a two-equation eddy-viscosity model that combines both the $k-\varepsilon$ and $k-\omega$ models. This method employs the $k-\omega$ formulation within the inner regions of the boundary layer, extending all the way down to the wall through the viscous sub-layer. In the free-stream zone, instead, it uses the $k-\varepsilon$ model. The transport equations employed in the SST $k-\omega$ model are derived from the Wilcox model previously described and from the standard $k-\varepsilon$. In particular, the k transport equation is defined as equation (1.85), while the variation in ω assumes the form of equation (1.129). In this model, the coefficients are redefined as

$$\sigma_d = 2(1-F_1)\sigma_{\omega 2}, \tag{1.132}$$

$$\sigma_k = \frac{1}{F_1/\sigma_{k1} + (1-F_1)/\sigma_{k2}}, \sigma_\omega = \frac{1}{F_1/\sigma_{\omega 1} + (1-F_1)/\sigma_{\omega 2}}, \tag{1.133}$$

while β and γ are given by

$$\beta = \beta_1 F_1 + \beta_2(1-F_1), \gamma = \gamma_1 F_1 + \gamma_2(1-F_1). \tag{1.134}$$

The blending function $F1$ is defined as

$$F_1 = \tanh\left\{\min\left[\max\left(\frac{\sqrt{k}}{\beta^*\omega d}, \frac{500\nu}{d^2\omega}\right), \frac{4\omega^2 k}{CD_{k\omega}d^2}\right]^4\right\}, \tag{1.135}$$

$$CD_{k\omega} = \max\left(2\sigma_{\omega 2}\frac{1}{\omega}\frac{\partial k}{\partial x_i}\frac{\partial \omega}{\partial x_i}, 10^{-10}\right), \tag{1.136}$$

where d is the distance from the wall. The other model constants are

$$\begin{aligned}\beta^* =& \frac{9}{100}, \gamma_1 = \frac{5}{9}, \gamma_2 = 0.44, \beta_1 = \frac{3}{40}, \beta_2 = 0.0828,\\ &\sigma_{k1} = 0.85, \sigma_{k2} = 1, \sigma_{\omega 1} = 0.5, \sigma_{\omega 2} = 0.856.\end{aligned} \tag{1.137}$$

In this model, the expression for the turbulent viscosity is as follows

$$\nu_t = \frac{a_1 k}{\max(a_1\omega, SF_2)}, \tag{1.138}$$

where

$$F_2 = \tanh\left[\max\left(\frac{2\sqrt{k}}{\beta^*\omega d}, \frac{500\nu}{d^2\omega}\right)^2\right], \; a_1 = 0.31\,. \tag{1.139}$$

Near Wall Behavior

We have already introduced the concept of a local time scale characteristic of dynamic turbulence τ_{lu}, through which we can define the turbulent viscosity

$$\nu_t = C_\mu k \tau_{lu}(k,\varepsilon)\,, \text{or}\,, \nu_t = C_\mu k \tau_{lu}(k,\omega) \tag{1.140}$$

The idea of modeling turbulent viscosity using a local time scale arises from some observations about the behavior of wall-bounded flows. Here, the traditional models fail to provide an accurate prediction for the velocity field. The reason why turbulence shows incorrect prediction near the wall depends on the following factors [34]. The first reason is due to the influence of low Reynolds numbers. In such cases, the predominant effects of molecular viscosity in the region adjacent to the wall require specific treatment. The second motivation is the influence of wall proximity, which causes preferential damping of velocity fluctuations in the direction normal to the wall. Since the influence of the viscous stress tensor is significant, properly modeling the turbulent viscosity is important for the correct simulation of wall turbulence.

The definition of time scales has seen significant developments over the years. One of the most important contributions was made in 1990 by Nagano and Tagawa [35], who provided the following definition of ν_t

$$\nu_t = C_\mu f_\mu \frac{k^2}{\varepsilon}, \tag{1.141}$$

from which the constant τ_{lu} can be written as

$$\tau_{lu} = f_\mu \frac{k}{\varepsilon}. \tag{1.142}$$

The modeling of the damping function f_μ can be derived from considerations regarding wall effects and asymptotic trends near the wall. In the vicinity of the wall, the following proportionalities hold

$$k \propto y^2,\; \varepsilon \propto 1,\; \nu_t \propto y^3\,, \tag{1.143}$$

therefore, the asymptotic behavior of the ratio k^2/ε is

$$\frac{k^2}{\varepsilon} \propto y^4\,. \tag{1.144}$$

From these considerations, we can deduce that the function f_μ has to satisfy $f_\mu \propto y^{-1}$. The model presented in [34] formulates the following expression for the damping function

$$f_\mu = \left[1 - \exp\left(-\frac{y^+}{26}\right)\right]^2 \left(1 + \frac{4.1}{R_d^{3/4}}\right). \tag{1.145}$$

However, it has been verified that this model fails in simulations near the separation point and the reattachment of the boundary layer due to its dependence on the friction velocity u_τ. The dimensionless wall distance y^+ is, in fact, effectively zero around separating and reattaching points because of the friction velocity value. As a result, f_μ tends to zero as well as the turbulent viscosity, and it forces the Reynolds stress tensor to vanish, contrary to what occurs in real phenomena.

For this reason, in a more recent model [36], the friction velocity has been replaced with the Kolmogorov velocity scale $u_\varepsilon = (\nu\varepsilon)^{1/4}$. According to this model, f_μ can be modeled as

$$f_\mu = \left\{1 - \exp\left[-\left(\frac{y^*}{14}\right)^2\right]\right\}^2 \left\{1 + \frac{5}{R_t^{3/4}} \exp\left[-\left(\frac{R_t}{200}\right)^2\right]\right\}, \tag{1.146}$$

where

$$y^* = \frac{u_\epsilon y}{\nu}. \tag{1.147}$$

Following the previous example, the function f_μ was further refined in [37] and replaced with

$$f_\mu = \left\{1 - \exp\left[\left(\frac{R_d}{26}\right)^2\right]\right\}^2 \left\{1 + \frac{35}{R_t^{3/4}} \exp\left[-\left(\frac{R_t}{30}\right)^{3/4}\right]\right\}, \tag{1.148}$$

where R_d is the wall dimensionless wall distance

$$R_d = \frac{\delta}{(\nu^3/\varepsilon)^{1/4}}. \tag{1.149}$$

The expression adopted for modeling τ_{lu} in [25] is given by

$$\tau_{lu} = f_{1\mu} A_{1\mu} + f_{2\mu} A_{2\mu}, \tag{1.150}$$

where

$$f_{1\mu} = \left[1 - \exp\left(-\frac{R_d}{14}\right)\right]^2, \quad A_{1\mu} = \tau_u, \tag{1.151}$$

$$f_{2\mu} = f_{1\mu} \exp\left[-2.5 \times 10^{-5} R_t^2\right], \quad A_{2\mu} = \tau_u \frac{5}{R_t^{3/4}}. \tag{1.152}$$

Thus, we can express the local time scale as

$$\tau_{lu} = \tau_u \left[1 - \exp\left(-\frac{R_d}{14}\right)\right]^2 \left[1 + \exp\left(-2.5 \times 10^{-5} R_t^2\right) \frac{5}{R_t^{3/4}}\right]. \tag{1.153}$$

1.4 Thermal Turbulence Modeling

Thermal eddy diffusivity models express the turbulent heat flux as a function of the mean temperature field. This class of models is based on the following assumption

$$\langle u_i' T' \rangle = \alpha_t \frac{\partial \langle T \rangle}{\partial x_i}, \tag{1.154}$$

where α_t is the turbulent thermal diffusivity. This approximation is commonly known as the Simple Gradient Diffusion Hypothesis (SGDH). Applying the SGDH to the Reynolds averaged energy equation (1.62), we can write

$$\frac{D\langle T \rangle}{Dt} = \frac{\partial}{\partial x_i}\left[(\alpha + \alpha_t) \frac{\partial \langle T \rangle}{\partial x_i}\right]. \tag{1.155}$$

In equation (1.155), the number of unknowns is reduced from the three components of the turbulent heat flux of Equation (1.62) to the scalar α_t. As in the dynamic models discussed above, to address the closure problem in Reynolds averaged equations, it is necessary to introduce models for defining the turbulent thermal diffusivity.

Eddy thermal diffusivity is often derived using an additional simplifying hypothesis called Reynolds analogy, which expresses α_t as a function of ν_t through a proportionality relationship

$$\alpha_t = \frac{\nu_t}{Pr_t}, \tag{1.156}$$

where Pr_t is the turbulent Prandtl number. This similarity reduce the closure problem to determine the unknown proportionality factor. Almost all the commercial codes assume $Pr_t \simeq 0.85$ implying that no additional closure equations beyond those for the velocity field are needed to determine the mean temperature field. However, this simple approach can yield acceptable results in simulations involving fluids with a molecular Prandtl number $Pr \simeq 1$. As discussed in [38], for medium to high Prandtl number fluids the spatial value

of the turbulent Prandt number is approximately independent of the distance from the wall and assuming a constant value of $\simeq 1$ can be an acceptable hypothesis. For low-Prandtl numbers fluid, instead, it has been demonstrated that Pr_t depends on many parameters, such as the Reynolds number and the distance from the wall. Therefore, for liquid metals simulations, more sophisticated closure models are required to overcome the Reynolds analogy deficiencies.

For fluids with $Pr \ll 1$, thermal turbulence models that do not rely on the similarity hypothesis can be employed. These models use a set of scalar differential equations to determine the characteristic time scale for thermal turbulence τ_α, in analogy with τ_u. Turbulent thermal diffusivity can be defined as

$$\alpha_t = C_\alpha k \tau_\alpha, \tag{1.157}$$

where C_α is a model constant.

$k_\theta - \varepsilon_\theta$ Model

The $k_\theta - \varepsilon_\theta$ model is based on two quantities: k_θ, which represents the temperature variance, and ε_θ, which represents the dissipation of the temperature variance field. These quantities are defined as follows

$$k_\theta = \frac{1}{2}\langle T'^2 \rangle, \quad \varepsilon_\theta = \alpha \frac{\partial T'}{\partial x_j}\frac{\partial T'}{\partial x_j}, \tag{1.158}$$

where T' is the fluctuating temperature field. The introduction of these two variables allows the identification of the thermal timescale similarly to the dynamic time scale. Indeed, we can define τ_α as

$$\tau_\alpha = \frac{k_\theta}{\varepsilon_\theta}. \tag{1.159}$$

The transport equation for the temperature variance is derived by multiplying the equation for the instantaneous temperature (1.45) by the temperature fluctuation and applying the Reynolds average process [39]. The equation for k_θ can be therefore written as

$$\begin{aligned} \frac{D\frac{1}{2}\langle T'^2 \rangle}{Dt} = -\langle u_j' T' \rangle \frac{\partial \langle T \rangle}{\partial x_j} - \alpha \langle \frac{\partial T'}{\partial x_j}\frac{\partial T'}{\partial x_j} \rangle + \\ + \frac{\partial}{\partial x_j}\left(\alpha \frac{\partial \frac{1}{2}\langle T'^2 \rangle}{\partial x_j} - \frac{\langle u_j' T'^2 \rangle}{2} \right). \end{aligned} \tag{1.160}$$

or in its compact form as

$$\frac{Dk_\theta}{Dt} = P_{k_\theta} - S_{k_\theta} + D_{k_\theta} \,. \tag{1.161}$$

The derivation of the transport equation for k_θ introduce other unknowns to the system and some simplification has to be applied. In particular, the production contribution (first term at right hand side of Equation (1.160)), can be reformulate using the SGDH model as

$$P_{k_\theta} = -\langle u_j' T' \rangle \frac{\partial \langle T \rangle}{\partial x_j} = \alpha_t \frac{\partial \langle T \rangle}{\partial x_j} \frac{\partial \langle T \rangle}{\partial x_j} \,. \tag{1.162}$$

The third term at right hand side is the diffusion term and can be modeled similarly to the diffusion terms in dynamic turbulent equations. Thus, the following expression can be adopted

$$D_{k_\theta} = \frac{\partial}{\partial x_j} \left(\alpha \frac{\partial k_\theta}{\partial x_j} - \frac{\langle u_j' T'^2 \rangle}{2} \right) = \frac{\partial}{\partial x_j} \left[\left(\alpha + \frac{\alpha_t}{\sigma_{k_\theta}} \right) \frac{\partial k_\theta}{\partial x_j} \right] \,. \tag{1.163}$$

where α_{k_θ} is a model constant. Lastly, the dissipation term is equivalent to ε_θ and it is obtained by solving the corresponding transport equation. The transport equation for the dissipation rate of temperature fluctuations is derived by taking the derivative of the equation of the instantaneous temperature (1.45) with respect to x_j, multiplying by the term $2\alpha \partial T'/\partial x_j$ and averaging. We therefore obtain the following equation

$$\begin{aligned} \frac{D\varepsilon_\theta}{Dt} = & -2\alpha \langle \frac{\partial T'}{\partial x_k} \frac{\partial u_j'}{\partial x_k} \rangle \frac{\partial \langle T \rangle}{\partial x_j} - 2\alpha \langle u_j' \frac{\partial T'}{\partial x_j} \rangle \frac{\partial^2 \langle T \rangle}{\partial x_j \partial x_k} + \\ & - 2\alpha \langle \frac{\partial T'}{\partial x_k} \frac{\partial T'}{\partial x_j} \rangle \frac{\partial \langle u_j \rangle}{\partial x_k} - 2\alpha \langle \frac{\partial T'}{\partial x_k} \frac{\partial u_j'}{\partial x_k} \frac{\partial T'}{\partial x_j} \rangle + \\ & - 2\alpha^2 \langle \left(\frac{\partial^2 T'}{\partial x_j \partial x_k} \right)^2 \rangle - \frac{\partial}{\partial x_j} \left(\langle \varepsilon_\theta u_j' \rangle - \alpha \frac{\partial \varepsilon_\theta}{\partial x_j} \right) \,, \end{aligned} \tag{1.164}$$

or in its compact form

$$\frac{D\varepsilon_\theta}{Dt} = P_{\varepsilon_\theta 1} + P_{\varepsilon_\theta 2} + P_{\varepsilon_\theta 3} + P_{\varepsilon_\theta} - S_{\varepsilon_\theta} + D_{\varepsilon_\theta} \tag{1.165}$$

where the production contribution $P_{\varepsilon_\theta 1}$, $P_{\varepsilon_\theta 2}$, $P_{\varepsilon_\theta 2}$ and P_{ε_θ} are, respectively, the first, second, third and fourth terms at the right-hand side of the equation (1.164). The dissipation S_{ε_θ} is the fifth term, while the diffusion D_{ε_θ} is the

last term at right hand side. The production term and the dissipation term can be modeled as

$$P_{\varepsilon_\theta 1} + P_{\varepsilon_\theta 2} + P_{\varepsilon_\theta 3} + P_{\varepsilon_\theta} - S_{\varepsilon_\theta} = c_{p1}\frac{\varepsilon_\theta}{k_\theta}P_{k_\theta} + c_{p2}\frac{\varepsilon_\theta}{k}P_k - c_{d1}\frac{\varepsilon_\theta^2}{k_\theta} - c_{d2}\frac{\varepsilon\varepsilon_\theta}{k}, \quad (1.166)$$

where $P_{\varepsilon_\theta 2}$ is generally neglected and the coefficients c_{p1}, c_{p2}, c_{d1}, and c_{d2} are characteristic parameters of the model. The diffusion term is modeled similarly to the other diffusion terms as

$$D_{\varepsilon_\theta} = -\frac{\partial}{\partial x_j}\left(\langle\varepsilon_\theta u_j'\rangle - \alpha\frac{\partial\varepsilon_\theta}{\partial x_j}\right) = \frac{\partial}{\partial x_j}\left[\left(\alpha + \frac{\alpha_t}{\sigma_{\varepsilon_\theta}}\right)\frac{\partial\varepsilon_\theta}{\partial x_j}\right]. \quad (1.167)$$

The $k-\varepsilon$ model can be summed up as

$$\frac{Dk_\theta}{Dt} = \frac{\partial}{\partial x_j}\left[\left(\alpha + \frac{\alpha_t}{\sigma_{k_\theta}}\right)\frac{\partial k_\theta}{\partial x_j}\right] + \alpha_t\frac{\partial\langle T\rangle}{\partial x_j}\frac{\partial\langle T\rangle}{\partial x_j} - \varepsilon_\theta, \quad (1.168)$$

$$\frac{D\varepsilon_\theta}{Dt} = \frac{\partial}{\partial x_j}\left[\left(\alpha + \frac{\alpha_t}{\sigma_{\varepsilon_\theta}}\right)\frac{\partial\varepsilon_\theta}{\partial x_j}\right] + \frac{\varepsilon_\theta}{k_\theta}(c_{p1}P_{k_\theta} - c_{d1}\varepsilon_\theta) + \frac{\varepsilon_\theta}{k}(c_{p2}P_k - c_{d2}\varepsilon). \quad (1.169)$$

where the coefficients are defined as in [31]

$$\begin{aligned} &\sigma_{k_\theta} = \sigma_{\varepsilon_\theta} = 1.4,\ c_{p1} = 0.925, c_{p2} = 0.9,\ c_{d1} = 0.9, \\ &c_{d2} = 1.9\left[1 - \exp\left(-0.0308R_d\right)\right]^2\left[1 - 0.3\exp\left(-0.0237R_t^2\right)\right]. \end{aligned} \quad (1.170)$$

$k_\theta - \omega_\theta$ Model

The use of the k_θ-ω_θ model for simulating thermal turbulence is carried out in [31] and [40], where it is discussed and validated. This model can be obtained by defining the specific dissipation rate of temperature variance as

$$\omega_\theta = \frac{\varepsilon_\theta}{C_\mu k_\theta}, \quad (1.171)$$

which leads to the definition of a time scale as

$$\tau_\alpha = \frac{1}{\omega_\theta}. \quad (1.172)$$

Substituting the definition of ω_θ in (1.168) and (1.169) we can deduce the following eddy diffusivity model

$$\frac{Dk_\theta}{Dt} = \frac{\partial}{\partial x_j}\left[\left(\alpha + \frac{\alpha_t}{\sigma_{k_\theta}}\right)\frac{\partial k_\theta}{\partial x_j}\right] + P_{k_\theta} - c_\mu k_\theta\omega_\theta, \quad (1.173)$$

$$\frac{D\omega_\theta}{Dt} = \frac{\partial}{\partial x_j}\left[\left(\alpha + \frac{\alpha_t}{\sigma_{\omega_\theta}}\right)\frac{\partial \omega_\theta}{\partial x_j}\right] + \frac{2}{k_\theta}\left(\alpha + \frac{\alpha_t}{\sigma_{\omega_\theta}}\right)\frac{\partial k_\theta}{\partial x_j}\frac{\partial \omega_\theta}{\partial x_j} + \\ + (c_{p1} - 1)\frac{\omega_\theta}{k_\theta}P_{k_\theta} + c_{p2}\frac{\omega_\theta}{k}P_k - (c_{d1} - 1)\omega_\theta^2 - c_{d2}c_\mu\omega\omega_\theta. \tag{1.174}$$

Here, the production and dissipation terms have been modeled through the following expressions

$$P_{\omega_\theta} = P_\varepsilon - \frac{\omega_\theta}{k_\theta}P_{k_\theta}, \quad S_{\omega_\theta} = S_\varepsilon - \frac{\omega_\theta}{k_\theta}S_{k_\theta}. \tag{1.175}$$

At right-hand side of Equation (1.174), we identify in order the diffusive term, the mixed term or cross diffusion, the contributions of thermal and mechanical production, and finally, the dissipation terms. The coefficients c_{p2} and c_{d2} have the same values as in (1.170), while c_{p1} and c_{d1} have been redefined to ensure the positivity of the coefficients of the production and dissipation terms

$$c_{p1} = 1.025 + C_{inc}, \quad c_{d1} = 1 + C_{inc}, \tag{1.176}$$

where $C_{inc} = 0.1$ is a constant determined by comparison with experimental results.

Near Wall Behavior

Experimental evidence shows that predicting turbulent heat flux using the similarity hypothesis $\alpha_t = \nu_t/Pr_t$ with $Pr_t \simeq 1$ is fairly accurate for fluids with $Pr \simeq 1$ and not for low Prandtl number fluids. Thus, several models have been proposed in order to model an appropriate characteristic thermal time scale. Recalling the modeling of the turbulent diffusivity as a function of a local characteristic thermal time scale, we have

$$\alpha_t = C_\alpha k \tau_m, \tag{1.177}$$

where the time constant τ_m is the mixed time-scale. Given the (1.177) and the definition of turbulent Prandtl number as $Pr_t = \nu_t/\alpha_t$, then the thermal time scale can be defined as

$$\tau_m \propto \tau_u \frac{1}{Pr_t}, \tag{1.178}$$

from which it follows that it is possible to define the thermal time scale based on the Pr_t behavior. The modeling of Pr_t as in [25] depends on three

contributions according to the distance from the wall. The central region depends on the harmonic average of the dynamical time scale and the thermal time scale as

$$\tau_m \propto \frac{1}{\frac{1}{\tau_u} + \frac{C_m}{\tau_t}} = \tau_u \frac{R}{C_m + R}. \tag{1.179}$$

where R is the time-scale ratio of the thermal to mechanical turbulent time scales defined as $R = \tau_\theta/\tau_u = k_\theta \varepsilon/(k\varepsilon_\theta)$. In the bulk region, τ_m is independent of the time ratio R, and the turbulent diffusion is assumed to be dominated only by velocity fluctuations. In the bulk region, therefore we assume that

$$\tau_m \propto \tau_u \frac{1}{Pr_{t,\infty}}, \tag{1.180}$$

where $Pr_{t,\infty}$ can be assumed constant and uniform or can be modeled, for example by means of Kays model $Pr_t = 0.85 + 0.7/Pr$ [41]. We also should introduce a model function in the τ_m expression to account for the wall-proximity effects. For the near-wall region, the characteristic time scale is

$$\tau_m \propto \tau_u \frac{\sqrt{2R}}{Pr R_t^{3/4}}, \tag{1.181}$$

The characteristic thermal time scale is then modeled as:

$$\tau_m = \tau_u f_{1\theta} \left(\frac{1}{Pr_t} + \frac{2R}{R + C_\gamma} f_{2\theta} + 1.3 \frac{\sqrt{2R}}{Pr R_t^{3/4}} f_{3\theta} \right), \tag{1.182}$$

where $C_\gamma = 0.25/Pr_t^{1/4}$ and the model functions are given by

$$f_{1\theta} = \left[1 - \exp\left(-\sqrt{Pr} R_d/19\right)\right] \left[1 - \exp\left(-R_d/14\right)\right], \tag{1.183}$$

$$f_{2\theta} = \exp\left[-(R_t/500)^2\right],\ f_{3\theta} = \exp\left[-(R_t/200)^2\right]. \tag{1.184}$$

1.5 Models for Reynolds Stress Tensor and Turbulent Heat Flux

All the eddy viscosity models presented in previous sections are based on Boussinesq's hypothesis, which reduces the unknowns of the Reynolds stress tensor to a single scalar unknown, ν_t. This assumption significantly simplifies the closure problem and implies that the Reynolds stress tensor is parallel to the strain-rate tensor of the mean field. However, it has been observed

that such alignment does not generally occur, and, in particular, it fails in scenarios involving flows with rapid variations in the strain-rate tensor. This limitation results in inaccuracies when predicting secondary flows caused by turbulence's anisotropy. Eddy viscosity models also fail to simulate flows over curved surfaces, within ducts exhibiting secondary flows, and in cases of boundary layer separation. Similarly, the eddy diffusivity models assume the turbulent heat flux parallel to the temperature gradient, and while simple, they are inappropriate for more complex problems, such as natural convection regimes.

These considerations have led to the development of more advanced methods aimed at enhancing the numerical accuracy of anisotropic phenomena. One main approach relies on modeling and solving a specific equation for each component of the Reynolds stress tensor and the turbulent heat flux. These models are categorized as nonlinear eddy viscosity models and nonlinear eddy diffusivity models. Among them, the anisotropic models belong to a class of intermediate models between first-order and second-order formulations, introducing a set of algebraic equations for the closure of the turbulence problem. In this section, the Explicit Algebraic Stress Model (EASM) for the Reynolds stress tensor and the Explicit Algebraic Heat Flux Model (EAHFM) for the turbulent heat flux are derived.

1.5.1 Explicit Algebraic Stress Models

The main contribution to the development of explicit algebraic models for the components of the Reynolds stress tensor, known as Explicit Algebraic Stress Models, is attributed to Pope [42]. He introduced a methodology to derive an explicit relationship for the Reynolds stress tensor, starting from the implicit algebraic model proposed by Rodi [43], which in turn derived a model based on the earlier model by Launder et al. [44]. In implicit form, the constitutive relation introduced by Rodi represents a model known as the Algebraic Stress Model (ASM). These implicit models were eventually replaced by explicit models for the Reynolds stress tensor thanks to the contribution of Abe et al. in 1997 [37] and Hattori and Nagano in 2004 [45]. This thesis focuses on a more recent formulation proposed by Hattori et al. in 2006 [46], derived from the earlier version [45]. This model introduces a new explicit algebraic formulation for the Reynolds stress tensor that incorporates buoyancy effects.

Recalling the transport equation for the Reynolds stress tensor

$$\frac{D\langle u_i' u_j'\rangle}{Dt} = -\langle u_i' u_k'\rangle \frac{\partial \langle u_j\rangle}{\partial x_k} - \langle u_j' u_k'\rangle \frac{\partial \langle u_i\rangle}{\partial x_k} - 2\nu \langle \frac{\partial u_i'}{\partial x_k}\frac{\partial u_j'}{\partial x_k}\rangle + \\ + \frac{1}{\rho}\left(\langle p' \frac{\partial u_j'}{\partial x_i}\rangle + \langle p' \frac{\partial u_i'}{\partial x_j}\rangle\right) + \frac{\partial}{\partial x_k}\left(\nu \frac{\partial \langle u_i' u_j'\rangle}{\partial x_k} - \langle u_i' u_j' u_k'\rangle + \right. \\ \left. - \langle \frac{p'}{\rho}(\delta_{ki} u_j' + \delta_{kj} u_i')\rangle\right) - \beta\left(g_j \langle u_i' T'\rangle + g_i \langle u_j' T'\rangle\right), \tag{1.185}$$

we report the compact form of the aforementioned equation as

$$\frac{D\langle u_i' u_j'\rangle}{Dt} = P_{ij} - \varepsilon_{ij} + \Phi_{ij} + D_{ij} + G_{ij}. \tag{1.186}$$

where P_{ij} is the production term, ε_{ij} is the dissipation term, Φ_{ij} is the pressure-strain correlation term, D_{ij} is the diffusion term including the molecular diffusion and the turbulent and pressure diffusion contribution, and finally G_{ij} is a buoyant term. We recall also the compact form of the turbulent kinetic energy transport equation as

$$\frac{Dk}{Dt} = P_k - S_k + D_k + G_k\,. \tag{1.187}$$

We can decompose the Reynolds stress tensor as a sum of the isotropic stress and a deviatoric part, a_{ij}, as

$$\langle u_i' u_j'\rangle = \frac{2}{3} k \delta_{ij} + a_{ij}\,. \tag{1.188}$$

The normalized anisotropy tensor b_{ij} is defined as

$$b_{ij} = \frac{a_{ij}}{2k} = \frac{\langle u_i' u_j'\rangle}{2k} - \frac{\delta_{ij}}{3}\,. \tag{1.189}$$

Considering the spatial and temporal variations in b_{ij}, we can obtain

$$\frac{Db_{ij}}{Dt} = \frac{1}{2k}\frac{D\langle u_i' u_j'\rangle}{Dt} - \frac{\langle u_i' u_j'\rangle}{2k^2}\frac{Dk}{Dt}\,. \tag{1.190}$$

Substituting (1.189) into (1.190) we obtain

$$\frac{Db_{ij}}{Dt} = \frac{1}{2k}\frac{D\langle u_i' u_j'\rangle}{Dt} - \frac{b_{ij} + \delta_{ij}/3}{k}\frac{Dk}{Dt}. \tag{1.191}$$

Applying equations (1.186) and (1.187) to equation (1.191) and negletting the diffusion terms we obtain

$$\frac{Db_{ij}}{Dt} = \frac{1}{2k}\left(P_{ij} + G_{ij} + \Phi_{ij} - \varepsilon_{ij}\right) - \frac{b_{ij} + \delta_{ij}/3}{k}\left(P_k + G_k - \varepsilon\right). \tag{1.192}$$

We now assume the hypothesis of local equilibrium state for which the spatial and temporal variations in b_{ij} are zero. This assumption yields the following equation

$$P_{ij} + G_{ij} + \Phi_{ij} - \varepsilon_{ij} = 2\left(b_{ij} + \frac{\delta_{ij}}{3}\right)(P_k + G_k - \varepsilon)\,. \tag{1.193}$$

Using the expression of the dissipation term introduced by Gatski and Speziale, in 1993 [47] that split it into isotropic and deviatoric parts

$$\varepsilon_{ij} = \frac{2}{3}\varepsilon\delta_{ij} + \varepsilon_{D_{ij}}\,, \tag{1.194}$$

and rewriting the production term P_{ij} as follows

$$\begin{aligned} P_{ij} = -\frac{4}{3}kS_{ij} - 2k\left(b_{ik}S_{jk} + b_{jk}S_{ik} - \frac{2}{3}b_{mn}S_{mn}\delta_{ij}\right) + \\ -2k(b_{ik}\Omega_{jk} + b_{jk}\Omega_{ik})\,, \end{aligned} \tag{1.195}$$

we can derive the following formulation of the equation (1.193)

$$\begin{aligned} (P_k + G_k - \varepsilon)b_{ij} = -\frac{2}{3}kS_{ij} - k\left(b_{ik}S_{jk} + b_{jk}S_{ik} - \frac{2}{3}b_{mn}S_{mn}\delta_{ij}\right) + \\ -k\left(b_{ik}\Omega_{jk} + b_{jk}\Omega_{ik}\right) + \frac{1}{2}\Pi_{ij} + \frac{1}{2}\left(G_{ij} - \frac{2}{3}\delta_{ij}G_k\right), \end{aligned} \tag{1.196}$$

where $\Pi_{ij} = \Phi_{ij} - \varepsilon_{D_{ij}}$. Modeling Π_{ij} using second-order closure models in b_{ij} [48] results in a class of models that are implicit, as the Reynolds stress tensor appears on both sides of the equation. However, the primary drawback of implicit algebraic stress models is the need for matrix inversions, which can be overcome by using explicit models, thereby offering significant computational savings. In order to derive the explicit equation for b_{ij} in terms of the mean velocity gradients, we can model Φ_{ij} using an expression that is linear in the anisotropy tensor b_{ij}. Explicit relations for b_{ij} were proposed first by Pope in 1975 in [42] for the two-dimensional problem and extended to three-dimensional flows by Gatski and Speziale in 1993 in [47]. The general linear model for Π_{ij}, which includes the buoyant term, is given by

$$\begin{aligned} \Pi_{ij} = -C_1\varepsilon b_{ij} + C_2kS_{ij} + C_3k\left(b_{ik}S_{jk} + b_{jk}S_{ik} - \frac{2}{3}b_{mn}S_{mn}\delta_{ij}\right) + \\ +C_4k\left(b_{ik}\Omega_{jk} + b_{jk}\Omega_{ik}\right) - C_5\left(G_{ij} - \frac{2}{3}\delta_{ij}G_k\right), \end{aligned} \tag{1.197}$$

where C_1–C_5 are model constants. By substituting Equation (1.197) into Equation (1.196), we derive the following relation

$$b_{ij}^* = -S_{ij}^* - \left(b_{ik}^* S_{jk}^* + b_{jk}^* S_{ik}^* - \frac{2}{3} b_{k\ell}^* S_{k\ell}^* \delta_{ij} \right) + b_{ik}^* \Omega_{kj}^* + b_{jk}^* \Omega_{ki}^* - f_{ij}^*, \quad (1.198)$$

where the nondimensional quantities introduced to obtain (1.198) are as follows

$$\begin{aligned} S_{ij}^* &= \frac{1}{2} g\tau (2 - C_3) S_{ij}, \\ \Omega_{ij}^* &= \frac{1}{2} g\tau (2 - C_4) \Omega_{ij}, \\ b_{ij}^* &= \left(\frac{C_3 - 2}{C_2 - \frac{4}{3}} \right) b_{ij}, \quad \tau = \frac{k}{\varepsilon}, \\ f_{ij}^* &= g\tau \frac{(C_5 - 1)(C_3 - 2)}{2(C_2 - \frac{4}{3})k} \left(G_{ij} - \frac{2}{3} G_k \delta_{ij} \right), \\ g &= \left(\frac{1}{2} C_1 + \frac{P_k}{\varepsilon} + \frac{G_k}{\varepsilon} - 1 \right)^{-1}. \end{aligned} \quad (1.199)$$

Equation (1.198) can be written in the matrix form as

$$\mathbf{b}^* = -\mathbf{S}^* - \left(\mathbf{b}^* \mathbf{S}^* + \mathbf{S}^* \mathbf{b}^* - \frac{2}{3} \{ \mathbf{b}^* \mathbf{S}^* \} \mathbf{I} \right) + \mathbf{b}^* \boldsymbol{\Omega}^* - \boldsymbol{\Omega}^* \mathbf{b}^* - \mathbf{f}^*, \quad (1.200)$$

where $\{\cdot\}$ indicates the trace operator. To derive an explicit form of $\mathbf{b}^*$ from Equation (1.200), we use the integrity basis definition proposed by Pope in 1975 [42], which is

$$\mathbf{b}^* = \sum_{\lambda} Q^{(\lambda)} \mathbf{T}^{(\lambda)}, \quad (1.201)$$

where $\mathbf{T}^{(\lambda)}$ is the integrity basis for functions of a symmetric and anti-symmetric tensor and $Q^{(\lambda)}$ are scalar functions of the irreducible invariants of $\mathbf{S}^*$ and $\boldsymbol{\Omega}^*$. Therefore, the Reynolds stresses can be expressed as known functions of a finite number of tensors $\mathbf{T}^{(\lambda)}$ and an equal number of unknown scalars $Q^{(\lambda)}$, which depend on a finite set of known invariants. The characteristics of the tensor $\mathbf{T}^{(\lambda)}$ are derived from $\mathbf{b}^*$. Thus, the tensors are linearly independent and symmetric with zero trace. In this derivation, we use the

seven basis tensors introduced by So et al. in 2002 [49], which are defined as

$$\begin{aligned}
&\mathbf{T}^{(1)} = \mathbf{S}^*,\ \mathbf{T}^{(2)} = \mathbf{S}^*\boldsymbol{\Omega}^* - \boldsymbol{\Omega}^*\mathbf{S}^*,\\
&\mathbf{T}^{(3)} = \mathbf{S}^{*2} - \frac{1}{3}\{\mathbf{S}^{*2}\}\mathbf{I},\ \mathbf{T}^{(4)} = \mathbf{f}^*,\\
&\mathbf{T}^{(5)} = \mathbf{f}^*\boldsymbol{\Omega}^* - \boldsymbol{\Omega}^*\mathbf{f}^*,\ \mathbf{T}^{(6)} = \mathbf{f}^{*2} - \frac{1}{3}\{\mathbf{f}^{*2}\}\mathbf{I},\\
&\mathbf{T}^{(7)} = \mathbf{f}^*\mathbf{S}^* + \mathbf{S}^*\mathbf{f}^* - \frac{2}{3}\{\mathbf{f}^*\mathbf{S}^*\}\mathbf{I}.
\end{aligned} \tag{1.202}$$

The following procedure is then introduced to determine the values of the scalars $Q^{(\lambda)}$. Expanding the summation (1.201) using (1.202) we obtain

$$\begin{aligned}
\mathbf{b}^* = {} & Q^{(1)}\mathbf{S}^* + Q^{(2)}\left(\mathbf{S}^*\boldsymbol{\Omega}^* - \boldsymbol{\Omega}^*\mathbf{S}^*\right) + Q^{(3)}\left(\mathbf{S}^{*2} - \frac{1}{3}\{\mathbf{S}^{*2}\}\mathbf{I}\right) + \\
& + Q^{(4)}\mathbf{f}^* + Q^{(5)}\left(\mathbf{f}^*\boldsymbol{\Omega}^* - \boldsymbol{\Omega}^*\mathbf{f}^*\right) + Q^{(6)}\left(\mathbf{f}^{*2} - \frac{1}{3}\{\mathbf{f}^{*2}\}\mathbf{I}\right) + \\
& + Q^{(7)}\left(\mathbf{f}^*\mathbf{S}^* + \mathbf{S}^*\mathbf{f}^* - \frac{2}{3}\{\mathbf{f}^*\mathbf{S}^*\}\mathbf{I}\right).
\end{aligned} \tag{1.203}$$

We introduce the scalar functions H and J defined in [42]. This functions are related with $\mathbf{T}^{(\lambda)}$ by the following expressions

$$\mathbf{T}^{(\lambda)}\mathbf{S}^* + \mathbf{S}^*\mathbf{T}^{(\lambda)} - \frac{2}{3}\{\mathbf{T}^{(\lambda)}\mathbf{S}^*\}\mathbf{I} = \sum_{\gamma} H_{\lambda\gamma}\mathbf{T}^{(\gamma)}, \tag{1.204}$$

and

$$\mathbf{T}^{(\lambda)}\boldsymbol{\Omega}^* - \boldsymbol{\Omega}^*\mathbf{T}^{(\lambda)} = \sum_{\gamma} J_{\lambda\gamma}\mathbf{T}^{(\gamma)}. \tag{1.205}$$

Substituting (1.204), (1.205) and definition (1.201) into (1.200) we obtain the following equation for $\mathbf{T}^{(\lambda)}$

$$\begin{aligned}
\sum_{\lambda} Q^{(\lambda)}\mathbf{T}^{(\lambda)} = {} & -\sum_{\lambda}\delta_{1\lambda}\mathbf{T}^{(1)} - \sum_{\lambda} Q^{(\lambda)}\left(\sum_{\gamma} H_{\lambda\gamma}\mathbf{T}^{(\gamma)}\right) + \\
& + \sum_{\lambda} Q^{(\lambda)}\left(\sum_{\gamma} J_{\lambda\gamma}\mathbf{T}^{(\gamma)}\right) - \sum_{\lambda}\delta_{4\lambda}\mathbf{T}^{(4)}.
\end{aligned} \tag{1.206}$$

which correspond to a linear system of equations for the determination of $Q^{(\lambda)}$ since the tensors $\mathbf{T}^{(\lambda)}$ are independent. Thus, the following equation holds

$$Q^{(\lambda)} = -\delta_{1\lambda} - \delta_{4\lambda} - \sum_{\gamma} Q^{(\lambda)}H_{\lambda\gamma} + \sum_{\gamma} Q^{(\lambda)}J_{\lambda\gamma}. \tag{1.207}$$

This linear system can be written in the following form

$$A_{\gamma\lambda}Q^{(\lambda)} = B_\lambda\,, \tag{1.208}$$

where $A_{\gamma\lambda} = -\delta_{\lambda\gamma} - H_{\lambda\gamma} + J_{\lambda\gamma}$, and $B_\lambda = \delta_{1\lambda} + \delta_{4\lambda}$. In order to solve Equation (1.208), we need to invert the matrix $A_{\gamma\lambda}$. By using Cayley–Hamilton identities [47], the matrices $H_{\lambda\gamma}$ and $J_{\lambda\gamma}$ can be determined and consequently the inverse of the matrix $A_{\gamma\lambda}$ is derived. This process leads to the definitions of $Q^{(\lambda)}$ as follows

$$\begin{aligned} &Q^{(1)} = \frac{1}{D_1}, \quad Q^{(2)} = \frac{1}{D_1}, \quad Q^{(3)} = -\frac{2}{D_1}, \quad Q^{(4)} = \frac{1}{D_2}, \\ &Q^{(5)} = \frac{1}{D_2}, \quad Q^{(6)} = 0, \quad Q^{(7)} = \frac{1}{D_2}. \end{aligned} \tag{1.209}$$

where

$$D_1 = \frac{3 - 2\eta^2 + 6\zeta^2}{3}, \quad D_2 = 1 + 2\zeta^2, \tag{1.210}$$

with $\eta = \left(S^*_{ij}S^*_{ij}\right)^{1/2}$ and $\zeta = \left(\Omega^*_{ij}\Omega^*_{ij}\right)^{1/2}$. Substituting the expression (1.209) into Equation (1.203), we can obtain the expression for the normalized anisotropy tensor

$$\begin{aligned} b^*_{ij} = &- \frac{3}{3 - 2\eta^2 + 6\zeta^2}\left[S^*_{ij} + \left(S^*_{ik}\Omega^*_{kj} - \Omega^*_{ik}S^*_{kj}\right) + \right. \\ &\left. - 2\left(S^*_{ik}S^*_{kj} - \frac{1}{3}S^*_{mn}S^*_{nm}\delta_{ij}\right)\right] + \\ &- \frac{1}{1 + 2\zeta^2}\left[f^*_{ij} + \left(f^*_{ik}\Omega^*_{kj} - \Omega^*_{ik}f^*_{kj}\right) + \right. \\ &\left. + \left(f^*_{ik}S^*_{kj} + S^*_{ik}f^*_{kj} - \frac{2}{3}f^*_{mn}S^*_{mn}\delta_{ij}\right)\right], \end{aligned} \tag{1.211}$$

where $f_{ij} = G_{ij} - (2/3)G_k\delta_{ij}$. Here, the non-dimensional forms of b^*_{ij}, S^*_{ij}, Ω^*_{ij} and f^*_{ij} are adopted as described by Abe et al. in [37] and by Nagano et al. in [50] as

$$\begin{aligned} b^*_{ij} &= C_D b_{ij}, \\ S^*_{ij} &= C_D\tau S_{ij}, \\ \Omega^*_{ij} &= 72C_D\tau\Omega_{ij}, \\ f^*_{ij} &= C_g\left(\frac{\tau_{mg}}{k}\right)f_{ij}. \end{aligned} \tag{1.212}$$

Consequently, the Explicit Algebraic Stress Model adopts the following expression to account for buoyancy effects

$$\begin{aligned}\langle u_i' u_j' \rangle = &\frac{2}{3}k\delta_{ij} - \frac{2\nu_t}{f_{R1}}S_{ij} - \frac{4C_D k f_\tau}{f_{R1}}\left(S_{ik}\Omega_{kj} - \Omega_{ik}S_{kj}\right) + \\ &+ \frac{4C_D k f_\tau}{f_{R1}}\left(S_{ik}S_{kj} - \frac{1}{3}S_{mn}S_{mn}\delta_{ij}\right) + \\ &- \frac{2C_g\tau_{mg}}{C_D f_{R2}}f_{ij} - \frac{4C_g\tau_{mg}^2}{f_{R2}}\left(f_{ik}\Omega_{kj} - \Omega_{ik}f_{kj}\right) + \\ &+ \frac{2C_g\tau_{mg}^2}{f_{R2}}\left(f_{ik}S_{kj} + S_{ik}f_{kj} - \frac{2}{3}f_{mn}S_{mn}\delta_{ij}\right).\end{aligned} \tag{1.213}$$

with

$$f_{R1} = 1 + \frac{22}{3}(C_D\tau_{R_0})^2\Omega^2 + \frac{2}{3}(C_D\tau_{R_0})^2\left(\Omega^2 - S^2\right)f_B, \tag{1.214}$$

$$f_{R2} = 1 + 8(C_D\tau_{R_0})^2\Omega^2. \tag{1.215}$$

where τ_{mg} is the mixed scale of velocity and thermal fields, and f_τ is the function of characteristic time-scale reflecting wall-limiting behavior introduced by Nagano and Hattori in [51] as follows

$$f_\tau = \tau_{R_0}^2 + \tau_{R_w}^2, \tag{1.216}$$

where the characteristic time-scale τ_{R_0} is given by ν_t/k, and τ_{R_w} is the wall-reflection time-scale. The other model constants and functions are indicated in Table 1.1.

1.5.2 Explicit Algebraic Heat Flux Models

To overcome the limitations of models based on the Simple Gradient Diffusion Hypotheses, we introduce the Explicit Algebraic Heat Flux Models (EAHFM). The model presented in the following is the model described by Hattori in [46] that is based on previous models introduced by Abe et al. in 1996 [52] and Nagano and Hattori in 2003 [51].

To derive the transport equation for the turbulent heat flux, the i-th fluctuating velocity component is multiplied by the equation for the instantaneous temperature (1.45), and the result is added to the i-th component of the instantaneous Navier-Stokes equation (1.44) multiplied by the fluctuating

C_μ	C_D	C_η	C_{v1}	C_{v2}	C_g	C_λ	τ_{mg}
0.12	0.8	5.0	0.4	1.0×10^2	-0.7	0.1	$C_\lambda f_\lambda \frac{k}{\varepsilon}$

f_λ	$f_w(\xi)$
$[1 - f_w(25)]\left\{1 + \sqrt{\frac{2R}{Pr}}\frac{15}{R_t^{3/4}} \exp\left[-\left(\frac{R_{tm}}{30}\right)^{3/4}\right]\right\}$	$\exp\left[-\left(\frac{R_{tm}}{\xi}\right)^2\right]$

τ_{R_0}	τ_{R_w}	f_B
$\frac{v_t}{k}$	$\sqrt{\frac{1}{6}\frac{f_{R1}/C_D}{f_{S\Omega}}}\left(1 - \frac{3C_{v1}f_{v2}}{8}\right) f_{v1}^2$	$1 + C_\eta \left(C_D \tau_{R_0}\right)^2 \left(\Omega^2 - S^2\right)$

$f_{S\Omega}$	S_f
$\frac{\Omega^2}{2} + \frac{S^2}{3} - \left[\left(\sqrt{\frac{S^2}{3}} - \sqrt{\frac{\Omega^2}{2}}\right) f_w(1)\right]^2$	$1 + \sqrt{S^2}\tau_u$

f_{v1}	f_{v2}	R_{tm}
$\exp\left[-\frac{(R_{tm}/52)^2 S_f}{(R_{tm}/52)^2 + S_f}\right]$	$1 - \exp\left(-\frac{\sqrt{R_t}}{C_{v2}}\right)$	$\frac{130 R_d R_t^{1/4}}{130 R_t^{1/4} + R_d}$

Table 1.1: Model constants and functions for EASM as in [46].

temperature T'. The resulting expression is then time-averaged

$$\begin{aligned}\frac{D\langle u_j' T'\rangle}{Dt} = &- \left(\langle u_k' T'\rangle \frac{\partial \langle u_j\rangle}{\partial x_k} + \langle u_j' u_k'\rangle \frac{\partial \langle T\rangle}{\partial x_k}\right) + \\ &- (\alpha + \nu)\langle \frac{\partial T'}{\partial x_k}\frac{\partial u_j'}{\partial x_k}\rangle - \frac{\partial}{\partial x_k}\left(\langle u_k' u_j' T'\rangle + \frac{\langle p' T'\rangle}{\rho}\delta_{jk}\right) + \\ &- \alpha \langle u_j' \frac{\partial T'}{\partial x_k}\rangle - \nu \langle T' \frac{\partial u_j'}{\partial x_k}\rangle + \frac{\langle p' \frac{\partial T'}{\partial x_j}\rangle}{\rho} - \beta g_j \langle T'^2\rangle.\end{aligned} \tag{1.217}$$

or in its compact form as

$$\frac{D\langle u_j' T'\rangle}{Dt} = P_{j\theta} - \varepsilon_{j\theta} + D_{j\theta} + \Phi_{j\theta} + G_{j\theta}, \tag{1.218}$$

where $P_{j\theta}$ is a production term, $\varepsilon_{j\theta}$ is a dissipation term, $D_{j\theta}$ is a diffusion term including the turbulent and molecular transport term, $\Phi_{j\theta}$ is a pressure–temperature gradient correlation and $G_{j\theta}$ is a buoyant term. We recall the equation for the temperature variance in its compact form as

$$\frac{Dk_\theta}{Dt} = P_{k_\theta} - \varepsilon_{k_\theta} + D_{k_\theta}\,. \tag{1.219}$$

Let the normalized heat flux a_j^* be equal to $\langle u_j' T' \rangle / \sqrt{k k_\theta}$, then we apply a similar procedure used for the normalized stress tensor in the previous section and we obtain the spatial and temporal variations in a_j^* as

$$\begin{aligned} \frac{D a_j^*}{Dt} = & \frac{1}{\sqrt{k k_\theta}} \left(P_{j\theta} + \Phi_{j\theta} - \varepsilon_{j\theta} + G_{j\theta} \right) + \\ & - \frac{1}{2k} a_j^* \left(P_k + G_k - \varepsilon \right) - \frac{1}{2k_\theta} a_j^* \left(P_{k_\theta} - \varepsilon_\theta \right) , \end{aligned} \tag{1.220}$$

where the diffusive effect $D_{j\theta}$ is neglected. Here, we have used the definition of the transport equations for the turbulent heat flux (1.218), for the turbulent kinetic energy (1.187), and for the temperature variance (1.219). In the local equilibrium state, the relation introduced by Abe et al. in [52] holds

$$\frac{D a_j^*}{Dt} = 0. \tag{1.221}$$

Thus, we can obtain the following equation

$$\frac{1}{\sqrt{k k_\theta}} \left(P_{j\theta} + \Pi_{j\theta} + G_{j\theta} \right) = \frac{a_j^*}{2} \left[\frac{1}{k} \left(P_k + G_k - \varepsilon \right) + \frac{1}{k_\theta} \left(P_{k_\theta} - \varepsilon_\theta \right) \right] , \tag{1.222}$$

where $\Pi_{j\theta} = \Phi_{j\theta} - \varepsilon_{j\theta}$. As regards this latter term, the general linear expression introduced by Launder [53] is employed considering also the buoyant effect proposed by Craft et al. in [54]. Therefore, it is modeled as follows

$$\begin{aligned} \Pi_{j\theta} = & -C_{t1} \frac{\langle u_j' T' \rangle}{\tau_u} + C_{t2} \langle u_k' T' \rangle \frac{\partial \langle u_j \rangle}{\partial x_k} + C_{t3} \langle u_k' T' \rangle \frac{\partial \langle u_k \rangle}{\partial x_j} + \\ & + C_{t5} g_j \beta k_\theta - C_{t4} \langle u_i' u_j' \rangle A_{ik} \frac{\partial \langle T \rangle}{\partial x_k} - C_{t6} A_{jk} g_k \beta k_\theta , \end{aligned} \tag{1.223}$$

where $A_{ik} = \langle u_i' u_k' \rangle / k$ is the non-dimensional Reynolds stress tensor. The term $\partial \langle T \rangle / \partial x_k$ is included because the production of $\varepsilon_{j\theta}$ is significantly influenced by the temperature gradient. We recall the definition of $P_{j\theta}$ and $G_{j\theta}$ as

$$P_{j\theta} = -\langle u_k' T' \rangle \frac{\partial \langle u_j \rangle}{\partial x_k} - \langle u_j' u_k' \rangle \frac{\partial \langle T \rangle}{\partial x_k}, \; G_{j\theta} = \beta g_j \langle T'^2 \rangle = 2 \beta g_j k_\theta \, . \tag{1.224}$$

By replacing the expression (1.223) and (1.224) into the Equation (1.222),

the following equation can be formulated

$$\begin{aligned}\frac{1}{\sqrt{kk_\theta}}\frac{1}{C_{\theta1}\tau_u}\Bigg[&-\langle u_j'T'\rangle - C_{\theta1}\tau_u\langle u_j'u_k'\rangle\frac{\partial\langle T\rangle}{\partial x_k} - C_{\theta2}\tau_u\langle u_k'T'\rangle S_{jk}+\\&- C_{\theta3}\tau_u\langle u_k'T'\rangle\Omega_{jk} - C_{\theta4}\tau_u\langle u_i'u_j'\rangle A_{ik}\frac{\partial\langle T\rangle}{\partial x_k} - C_{\theta5}\tau_u g_j\beta k_\theta+\\&- C_{\theta6}\tau_u A_{jk}g_k\beta k_\theta\Bigg] = -\frac{1}{2}\frac{\langle u_j'T'\rangle}{\sqrt{kk_\theta}}\left[\frac{\varepsilon}{k}\left(\frac{P_k}{\varepsilon}+\frac{G_k}{\varepsilon}-1\right)\right]+\\&+\frac{\varepsilon_\theta}{k_\theta}\left(\frac{P_{k_\theta}}{\varepsilon_\theta}-1\right),\end{aligned} \tag{1.225}$$

where $C_{\theta1} = 1/C_{t1}$, $C_{\theta2} = (1 - C_{t2} - C_{t3})/C_{t1}$, $C_{\theta3} = (1 - C_{t2} + C_{t3})/C_{t1}$, $C_{\theta4} = C_{t4}/C_{t1}$, $C_{\theta5} = (2 - C_{t5})/C_{t1}$, $C_{\theta6} = C_{t6}/C_{t1}$, are model constants. Substituting the decomposition of the Reynolds stress tensor in its isotropic and deviatoric parts and applying some algebraic manipulations, we obtain the following equation

$$\begin{aligned}&\frac{\langle u_j'T'\rangle}{\sqrt{kk_\theta}}\frac{1}{\tau_u}\left[1+\frac{C_{\theta1}}{2}\left(\frac{P_k}{\varepsilon}+\frac{G_k}{\varepsilon}-1\right)+\frac{C_{\theta1}}{2R}\left(\frac{P_{k_\theta}}{\varepsilon_\theta}-1\right)\right] =\\&= -\left(\delta_{jk}+3b_{jk}\right)\frac{2}{3}\frac{C_{\theta1}k}{\sqrt{kk_\theta}}\frac{\partial\langle T\rangle}{\partial x_k} - \left(C_{\theta2}S_{jk}+C_{\theta3}\Omega_{jk}\right)\frac{\langle u_k'T'\rangle}{\sqrt{kk_\theta}}+\\&- \left(\delta_{ij}+3b_{ij}\right)\frac{2}{3}\frac{kC_{\theta4}}{\sqrt{kk_\theta}}A_{ik}\frac{\partial\langle T\rangle}{\partial x_k} - \frac{C_{\theta5}}{\sqrt{kk_\theta}}g_j\beta k_\theta - \frac{C_{\theta6}}{\sqrt{kk_\theta}}A_{jk}g_k\beta k_\theta.\end{aligned} \tag{1.226}$$

We introduce the following non-dimensional quantities

$$b_{jk}^* = 3b_{jk}, \tag{1.227}$$

$$\Theta_k^* = \frac{2}{3}\frac{C_{\theta1}k\tau_m}{\sqrt{k}\sqrt{k_\theta}}\frac{\partial\langle T\rangle}{\partial x_k}, \tag{1.228}$$

$$S_{jk}^* = C_{\theta2}\tau_m S_{jk}, \tag{1.229}$$

$$\Omega_{jk}^* = C_{\theta3}\tau_m\Omega_{jk}, \tag{1.230}$$

$$T_k^* = \frac{2}{3}\frac{C_{\theta4}k\tau_m}{\sqrt{k}\sqrt{k_\theta}}\frac{\partial\langle T\rangle}{\partial x_k}, \tag{1.231}$$

$$G_j^* = \frac{2}{3}\frac{C_{\theta5}\tau_m}{\sqrt{k}\sqrt{k_\theta}}g_j\beta k_\theta, \tag{1.232}$$

$$F_k^* = \frac{2}{3}\frac{C_{\theta6}\tau_m}{\sqrt{k}\sqrt{k_\theta}}g_k\beta k_\theta, \tag{1.233}$$

used to obtain the following non-dimensional equation

$$\begin{aligned} a_j^* = &- \left(\delta_{jk} + b_{jk}^*\right) \Theta_k^* - a_k^* \left(S_{jk}^* + \Omega_{jk}^*\right) + \\ &- \left(\delta_{ij} + b_{ij}^*\right) A_{ik} T_k^* - G_j^* - A_{jk} F_k^*, \end{aligned} \tag{1.234}$$

which represents the implicit form of the Algebraic Stress Models in non-dimensional form since the normalized turbulent heat flux appears on both sides of the equation. The explicit form in a_j^* can be derived from (1.234) as

$$\begin{aligned} a_j^* = F \Big\{ &\left[- \left(\delta_{jk} + b_{jk}^*\right) + \left(\delta_{\ell k} + b_{\ell k}^*\right) \left(S_{j\ell}^* + \Omega_{j\ell}^*\right) \right] \Theta_k^* + \\ &+ \left[- \left(\delta_{jk} + b_{jk}^*\right) A_{ik} + \left(\delta_{\ell i} + b_{\ell i}^*\right) \left(S_{j\ell}^* + \Omega_{j\ell}^*\right) A_{ik} \right] T_k^* + \\ &- \left[\delta_{jk} - \left(S_{jk}^* + \Omega_{jk}^*\right) \right] G_k^* - \left[\delta_{j\ell} - \left(S_{j\ell}^* + \Omega_{j\ell}^*\right) \right] A_{\ell k} F_k^* \Big\}. \end{aligned} \tag{1.235}$$

where

$$F = \frac{1}{1 + \frac{1}{2}\left(\Omega^{*2} - S^{*2}\right)}, \Omega^{*2} = \Omega_{ij}^* \Omega_{ij}^*, \; S^{*2} = S_{ij}^* S_{ij}^* \, . \tag{1.236}$$

The dimensional form of Equation (1.235) is then derived

C_{tm}	R_{tm}	$f_w(\xi)$	$C_{\theta 1}$	$C_{\theta 2}$
1.3×10^2	$\frac{C_{tm} R_d R_t^{1/4}}{C_{tm} R_t^{1/4} + R_d}$	$\exp\left[- \left(\frac{R_{tm}}{\xi}\right)^2 \right]$	$0.14\left[1 - f_w(40)\right]$	0.05
$C_{\theta 3}$	$C_{\theta 4}$	$C_{\theta 5}$	$C_{\theta 6}$	
0.11	0.3	$0.3\left[1 - f_w(40)\right]^2$	$0.2\left[1 - f_w(20)\right]$	
$C_{\theta 7}$	C_m	$B_{\lambda 1}$	τ_u	τ_θ	R
$0.2\left[1 - f_w(20)\right]$	$0.25/Pr^{1/4}$	$\frac{1+2Pr}{20Pr^{0.4}}$	$\frac{k}{\varepsilon}$	$\frac{k_\theta}{\varepsilon_\theta}$	$\frac{\tau_\theta}{\tau_u}$
τ_m					
$\left\{ \frac{2R}{R + C_m} + \sqrt{\frac{2R}{Pr}} \frac{30}{R_t^{3/4}} \exp\left[- \left(\frac{R_{tm}}{B_{\lambda 1}}\right)^{3/4} \right] \right\} \tau_u$					

Table 1.2: Model constants and functions for EAHFM as in [46].

$$\begin{aligned}\langle u_j' T' \rangle = &-\alpha_{jk}^t \frac{\partial \langle T \rangle}{\partial x_k} + \frac{\tau_m^2}{f_{RT}} \left(C_{\theta 1} \langle u_\ell' u_k' \rangle + C_{\theta 5} \langle u_\ell' u_i' \rangle A_{ik} \right) \left(C_{\theta 2} S_{j\ell} + \right. \\ &\left. + C_{\theta 3} \Omega_{j\ell} \right) \frac{\partial \langle T \rangle}{\partial x_k} - \frac{2 C_{\theta 6} \tau_m g_k \beta k_\theta}{f_{RT}} \left[\delta_{jk} - \tau_m \left(C_{\theta 2} S_{jk} + C_{\theta 3} \Omega_{jk} \right) \right] \\ &- \frac{2 C_{\theta 7} \tau_m A_{\ell k} g_k \beta k_\theta}{f_{RT}} \left[\delta_{j\ell} - \tau_m \left(C_{\theta 2} S_{j\ell} + C_{\theta 3} \Omega_{j\ell} \right) \right] ,\end{aligned} \tag{1.237}$$

The anisotropic eddy diffusivity is given as

$$\alpha_{jk}^t = \frac{\tau_m}{f_{RT}} \left(C_{\theta 1} \overline{u_j u_k} + C_{\theta 4} \overline{u_i u_j} A_{ik} \right) , \tag{1.238}$$

where

$$f_{RT} = 1 + \frac{1}{2} \left\{ \tau_m \left[1 - f_w(40) \right] \right\}^2 \left(C_{\theta 3} \Omega^2 - C_{\theta 2} S^2 \right) . \tag{1.239}$$

The other model constants and functions are indicated in Table 1.2.

1.6 The Anisotropic Four-parameter Turbulence Model

This section provides the overview of the equations implemented in the turbulent solver of FEMuS code for simulating both forced and natural convection phenomena. This solver is based on the previously introduced anisotropic Reynolds stress tensor and anisotropic turbulent heat flux, overcoming the limitations of the Boussinesq hypothesis of eddy viscosity and diffusivity.

The governing equation for the generic turbulent problem in RANS form is

$$\frac{\partial \langle u_i \rangle}{\partial x_i} = 0 , \tag{1.240}$$

$$\frac{D \langle u_i \rangle}{Dt} = -\frac{1}{\rho} \frac{\partial \langle p \rangle}{\partial x_i} + \frac{\partial}{\partial x_j} \left[\nu \left(\frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial \langle u_j \rangle}{\partial x_i} \right) - \langle u_i' u_j' \rangle \right] - g_i \beta \langle T \rangle , \tag{1.241}$$

$$\frac{D \langle T \rangle}{Dt} = \frac{\partial}{\partial x_i} \left(\alpha \frac{\partial \langle T \rangle}{\partial x_i} - \langle u_i' T' \rangle \right) + \frac{Q}{\rho c} . \tag{1.242}$$

In this work, the Reynolds stress tensor and the turbulent heat flux have been implemented using the EASM and EAHFM models as presented in the Section 1.5 and reported here for clarity. These models consider the buoyancy

term added to treat the natural convection problem as

$$\begin{aligned}\langle u_i' u_j' \rangle = &\frac{2}{3}k\delta_{ij} - \frac{2\nu_t}{f_{R1}}S_{ij} - \frac{4C_D k f_\tau}{f_{R1}}\left(S_{ik}\Omega_{kj} - \Omega_{ik}S_{kj}\right) + \\ &+ \frac{4C_D k f_\tau}{f_{R1}}\left(S_{ik}S_{kj} - \frac{1}{3}S_{mn}S_{mn}\delta_{ij}\right) + \\ &- \frac{2C_g\tau_{mg}}{C_D f_{R2}}f_{ij} - \frac{4C_g\tau_{mg}^2}{f_{R2}}\left(f_{ik}\Omega_{kj} - \Omega_{ik}f_{kj}\right) + \\ &+ \frac{2C_g\tau_{mg}^2}{f_{R2}}\left(f_{ik}S_{kj} + S_{ik}f_{kj} - \frac{2}{3}f_{mn}S_{mn}\delta_{ij}\right),\end{aligned} \tag{1.243}$$

while the turbulent heat flux uses

$$\begin{aligned}\langle u_j' T' \rangle = &-\alpha_{jk}^t\frac{\partial\langle T\rangle}{\partial x_k} + \frac{\tau_m^2}{f_{RT}}\left(C_{\theta1}\langle u_\ell' u_k'\rangle + C_{\theta5}\langle u_\ell' u_i'\rangle A_{ik}\right)\left(C_{\theta2}S_{j\ell} + \right. \\ &\left. + C_{\theta3}\Omega_{j\ell}\right)\frac{\partial\langle T\rangle}{\partial x_k} - \frac{2C_{\theta6}\tau_m g_k\beta k_\theta}{f_{RT}}\left[\delta_{jk} - \tau_m\left(C_{\theta2}S_{jk} + C_{\theta3}\Omega_{jk}\right)\right] \\ &- \frac{2C_{\theta7}\tau_m A_{\ell k}g_k\beta k_\theta}{f_{RT}}\left[\delta_{j\ell} - \tau_m\left(C_{\theta2}S_{j\ell} + C_{\theta3}\Omega_{j\ell}\right)\right].\end{aligned} \tag{1.244}$$

The closure of the turbulent model requires the determination of k and ω, along with the turbulent viscosity ν_t and its characteristic dynamic time scale τ_{lu} for the dynamic turbulence. Additionally, the thermal turbulence requires k_θ and ω_θ, the turbulent thermal diffusivity α_t, and its associated mixing time scale τ_m. By solving the transport equations for the turbulent variables, we can determine the turbulent kinetic energy, its specific dissipation, the temperature variance, and its specific dissipation. In this thesis, the four-parameter model employed is a logarithmic version of the standard models already included in FEMuS code [55, 56, 57]. The use of logarithmic variables, denoted as $K-\Omega$ and $K_\theta-\Omega_\theta$, improves the stability of the standard models by ensuring that the state variables remain positive throughout the solution process [58]. The dynamic logarithmic variables are defined as follows

$$K = \ln(k), \quad \Omega = \ln(\omega), \tag{1.245}$$

and their corresponding transport equations derived from (1.120) and (1.131) are given by

$$\frac{DK}{Dt} = \frac{\partial}{\partial x_i}\left[\left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial K}{\partial x_i}\right] + \left(\nu + \frac{\nu_t}{\sigma_k}\right)\frac{\partial K\partial K}{\partial x_i\partial x_i} + \frac{P_k}{e^K} + \frac{G_k}{e^K} - c_\mu e^\Omega, \tag{1.246}$$

$$\frac{D\Omega}{Dt} = \frac{\partial}{\partial x_i}\left[\left(\nu + \frac{\nu_t}{\sigma_\omega}\right)\frac{\partial \Omega}{\partial x_i}\right] + \left(\nu + \frac{\nu_t}{\sigma_\omega}\right)\frac{\partial \Omega \partial \Omega}{\partial x_i \partial x_i} + \frac{P_k}{e^K}\left(c_{\varepsilon 1} - 1\right) + \\ + 2\left(\nu + \frac{\nu_t}{\sigma_\omega}\right)\frac{\partial K \partial \Omega}{\partial x_i \partial x_i} + \left(c_b - 1\right)\frac{G_k}{e^K} - c_\mu\left(c_{\varepsilon 2} f_\varepsilon - 1\right)e^{\Omega}, \tag{1.247}$$

where the production rate of the turbulent kinetic energy is defined as $P_k = -\langle u_i' u_k' \rangle \partial \langle u_i \rangle / \partial x_k$ and the buoyancy production term is $G_k = -\beta g_i \langle u_i' T' \rangle$. The model constants and the model functions are detailed in (1.111) and (1.114).

As regards the thermal turbulent variables, we can introduce logarithmic quantities as

$$K_\theta = \ln(k_\theta), \quad \Omega_\theta = \ln(\omega_\theta), \tag{1.248}$$

and the system of equations is obtained substituting (1.248) in Equations (1.173) and (1.174) as

$$\frac{DK_\theta}{Dt} = \frac{\partial}{\partial x_i}\left[\left(\alpha + \frac{\alpha_t}{\sigma_{k\theta}}\right)\frac{\partial K_\theta}{\partial x_i}\right] + \left(\alpha + \frac{\alpha_t}{\sigma_{k\theta}}\right)\frac{\partial K_\theta}{\partial x_i}\frac{\partial K_\theta}{\partial x_i} + \\ + P_{k_\theta} e^{-K_\theta} - c_\mu e^{\Omega_\theta}, \tag{1.249}$$

$$\frac{D\Omega_\theta}{Dt} = \frac{\partial}{\partial x_i}\left[\left(\alpha + \frac{\alpha_t}{\sigma_{\omega\theta}}\right)\frac{\partial \Omega_\theta}{\partial x_i}\right] + \left(\alpha + \frac{\alpha_t}{\sigma_{\omega\theta}}\right)\frac{\partial \Omega_\theta}{\partial x_i}\frac{\partial \Omega_\theta}{\partial x_i} + \\ + 2\left(\alpha + \frac{\alpha_t}{\sigma_{\omega\theta}}\right)\frac{\partial K_\theta}{\partial x_i}\frac{\partial \Omega_\theta}{\partial x_i} + P_{k_\theta} e^{-K_\theta}\left(c_{p1} - 1\right) + \\ - \left(c_{d1} - 1\right) c_\mu e^{\Omega_\theta} - c_{d2} c_\mu e^{\Omega_\theta} + c_{p2} P_k e^{-K}, \tag{1.250}$$

where the thermal and mechanical production production terms are $P_{k_\theta} = -\langle u_j' T' \rangle \partial \langle T \rangle / \partial x_j$ and $P_k = -\langle u_i' u_k' \rangle \partial \langle u_i \rangle / \partial x_k$. The model constants and the model functions are defined as in [57]

$$c_{d2} = \left\{1.9\left[1 - 0.3\exp\left(-\frac{R_t^2}{42.25}\right)\right] - 1\right\}\left[1 - \exp\left(-\frac{R_d^2}{25}\right)\right], \tag{1.251}$$

$$c_{p1} = 1.025,\ c_{p2} = 0.9,\ c_{d1} = 1.1,\ \sigma_{k\theta} = \sigma_{\omega\theta} = 1.4. \tag{1.252}$$

The eddy viscosity has been computed using the following expression

$$\nu_t = C_\mu k \tau_{lu}\,, \tag{1.253}$$

where τ_{lu} has been defined according to [25] as

$$\tau_{lu} = \tau_u\left[1 - \exp\left(-\frac{R_d}{14}\right)\right]^2\left[1 + \exp\left(-2.5 \times 10^{-5} R_t^2\right)\frac{5}{R_t^{3/4}}\right]. \tag{1.254}$$

Similarly, the eddy diffusivity is modeled as

$$\alpha_t = C_t k \tau_m, \tag{1.255}$$

where the mixing time scale is given by

$$\tau_m = \tau_u f_{1\theta} \left(\frac{1}{Pr_t} + \frac{2R}{R + C_\gamma} f_{2\theta} + 1.3 \frac{\sqrt{2R}}{Pr R_t^{3/4}} f_{3\theta} \right), \tag{1.256}$$

where $C_\gamma = 0.25/Pr_t^{1/4}$ and the functions $f_{1\theta}$, $f_{2\theta}$ and $f_{3\theta}$ are defined as in (1.183) and (1.184). All the other model constants and functions used in Equations (1.243) and (1.244) have been modeled accordingly to [46] and are reported in Table 1.1 and 1.2.

Chapter 2

Numerical Code Coupling

In the past few years, the study of multiscale and multiphysics problems has emerged as a topic of intensive research. Multiscale modeling involves simulating processes at different spatial and temporal scales. Multiphysics modeling combines different physical models to simulate interactions between various phenomena. For instance, complex systems such as nuclear reactor plants, where neutronic, thermohydraulic, and thermomechanics are strongly coupled, or heat exchangers, where analyzing thermal performances requires the Conjugate Heat Transfer (CHT) problem to integrate fluid dynamics with solid heat conduction.

Evaluation of the whole system requires a modeling effort for all scales and interactions among system components. Thus, the development of numerical tools has to account for phenomena with multiple scales and physics [59, 60, 61, 62, 63, 64].

Nowadays, CFD techniques are an integral part of the design process of most engineering systems, even the more complex ones. Moreover, High-Performance Computing (HPC) significantly enhances multiphysics and multiscale CFD simulations by providing the necessary computational power to handle large calculations and datasets. Despite all the advancements in this field, simulating highly complex systems remains a significant challenge. Indeed, no single code can handle the full range of physical interactions involved in any given phenomenon. Existing commercial codes generally cater to a

specific problem type. For instance, we can find a plethora of open-source simulation software that can tackle a subset of the physical systems. We can refer to codes such as OpenFOAM [65], TrioCFD [66], and code_Saturne [67] for fluidynamical simulations. These codes solve incompressible and compressible Navier-Stokes equations with multi-phase flow and turbulence phenomena (at different scales, through RANS, LES, or DNS). Several solvers are available in the thermomechanical field, including elasticity problems and fracture propagation. Among them, we can mention Code-Aster [68] or TFEL/MFront [69]. Codes such as Dragon/Donjon have been developed to tackle neutronic problems. Other open-source FEM-based numerical platforms such as libMesh [70], Deal-II [71], and FEniCS [72] are widely used to solve generic PDE problems.

Two main strategies have been explored over the years to simulate multiscale and multiphysics problems. One way is to develop a new numerical code to model all the relevant physical phenomena. This strategy is commonly defined as the monolithic approach or the direct method. Alternatively, one can choose to couple existing and validated codes for each phenomenon of interest. The main advantage of the monolithic approach is that it can lead to a more stable and accurate solution. Since all physical interactions are fully integrated into the system of equations, the feedback between subsystems is captured more accurately, with less risk of introducing errors or instabilities from decoupling. However, the formulation is generally more difficult because all physical phenomena must be considered simultaneously, often requiring specialized software or adaptations to existing solvers. In contrast, the partitioned (or coupled) method keeps the individual solvers. The idea is to solve each subsystem independently and exchange information (typically at each time step). This approach is advantageous when reliable solvers for each subsystem already exist, as it allows existing codebases to use strategies designed for those subsystems. The partitioned approach can also enable parallel computations, where different subsystems are solved concurrently, improving computational efficiency. This method requires efficient data exchange between codes, especially in the HPC framework. Therefore, output and input exchange data on code coupling are performed directly in memory to avoid writing and reading processing from external files [73].

In this thesis, the development of the coupling application aims to improve the in-house numerical platform [74] for enabling multiphysics and multiscale simulations. The proposed platform models several physical PDEs (Partial

Differential Equations) using both FEM (finite element method) and FVM (finite volume method) for modeling fluids and solids [75]. By integrating the open source software, OpenFOAM [65] and the in-house library FEMuS [76], the numerical platform aims to leverage their strengths and their simulation capabilities. In this Chapter, the code coupling strategy is presented, and its implementation is discussed. This Chapter introduces the computational framework in which the code coupling application has been developed. Then, it presents the coupling strategy and the numerical algorithm implemented, concluding with a discussion of two numerical examples of code coupling between FEMuS and OpenFOAM, one involving the exchange of volumetric fields and the other coupling boundary conditions. Numerical results are provided and compared with literature data of the same physical problems performed with the monolithic approach.

2.1 Numerical Platform Environment

A numerical platform has been developed over the years at the Laboratory of Montecuccolino of the University of Bologna to improve portability and communication among in-house numerical codes [74]. The idea behind the numerical platform is to create a numerical environment for multiphysics and multiscale simulations, integrating different physical PDE models with FEM and FVM discretizations for fluids and solids.

This numerical platform provides several environments for different user levels: one for implementing engineering applications and another suitable for developing in-house code. These environments include the data entry for "input/output" using CAD and Mesh generators and visualization or post-processing with tools typical of the SALOME computing platform [77], i.e., the software Paraview [78]. Implementations of PDE models on the FE and FV discretizations are available from two standalone codes: the in-house FEM solver named FEMuS [76] and the open-source software OpenFOAM [65]. The Message Passing Interface (MPI) manages parallel computations for both CFD codes. MPI is a standardized and portable protocol designed for programming with parallel computing architectures [79, 80]. In particular, both CFD codes use the Open Message Passing Interface (OpenMPI), an open-source implementation of MPI developed and maintained by a consortium of academic, research, and industry partners [81]. For the solution of partial differential equations and, in particular, for the solution of linear

systems, FEMuS uses a library called PETSc, which stands for the "Portable, Extensible Toolkit for Scientific Computation" [82]. PETSc is a suite of data structures and routines developed by Argonne National Laboratory, designed for the scalable (parallel) solution of scientific applications modeled by partial differential equations. On the other hand, OpenFOAM relies on its in-house matrix and solver libraries to handle the numerical solution of PDEs rather than an external library like PETSc. However, it has recently introduced the option to integrate PETSc for these computations [83].

2.1.1 Strategies for Code Integration

Two main strategies can be adopted to integrate multiple numerical codes within this numerical platform. The first strategy involves implementing specific procedures to couple the new code with each existing platform library. Alternatively, a new library can be added to the coupling framework by developing a single wrapper that maps its data fields to a common coupling format. In this case, heterogeneous fields with different data structures from various numerical codes are exchanged through a shared intermediate representation. These two approaches are referred to as the point-to-point and hub-and-spoke models, respectively. A schematic example of the two coupling approaches is reported in Figure 2.1. Point-to-point integration pro-

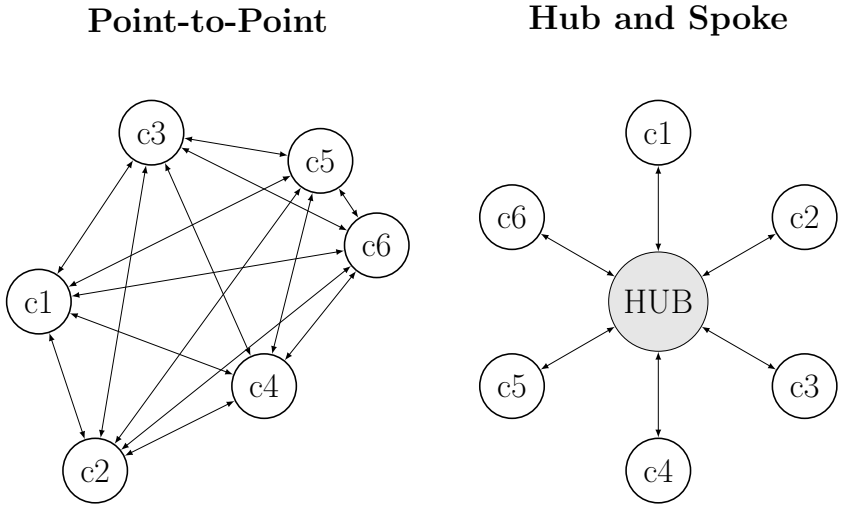

Figure 2.1: Coupling strategy models: point-to-point on the left and hub-and-spoke on the right.

vides direct communication and data exchange between two endpoints. This

approach does not need an intermediary between the subsystems, ensuring continuous interaction and optimized data interchanging mechanism. On the other hand, hub-and-spoke integration follows a different approach by eliminating the need for direct connections between every system involved in data sharing. Each endpoint is connected to a centralized *hub*, which manages all data exchanges between senders and receivers. Point-to-point communication is inherently limited in scalability and requires significant efforts to work with large systems with several integrated software. As the number of connected subsystems grows, the complexity increases exponentially, making the approach less efficient. In contrast, the hub-and-spoke model provides a more efficient alternative to the point-to-point model, requiring fewer connection routes. More particularly, in a network with n endpoints, the hub-and-spoke model requires only $n-1$ routes to connect all nodes, setting the upper bound of $n-1$ connections and a complexity of $O(n)$. This approach is significantly more efficient than the $n(n-1)/2$ routes, or $O(n^2)$, needed for direct connections between every pair of nodes in a point-to-point network.

Among the intermediary representations, MED (Model for Exchange Data) libraries can be adopted as a *hub* intermediary in the hub-and-spoke model. This open-source library provides a standard coupling format (the MED format) for the numerical codes integrated into the aforementioned platform. This library is a module of the SALOME platform for retrieving, processing, and sharing data at the memory level, avoiding external disk files. The SALOME platform and its modules are further detailed in the following section.

SALOME Platform

SALOME is a numerical platform developed by CEA (Commissariat à l'énergie atomique et aux énergies alternatives) and EDF (Électricité de France) to provide open-source software for computer-aided engineering (CAE) [84]. This platform offers several modules, such as GUI, Geometry, Mesh, Fields, YACS, JobManager, and ParaViS, that may manage every stage of a computational simulation performed by multiple external standalone codes. The software implements tools for parametric CAD modeling, tetrahedral and hexahedral mesh generation, code supervision, post-processing, and data analysis [77]. In particular, the supervisor can generate simulation workflows that connect different computational units, with the support of the FIELDS and MED libraries. This controls data communication by manipulating input and output simulations. It can also perform data transfer and

analysis.

A brief description of the main functionalities may offer a clear understanding of the different aspects and capabilities and explain how each feature contributes to the overall workflow in addressing different tasks and requirements. SALOME is equipped with a parametric and variational CAD modeler called SHAPER. This modeler allows users to define parametric dimensions and constraints for a sketch, enabling the automatic update of the final shapes when the CAD element parameters are modified. With this feature, designers can automatically create and update groups, regions, and domains when geometry is modified. Additionally, it allows the creation of non-manifold geometries (e.g., an edge shared by more than two faces) and the assembly of multi-dimensional parts to create a complex geometry. SMESH module represents the SALOME mesh module, providing several discretization algorithms for finite element and finite volume methods. This module gives the possibility to exploit SALOME meshing tools (quadrangles, hexahedra, boundary-layer meshing, etc.), open-source meshing tools (NETGEN and Gmsh), and commercial meshing tools from the MeshGems suite. Multiple types of meshes, generated using various available meshing algorithms, can be applied to different regions. To simplify the mesh creation process, several transformation options are provided, such as rotation, symmetry, and scaling, as well as mesh optimization and local refinement. The mesh object can be equipped with groups and labels to recognize the geometry of different regions, including the boundary elements. The SALOME platform includes a module called PARAVIS for the visualization and analysis of numerical results. This module uses ParaView functionalities, an open-ource software for interactive scientific visualization. The PARAVIS module offers various types of data representations and a wide range of filters customized for effective data visualization.

Several functionalities provided by SALOME, which are strictly connected to multi-physics simulations, are managed by the supervisor. The supervisor's main responsibility is to create a simulation workflow by integrating domain-specific solvers through the YACSGEN interface, simplifying the coupling of different physical domains. Within the coupling framework, SALOME provides modules for field manipulation called FIELDS. The FIELDS module for field manipulation consists of a graphical interface designed to implement standard use cases of field manipulation, along with a library of functions for processing data on meshes and fields based on the MED

model. Among its features, the most relevant for this work are reading/writing from/to files, the aggregation and exchange of data, interpolation between different computational grids, format conversion, and renumbering or partitioning of data for multiprocessor frameworks.

The XMED Library from the SALOME Platform. The FIELDS module for field manipulation is called XMED. It consists of a small set of atomic library files. Its structure is depicted in Figure 2.2, where the architecture layer is reported. In particular, each element depends on the blocks

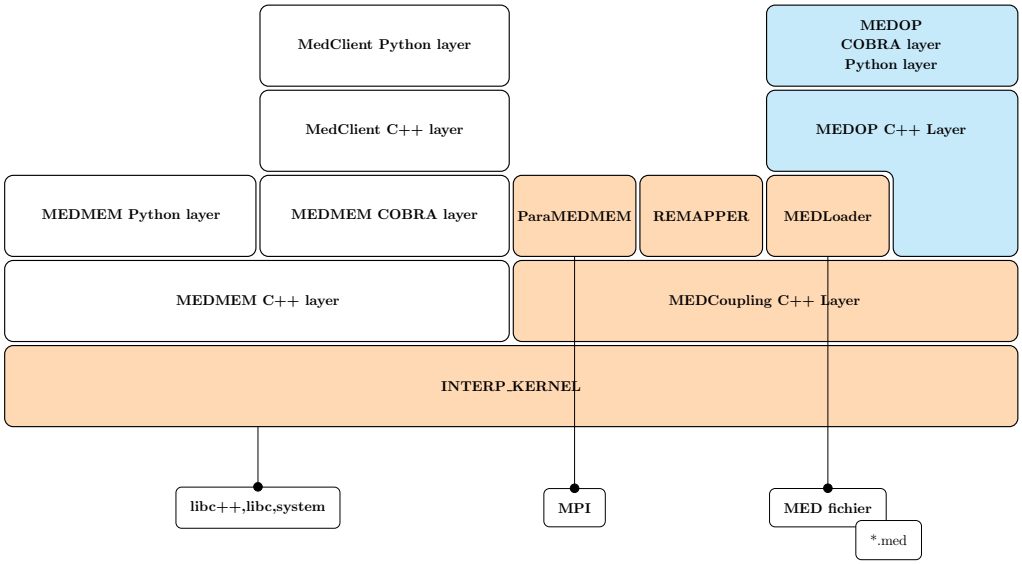

Figure 2.2: Packages structure of the XMED library in FIELDS module.

it covers (fully or partially) along the vertical line. The core of the MED module is the MEDMEM library represented by the orange blocks, while the white blocks represent the old deprecated version of the MEDMEM library. The blue packages represent the additional components for field manipulation through the user interface (TUI and GUI). It is built around the MED format, which is a standardized format for storing and exchanging numerical data in scientific computing. It can be used in both C++ and Python environments. The library provides data structures (C++ classes) for meshes or fields and various functionalities for data manipulation. It also includes routines for reading and writing MED files. MEDMEM library is mainly designed for advanced data processing, such as interpolation and localization of fields or projecting data between different meshes. The MEDMEM

library manages these functionalities through its submodules, each designed for specific tasks. The MEDCoupling package handles the data structures for meshes and fields and data manipulation routines. The MEDLoader module provides I/O operations for the MED file format. INTERP_KERNEL and REMAPPER modules provide the mathematical frameworks and algorithms for data exchange, focusing on interpolation between different numerical domains. In the following, we will refer only by the name MED to the MED library or the MEDMEM module.

2.2 OpenFOAM and FEMuS Integration

Thanks to its efficiency, the hub-and-spoke approach represents the most effective strategy for coupling multiple codes and supporting multiscale and multiphysics simulations. Accordingly, in this thesis, the FEMuS and OpenFOAM codes have been coupled into the numerical platform using the MED library as a common intermediate representation for exchanging heterogeneous data structures.

Figure 2.3 represents the simplified scheme of the hub-and-spoke model adopted as a code coupling framework. The diagram outlines the coupling system between two computational codes, identified as *Code 1* and *Code 2*, interacting under the direction of a *Supervisor*. It is worth noting that the approach described in this scheme is not limited to two codes but can be extended to integrate multiple coupled codes. In this context, the terms *Code 1* and *Code 2* represent FEMuS and OpenFOAM. The *Supervisor* object is represented as a main program used to manage the structures of both codes and the coupling between them. In particular, it is responsible for managing the instantiation and initialization of *Code 1*, *Code 2*, and *Coupling* objects class. The *Supervisor* controls the progression of numerical simulations over time and coordinates the integration process over shared or overlapping time intervals. In addition to code-related functionalities, it is also responsible for synchronizing the data transfer between the two codes, ensuring a consistent exchange of information throughout the simulation process.

Each code internally manages its solver, *Solver 1* and *Solver 2*, which operate on their respective datasets and solve their respective physical problems. However, in the context of a coupled application, both codes may require physical quantities solved by the other subsystems. These specific quantities are represented in the scheme of Figure 2.3 as $\Phi_{s,*}$ and $\Phi_{t,*}$, where the

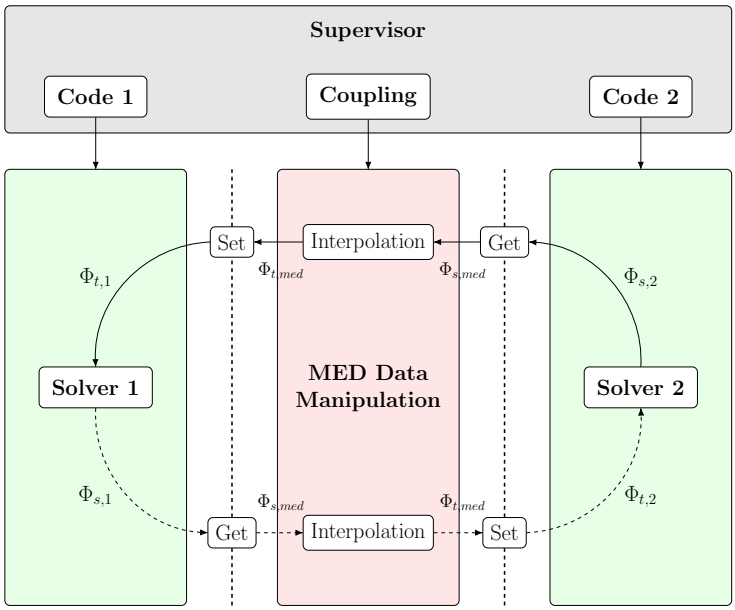

Figure 2.3: Coupling procedure scheme.

subscripts t and s refer to "target" and "source", respectively. These fields are data structures specific to each code, which may differ depending on the coupled application. For instance, FEMuS uses PETSc data structures to store the solution of the problem equation, whereas OpenFOAM employs its own data structures, such as GeometricField, dimensionedField, and others. Code interfaces are explicitly developed to ensure compatibility between two types of data structures. These interfaces are implemented as C++ classes with multiple functionalities, including converting data from specific data structures to the MED format and vice versa. In Figure 2.3, these interfaces are represented by dashed lines connecting the green blocks (code libraries) to the red one. The latter block represents the C++ class developed for collecting routines from the MEDMEM module. It acts as the intermediary for data exchange between *Code 1* and *Code 2*. In particular, given the fields from *Code 1*, this class projects the corresponding MED field, $\Phi_{s,\text{med}}$, defined over a source MED mesh, onto the target MED mesh to compute the target MED field ($\Phi_{t,\text{med}}$). The interface of *Code 2* receives this field in MED format and converts it into *Code 2* data structure.

As just mentioned, data flows between the two codes have been possible by implementing three classes. The first class, named `foamMED`, is the intermediary link between OpenFOAM and the MED library, covering the role

of the OpenFOAM-MED interface. Similarly, the second class, `femusMED`, is the interface between FEMuS and the MED library. Finally, the third class, namely `MEDclass`, is responsible for managing operations within the MED library itself.

2.2.1 MED Communication Class

In this section, we explore the development of the `MEDclass`, which is the core of the coupling applications discussed in the following chapters. In particular, a brief overview of the features implemented in this class is provided.

MED Class Structure

The MED library provides a comprehensive set of routines for creating meshes, initializing fields, and managing data transfer between meshes. It supports a wide range of element types categorized by their geometric dimensionality and shape. These include line segments for 1D meshes, triangles, quadrilaterals, and polygons for 2D meshes, and tetrahedra, hexahedra, prisms, pyramids, and polyhedra for 3D meshes. The library can also create fields with different dimensionalities and discretizations. It supports scalar, vector, and tensor fields, which can be associated with various types of mesh entities, such as nodes (nodal fields) or cells (cell-centered fields). Node fields are classified as P_1 or P_2, corresponding to linear or quadratic discretizations. Cell-centered fields are labeled P_0. Moreover, the MED library can handle field interpolation from one mesh (source mesh) to another (target mesh) while preserving physical consistency in the built-in projection method.

The C++ class `MEDclass` has been created to employ all these functionalities, and a brief code snippet is provided below to illustrate its structure:

```
1 class MEDclass {
2  private:
3   MPI_Comm _comm;
4   vector<MEDCoupling::MEDCouplingUMesh*> _mesh;
5   vector<MEDCoupling::MEDCouplingFieldDouble*> _field;
6   vector<MEDCoupling::DataArrayDouble*> _array_tmp;
7   vector<MEDCoupling::MEDCouplingRemapper*> _remapper;
8   vector<c_type> _element_type;
9  public:
```

```
  map<string, int> _map_field_name;
  map<string, int> _map_mesh_name;
  map<int, int> _map_field_mesh;

  int problem_dimension;

public:
  MEDclass() = default;
  ~MEDclass() = default;

  MEDclass(MPI_Comm comm) : _comm{comm} {}

  //beginFunctions
  ...
  //endFunctions
}
```

This class includes all the MED structures for storing data, i.e., meshes, fields, interpolation matrices, as well as structures designed to facilitate information retrieval at the supervisor level, such as `_map_field_name` and `_map_mesh_name`. The current version of the `MEDclass` includes support for an MPI communicator. However, it does not yet handle distributed data exchange between processes. This functionality should be implemented for future developments to support efficient coupling in high-performance computing environments.

To complete this class, the following data structures are defined:

```
enum cell_type {
  quad9 = INTERP_KERNEL::NORM_QUAD9,
  quad4 = INTERP_KERNEL::NORM_QUAD4,
  seg3 = INTERP_KERNEL::NORM_SEG3,
  seg2 = INTERP_KERNEL::NORM_SEG2,
  hexa8 = INTERP_KERNEL::NORM_HEXA8,
  polyhed = INTERP_KERNEL::NORM_POLYHED,
  ...
};
struct c_type{
  cell_type type;
  int dof;
```

```
13   int dimension;
14 };
```

These structures are used to group information about the MED mesh, such as the type of cell elements (`cell_type`), the degree of freedom (DOF) of the elements (`dof`) and the mesh dimension. The `cell_type` structure includes a variety of element types supported by the `MEDclass`, including two-dimensional quadratic elements such as `NORM_QUAD9`, commonly used in the FEMuS code, as well as three-dimensional elements typically found in OpenFOAM meshes, such as hexahedral (`NORM_HEXA8`) and polyhedral (`NORM_POLYHED`) cells.

Main Functionalities

In the coupling procedure of this Thesis, one of the first step is to define computational meshes in MED format over which the MED fields can be stored. The routine responsible for managing the creation of unstructured meshes is called `create_mesh()`. The creation of the MED mesh requires information about connectivity, coordinate nodes, discretization type, and mesh dimension. The vector `vector<MEDCoupling::MEDCouplingUMesh*> _mesh` groups the mesh copies, with the number of copies corresponding to the interfaces required by the coupling application.

The `MEDclass` provides the possibility to define several entities, `_field`, of type `MEDCouplingFieldDouble`, stored in a C++ vector, which correspond to the coupled fields involved. Two distinct functions have been developed to create MED field data structures responsible for storing field information from the coupled codes. These routines are named `init_med_field_on_nodes()` and `init_med_field_on_cells()`. The former creates and initializes a MED field over the mesh nodes, which can be linear or bi-quadratic. The latter function performs a similar initialization but defines a field over cells instead of nodes. During the solution process, the `set_field()` routine updates the field values. It uses a MED data structure (`_array_tmp`), where code solutions are temporarily stored. Then, it copies the fields from the `DataArrayDouble` to the `MEDCouplingFieldDouble` structure. The inverse process, named `get_field()`, is used to retrieve MED field information and transfer it to the intermediate data structure of type `DataArrayDouble`.

The other main functionality implemented in `MEDclass` is the interpolation mechanism. The interpolation of a source field over a target mesh is performed by constructing the projection operator $\mathbf{P}$ to compute the follow-

ing matrix operation

$$\Phi_t = \mathbf{P}\Phi_s \,, \tag{2.1}$$

where Φ_t and Φ_s are the target and the source fields, respectively. Therefore, the interpolation routine consists of two main steps: the construction of the operator $\mathbf{P}$, named `_remapper`, and the projection of the source field onto a target field. The operator $\mathbf{P}$ is a matrix with dimensionality $n \times m$, where n is the number of nodes/cells in the target mesh and m is the number of nodes/cells in the source mesh. This matrix is constructed through algorithms that can locate the nodes or cells of the source mesh within the elements of the target mesh. The creation of $\mathbf{P}$, implemented in the `get_remapper()` routine, is managed by the REMAPPER class of the MED library through the `prepare()` function, which is available within the library. Once the operator has been built, the field projection step in (2.1) is executed in the `interpolate_field()` routine using the built-in MED function named `transferField()`. The interpolation routine is available for both P_0 and P_1 fields and the algorithms can combine different field types, e.g. it is possible to interpolate from P_0 to P_0, from P_0 to P_1, from P_1 to P_0, and from P_1 to P_1. More details about these routines are provided in Appendix A.1.

MED library provides several methods to construct the matrix $\mathbf{P}$ according to the type of source and target meshes. The simpler interpolation mechanism occurs when the two meshes are perfectly overlapping. Point-to-point interpolation transfers field values from nodes in one mesh to corresponding nodes in another mesh. This is a straightforward method used when the two meshes have the same topology and point locations, allowing a direct transfer of values. Similarly, in cell-to-cell interpolation, the values of a field defined on the cells of one mesh are transferred to the corresponding elements of another mesh.

When the mesh nodes or cells are aligned, the interpolation can be exact, with the projection operator taking the form of a diagonal matrix. However, this approach is not appropriate for meshes with different geometries or element distributions, which typically require more complex projection methods. To address such cases, two MED built-in algorithms were employed for the interpolation of fields in this thesis: the *Triangulation* algorithm and the *Geometric2D* algorithm. The former decomposes each cell into triangles and computes triangle-triangle intersections by determining segment crossings and node inclusions. The latter is very flexible because it supports any type of polygon, including linear, quadratic, convex polygons, and non-convex

polygons. This projection method can also handle interpolations between edges.

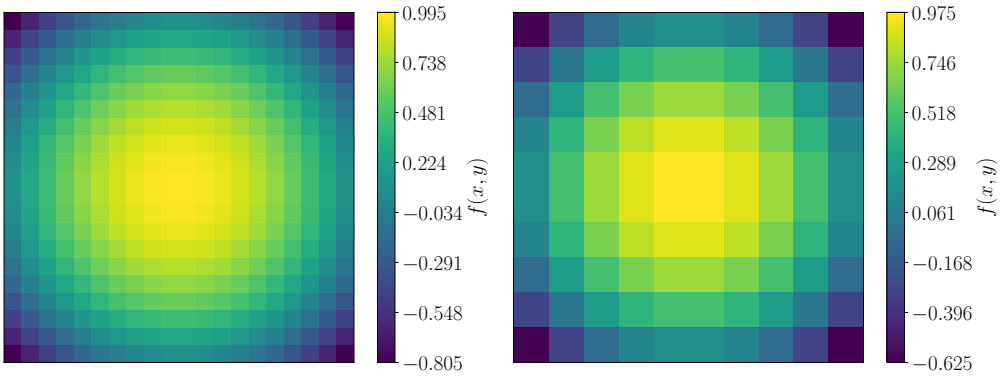

Figure 2.4: $P_0 - P_0$ interpolation from the piece-wise representation of the analytical function $f(x, y) = 1 - (x - 1)^2 + (y - 1)^2$ over a finer mesh to the piece-wise representation on a coarser one.

When referring to non-overlapping meshes, we may consider differences such as element types (e.g., triangle or quadrilateral elements), discretization methods (e.g., piecewise or linear discretization), resolution, or simply variations in point or cell locations. Differences in resolution can be classified as either upscaling or downscaling, depending on whether data are transferred from a coarse mesh to a fine mesh or from a fine mesh to a coarse mesh, respectively. An example of downscaling interpolation obtained by exploiting `MEDclass` routines is shown in Figures 2.4, where the MED field of the finer source mesh is interpolated onto the coarser target mesh. In this specific case, the source field is the P_0 representation of the analytic function $f(x, y) = 1 - (x - 1)^2 + (y - 1)^2$. It has been initialized on a 20×20 mesh with QUAD4 elements. The target mesh consists of a 10×10 computational grid.

Figure 2.5 illustrates another example of $P_0 - P_0$ interpolation for nonoverlapping meshes. The two meshes are shifted, with the overlapping region limited to the domain defined by the target mesh lower left corner and the source mesh upper right corner. The interpolations performed include upscaling for the case shown on the left and downscaling for the case shown on the right. The interpolated source field is a piecewise representation of the analytic function $f(x, y) = \sin \pi x \sin \pi y$.

`MEDclass` has been developed to handle also interpolation for mixed-

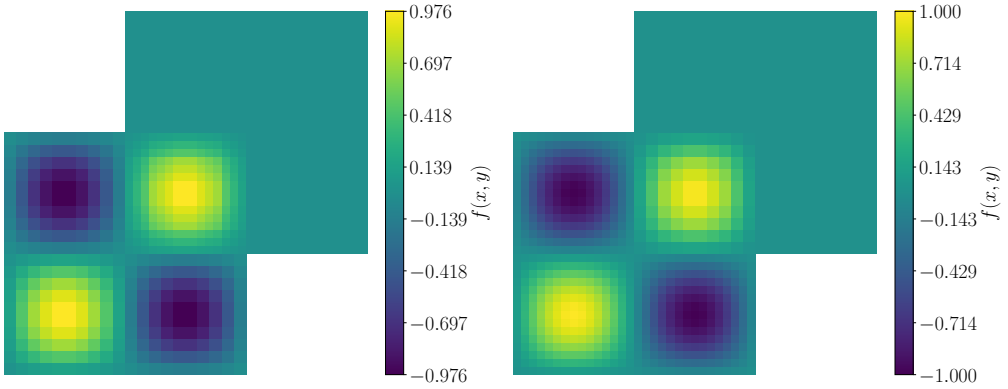

Figure 2.5: $P_0 - P_0$ interpolation for partially overlapped meshes. Interpolation of the piece-wise representation of the analytical function $f(x, y) = \sin \pi x \sin \pi y$ from the coarser to the finer mesh on the left and from the finer to the coarser mesh on the right.

dimension meshes, where different regions of the domain are discretized with different dimensions. Dimensional interpolation is the process of mapping field values between meshes with different dimensions, such as from 3D to 2D, 2D to 1D, or viceversa. A field defined over a 3D volume mesh (e.g.,

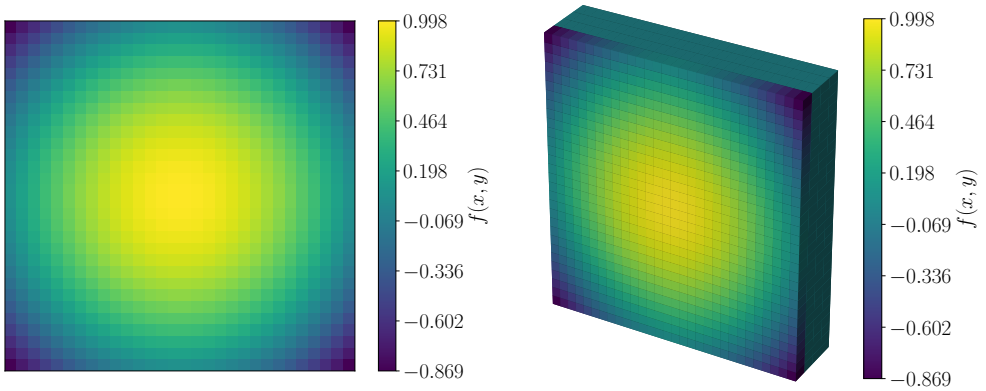

Figure 2.6: $P_0 - P_0$ interpolation from a 2D representation of the analytical function $f(x, y) = 1 - (x - 1)^2 + (y - 1)^2$ to a 3D mesh.

temperature, pressure, or velocity) often needs to be transferred to a 2D surface mesh. This scenario is common when information from a 3D fluid domain must be projected onto a two-dimensional boundary. In the applications presented in this thesis, such an interpolation occurs when dealing with OpenFOAM to FEMuS codes. OpenFOAM can handle only 3D simulations,

even for 2D problems, while FEMuS operates with 2D meshes. As a result, the interpolation required in this case involves transferring data from a 3D mesh to a 2D mesh. The inverse interpolation function has to be applied when fields from FEMuS 2D meshes is projected onto OpenFOAM 3D meshes. Figures 2.6 and 2.7 provide examples of multidimensional interpolation managed using `MEDclass` functionalities. The former shows the 2D to 3D interpolation of a piecewise field representing the function $f(x, y) = 1 - (x-1)^2 + (y-1)^2$. The latter report the target field of the 1D to 2D interpolation of the same function in its 1D form, $f(x) = 1 - (x-1)^2$.

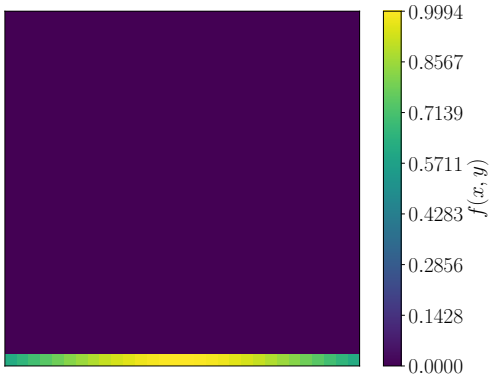

Figure 2.7: $P_0 - P_0$ interpolation from a 1D representation of the analytical function $f(x) = 1 - (x-1)^2$ to a 2D mesh.

2.2.2 FEMuS Interface Class

The FEMuS code includes solvers for a wide range of physical problems, such as incompressible Navier-Stokes equations, heat transfer, turbulence models, fluid-structure interaction problems, multiphase flows, and optimal control. One notable advantage of FEMuS is its flexibility in implementing new models by directly integrating the constitutive equations into the FEM paradigm. This approach allows for the efficient modeling of novel physical phenomena, such as the complex dynamics of turbulent flows in unconventional fluids like liquid metals. For further details on the FEMuS library, readers are referred to [85, 86, 76, 87, 88].

The FEMuS library has been integrated into the coupling application of the numerical platform by extending it with a MED-compatible C++ interface, named `femusMED`.

femusMED Class Structure

The `femusMED` class is built to manage the interaction between the FEMuS framework and the MED library within a coupled CFD simulation environment. Its primary aim is to provide all the functionalities for transferring the mesh-related data to the MED environment. The C++ class is defined as follows:

```
class femusMED {
  private:
    MPI_Comm _comm;
    int probl_dimension = 2;
    std::vector<interfaces_femus> interface;
  public:
    femusMED() = default;
    ~femusMED() = default;
    femusMED(MPI_Comm comm) : _comm{comm} {}

  //beginFunctions
    ...
  //endFunctions
}
```

This class creates `interfaces_femus` structures, which are data collections for creating the interface between FEMuS and MED format:

```
struct interfaces_femus{
  vector<int> _conn;
  vector<double> _coords;
  vector<vector<int>> _indices;
  vector<double> _field_val;
  map<int,int> _map_med2mg;
  map<int,int> _map_mg2med;
  int _n_nodes;
  string _mesh_name;
  ...
};
```

These structures store mesh data, such as connectivity information, mesh coordinates, and mappings between the FEMuS and MED node indices. The

interface also holds field values to be transferred along with the reference of the corresponding indexes to enabling the coupling calculation. Each part of the domain that needs to be coupled externally has its own interface structure, allowing the management of more than one interface at a time.

Main Functionalities

The main feature of the `femusMED` class deals with initializing and managing the interface structure. The routine developed for this purpose is named `init_interface()`. During initialization, the interfaces collect node connectivity (`vector<int> _conn`) and coordinate (`vector<int> _coords`) information from the FEMuS mesh object using the `set_mesh_connectivity()` and `set_mesh_coordinates()` functions. Moreover, `init_interface()` gathers all the data necessary for subsequent steps to streamline and accelerate the coupling process. This includes assigning a unique name to the mesh for easy information retrieval (`string _mesh_name`), determining the number of DOFs (`_n_nodes`), and other essential parameters. This data is then transferred to the MED class to create the mesh entity in MED format, as described in Section 2.2.1.

To ensure compatibility with another solver in the MED frameworks, `femusMED` class also provides methods to handle mesh conversions. The developed routine `convert_to_linear_mesh()` provides information for creating a linear MED mesh starting from biquadratic mesh connectivity and coordinates information. The class can also extract information to create boundary interfaces. The overloaded routines `set_mesh_connectivity()` and `set_mesh_coordinates()` are used to establish the local connectivity of the boundary mesh from the original FEMuS mesh and to map the local renumbered nodes to the global index numbering of the mesh.

The class `femusMED` is also responsible for getting and setting the MED field from/to the FEMuS structure. The `get_field_from_femus()` routine is employed to extract a field from FEMuS as a PETSc vector, which is the structure FEMuS uses to store the solver field solutions. The retrieved field can be copied to the temporary MED data structures, making it accessible to the `MEDclass`. Since the FEMuS code uses a P_2 representation for the solution fields, the interface class has been extended to perform a $P_2 - P_0$ interpolation. This projection has been extensively described in [89], and the corresponding functionality has been incorporated into the developed class.

A similar approach transfers the MED field to the FEMuS code structures

through the `set_field_to_femus()` routine. This routine has been developed to write MED fields into PETSc data structures to update the FEMuS solver fields. In this process, it may be necessary to calculate a quadratic field from the cell-centered field provided by the MED framework. An interpolation routine for transforming a P_0 field into a P_2 field has been included as in [89]. More details about these class methods are provided in Appendix A.1.

2.2.3 OpenFOAM Interface Class

OpenFOAM is a well-known, open-source, object-oriented C++ library developed primarily for CFD simulations, maintained separately by ESI (Engineering System International) Group and the OpenFOAM Foundation [65]. Its versatility, scalability, and extensive set of solvers and libraries make it one of the most widely used codes in industry and academia. OpenFOAM provides a comprehensive modeling platform for complex fluid dynamics scenarios, such as multiphase flow, aerodynamics, turbulence, and heat transfer phenomena.

In this thesis, the interface class has been created for version 11 of the OpenFOAM Foundation distribution. Unlike other versions and distributions, it introduces a significant change to its CFD toolbox by implementing modular solvers. Solvers are now written as modules, e.g., *incompressibleFluid*, *incompressibleVoF*, and *solid* module, and they are compiled as a separate library. These modules, written as classes, offer more flexibility than the traditional application-based solvers that have been part of OpenFOAM since 1993. This version also introduces a generalized application called `foamRun`, that loads and runs the modular solvers, providing a standardized way to call the solver routines. This approach does not require switching applications for each specific case. The OpenFOAM Foundation version v11 introduces a `foamMultiRun` application for more complex simulations. It enables the use of different equations in separate mesh regions, making it particularly useful for coupled problems.

In this thesis, a C++ wrapper for the OpenFOAM simulation environment has been developed to provide a structured and object-oriented interface to the `foamRun` and `foamMultiRun` solvers. The architecture of this wrapper is based on inheritance, consisting of a base class and two derived (child) classes. The two child classes extend the base class for the single and multi-region OpenFOAM solvers.

foamMED Class Structure

The base `foamMED` class has been developed as the interface between the OpenFOAM framework and the MED library:

```
class foamMED {
  private:
    int probl_dimension = 3;
    vector<interfaces_foam> interface;
    Foam::fvMesh * mesh;
    string problem_path;
  public:
    foamMED() = default;
    explicit foamMED(string_view path) : problem_path(path){}
    ~foamMED() = default;

    //beginFunctions
        ...
    //endFunctions
};
```

The core data exchange happens through the `interfaces_foam` structure, which stores information such as field values, node coordinates, connectivity details, and mappings between MED and OpenFOAM node indices. As in `femusMED` class, the following structure allows the interface class to translate OpenFOAM's internal data into a format compatible with the MED library:

```
struct interfaces_foam{
  vector<vector<int>> _indices;
  vector<double> _field_val;
  map<int,int> _map_med2of;
  map<int,int> _map_of2med;
  vector<mcIdType> _conn;
  vector<double> _coords;
  int _n_nodes;
  string _mesh_name;
};
```

The main features of this class are similar to the main features of the FEMuS interface. OpenFOAM interface class uses `init_interface()` routine to set

up the domain interface information such as node connectivity and coordinates. The `set_mesh_connectivity()` and `set_mesh_coordinates()` functions are used to retrieve mesh information from `mesh` object. The get and set routines, `get_field_from_of()` and `set_field_to_of()`, are employed to manage OpenFOAM fields converting built-in OpenFOAM data structures to MED format fields and vice versa.

The derived classes exploit the two generalized application `foamRun` and `foamMultiRun` to execute the OpenFOAM problem. The `foamSingleProblem` class refers to the single-region application `foamRun`, while the class named `foamMultiProblem` refers to the multi-region solver. The classes are as:

```
class foamSingleProblem : public foamMED
{
  private:
    std::unique_ptr<Foam::Time> _runTime;
    std::unique_ptr<Foam::solver> _solver;
    std::unique_ptr<Foam::pimpleSingleRegionControl> _pimple;
    Foam::word solverName;

  public:
    foamSingleProblem(std::string path) : foamMED(path) {};
    ~foamSingleProblem() = default;

    //beginFunctions
    void init(int argc, char * argv[]);
    void init_mesh();
    void init_solver();
    void init_pimple_control();
    bool run();
    void pre_solve();
    void solve();
    void post_solve();
    void write();
    void setup_dt();
    void adjust_dt();
    //...
    //endFunctions
};
```

```
class foamMultiProblem : public foamMED{
  private:
    std::unique_ptr<Foam::Time> _runTime;
    std::unique_ptr<Foam::regionSolvers> _solver;
    std::unique_ptr<Foam::pimpleMultiRegionControl> _pimple;

  public:
    foamMultiProblem(std::string path) : foamMED(path) {};
    ~foamMultiProblem() = default;

    //beginFunctions
    void init(int argc, char * argv[]);
    void set_mesh(Foam::word region_name);
    bool run();
    void pre_solve();
    void solve();
    void post_solve();
    void write();
    void setup_dt();
    void adjust_dt();
    //...
    //endFunctions
};
```

More details about the routines are provided in Appendix A.1.

2.3 Coupling Algorithm

As outlined in previous sections, numerical models based on single solvers are conventionally developed to address a single physical problem. In such solutions, interactions with other phenomena usually rely on strong hypotheses. In this context, the partitioned approach overcomes the lack of integration between physical phenomena to allow a more realistic representation of complex systems. This strategy systematically integrates different solvers based on exchanging data through iterative coupling algorithms. A generic scenario involving two independent codes, each designed to simulate a specific phenomenon, is schematically illustrated in Figure 2.8. In this setup, the solvers, referred to as *Solver 1* and *Solver 2*, receive inputs named $x_{1,i}$ and

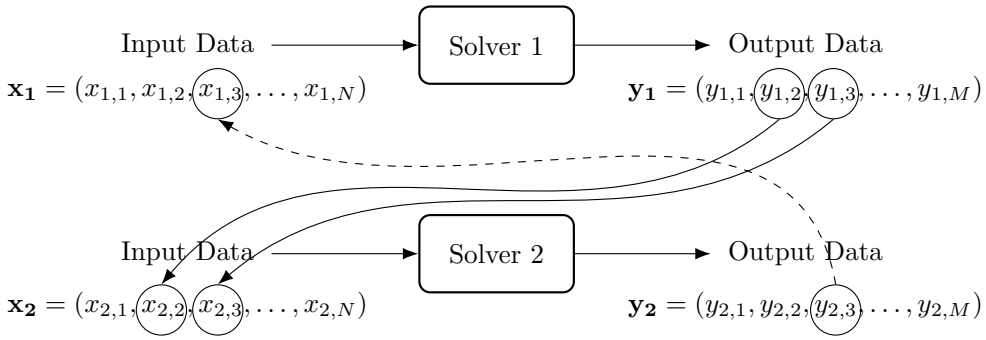

Figure 2.8: Schematic representation of weak (solid lines) and strong (solid and dashed lines) coupling between *Solver 1* and *Solver 2*.

$x_{2,i}$, respectively, where $i = 1, ..., N$ with N being the total input variables, and return $y_{1,j}$ and $y_{2,j}$ outputs, where $j = 1, ..., M$ with M being the total output variables. The interactions between the two solvers can take on two distinct forms. The simplest form of coupling is called *weak coupling*, *semi-coupling*, *partial coupling*, or *loose coupling*. Here, *Solver 1*, referred to as the feeder, produces output data that serves as input for *Solver 2*, known as the consumer [90]. This coupling approach is illustrated in Figure 2.8 using solid lines connecting the feeder subsystem's output to the consumer subsystem's input. During the execution of a single time step, the output of *Solver 1* directly passes to *Solver 2* input. It is worth noting that, in loose coupling, the consumer output has no impact on the feeder solver input data, reflecting a unidirectional flow of information. Due to its inherent nature, this approach may necessitate subcycling or midpoint correction at synchronization time steps. Although weak coupling is relatively simple to implement, its use is limited since the consumer solver output directly influences the feeder solver input. In contrast to semi-coupling, a second approach is known as strong coupling (called "full coupling" or "tight coupling"). Strong coupling also includes a feedback loop where the output of *Solver 2* is connected to the input of *Solver 1*, represented in Figure 2.8 by a dashed line. In this way, both solvers are interdependent because each consumer solver relies on the output of the previous one. For this reason, the two solvers must be executed sequentially, and neither solver can be run simultaneously.

In the coupling problem context, input variables are classified into dependent and independent categories. The former are influenced by the output

of a feeder solver and are considered interface-matching unknowns. The primary challenge in addressing *strong coupling* problems lies in determining the values of the dependent variables through a coupling algorithm. The coupling framework developed in this work can handle both the weak and strong coupling types. We focus on describing the two main methods for solving *strong coupling* problems, as *tight coupling* represents the most interesting application and is likely the most valuable for realistic simulations.

Given a first subsystem with input parameters $\mathbf{x}_1$, its evolution over time is described by the following differential equation

$$\dot{\mathbf{x}}_1 = f_1(\mathbf{x}_1, \mathbf{x}_2), \tag{2.2}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are elements of function spaces, $\mathcal{H}_1$ and $\mathcal{H}_2$, respectively, and f_1 is a differential operator. As a result, equation (2.2) describes subsystems governed by partial differential equations. In this first subsystem, the input variables $\mathbf{x}_2$ are not explicitly defined and represent the dependent parameters for the coupling interface with the second subsystem, which is similarly described by the following partial differential equation

$$\dot{\mathbf{x}}_2 = f_2(\mathbf{x}_2, \mathbf{x}_1). \tag{2.3}$$

The system of equations (2.2) and (2.3), which is the differential evolution system for $(\mathbf{x}_1, \mathbf{x}_2) \in \mathcal{H}_1 \times \mathcal{H}_2$, can be referred to as pure differential coupling. Consider applying a time discretization method to both equations (2.2) and (2.3), assuming, for simplicity, that the time step is the same and equal to Δt. When they do not share the same time step, one must employ sub-cycling since the temporal interval between synchronization points, Δt, consists of multiple smaller steps. Let $\mathbf{x}_j^{(n)}$, $(j = 1, 2)$ represents the discrete approximation of the solutions to (2.2) and (2.3) at time step n. The integration algorithms, as illustrated in Figure 2.8, can be expressed as follows

$$\mathbf{x}_1^{(n)} = \varphi_1(\mathbf{x}_1^{(n)}, \mathbf{x}_1^{(n-1)}, \mathbf{y}_2), \tag{2.4}$$

$$\mathbf{x}_2^{(n)} = \varphi_2(\mathbf{x}_2^{(n)}, \mathbf{x}_2^{(n-1)}, \mathbf{y}_1). \tag{2.5}$$

Here, the time dependence is omitted for simplicity of notation. The unknown time-dependent functions, $\mathbf{y}_j(t)$, $(j = 1, 2)$, represent the dependent variables provided by the other solver over the time interval $\Delta t = t_n - t_{n-1}$. To set the dependent input variables $\mathbf{y}_j(t)$, $(j = 1, 2)$, several numerical methods relying on iterative procedures for exchanging inputs and outputs have been developed over the years. For example, the Block Jacobi (BJ) coupling process

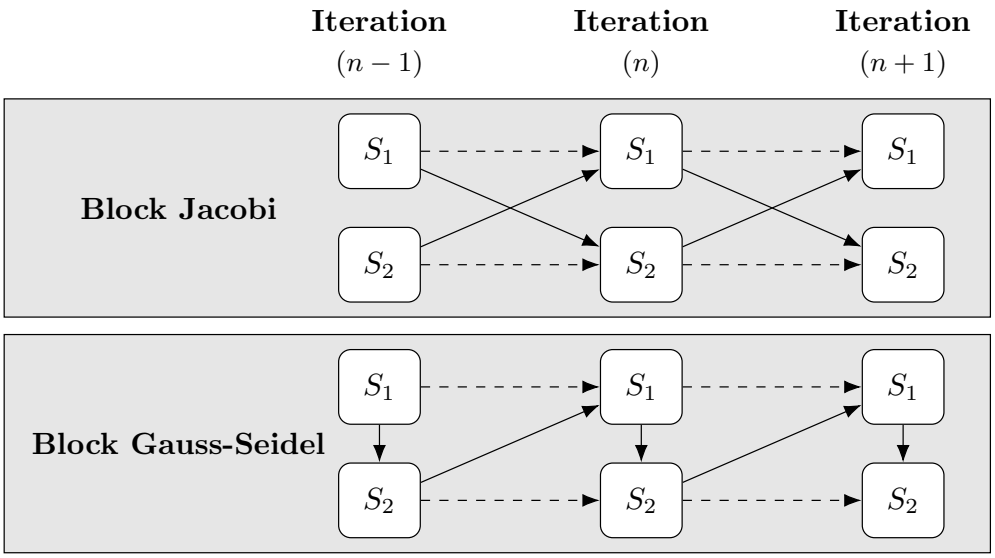

Figure 2.9: Coupling algorithm strategies for the strongly coupled problem.

is the most conceptually simple iterative approach to achieve strong coupling between two or more subsystems [91]. Given the function $\Psi_j(\mathbf{x}_j^{(n-1)})$, $(j = 1, 2)$, which depends on the previous values of the approximate solution, the BJ approach assumes that the value at t_{n-1} remains constant at each iteration interval. In particular, $\mathbf{y}_j = \Psi_j(\mathbf{x}_j^{(n-1)}) \equiv \mathbf{x}_j^{(n-1)}$, $(j = 1, 2)$. This approximation results in the following system of equations for the two subsystems

$$\mathbf{x}_1^{(n)} = \varphi_1(\mathbf{x}_1^{(n)}, \mathbf{x}_1^{(n-1)}, \Psi_2(\mathbf{x}_2^{(n-1)})) = \varphi_1(\mathbf{x}_1^{(n)}, \mathbf{x}_1^{(n-1)}, \mathbf{x}_2^{(n-1)}), \tag{2.6}$$

$$\mathbf{x}_2^{(n)} = \varphi_2(\mathbf{x}_2^{(n)}, \mathbf{x}_2^{(n-1)}, \Psi_1(\mathbf{x}_1^{(n-1)})) = \varphi_2(\mathbf{x}_2^{(n)}, \mathbf{x}_2^{(n-1)}, \mathbf{x}_1^{(n-1)}). \tag{2.7}$$

At time step n, the BJ method uses the solution from the previous time step of the other subsystem as input-dependent parameters. This approach is illustrated in Figure 2.9 (top), where the dashed lines represent the solution process of each specific solver advancing from step $(n-1)$, providing the previous solutions $\mathbf{x}_j^{(n-1)}$, $(j = 1, 2)$ for time step n. Meanwhile, the solid lines represent the coupled dependent parameters computed at the time step $n-1$ provided by the other solver, $\mathbf{x}_j^{(n-1)}$, $(j = 2, 1)$. Both codes solve their problem at the time step n using old solutions. The system solvers have an associated critical time step Δt_{cj}, $(j = 1, 2)$. By coupling the systems, the critical time step for the global system is $\min \Delta t_{cj}$.

Starting from (2.6) and (2.7) equations, the Gauss-Seidel iterative approach can be formulated

$$\mathbf{x}_1^{(n)} = \varphi_1(\mathbf{x}_1^{(n)}, \mathbf{x}_1^{(n-1)}, \Psi_2(\mathbf{x}_2^{(n-1)})) = \varphi_1(\mathbf{x}_1^{(n)}, \mathbf{x}_1^{(n-1)}, \mathbf{x}_2^{(n-1)}), \tag{2.8}$$

$$\mathbf{x}_2^{(n)} = \varphi_2(\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n-1)}, \Psi_1(\mathbf{x}_1^{(n-1)})) = \varphi_2(\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n-1)}, \mathbf{x}_1^{(n)}), \tag{2.9}$$

where the new value $\mathbf{x}_1^{(n)} = \Psi_1(\mathbf{x}_1^{(n-1)}) := \varphi_1(\mathbf{x}_1^{(n-1)}, \mathbf{x}_2^{(n-1)})$ is used in the same iteration to compute solution of *Solver 2*. This partly implicit approach converges better than Block Jacobi's method because it uses information about the dependent parameters as soon as they become available. The Block Gauss-Seidel strategy for strong coupling is shown in Figure 2.9 (bottom). In the first step of the algorithm, all independent parameters and an initial estimate value for all dependent parameters are entered into the two solvers. For given dependent and independent input parameters, *Solver 1* delivers an output that becomes a new input for *Solver 2*. The procedure is repeated until the difference between two subsequent iterations satisfies the proposed convergence criteria.

2.3.1 Algorithm Routines

This section explains the coupling approach implemented between FEMuS and OpenFOAM, referred to as *Code 1* and *Code 2*. This implementation can easily be generalized to additional software with simple modifications, mainly by translating the internal data structures into the MED format. Figure 2.3 offers a concise overview of the main functionalities of the coupling environment. The Algorithm 1, instead, provides a detailed explanation of the coupling process used in this thesis and offers a more comprehensive description of the coupling process, illustrating how the routines outlined in previous sections are implemented using the Block Gauss-Seidel algorithm. This framework can be exploited in various coupling simulations between volume or boundary fields. For simulations requiring volume field transfer, the application allows the exchange of numerical data representing physical quantities distributed in the whole computational domain. Simulations involving boundary data transfer focus on the interaction between different physical domains or interfaces within the computational domain. By transferring boundary conditions, forces, or constraints between *Code 1* and *Code 2*, it is possible to simulate complex fluid-structure interactions, conjugate heat transfer processes, and multi-phase flow phenomena.

Algorithm 1 Code setup for a coupling algorithm.

```
1: procedure SUPERVISOR CONTROL
2:     Instantiate code class objects code1_obj and code2_obj
3:     Instantiate MED class object MED_obj
4:     Initialize interfaces
5:     if not moving_mesh then
6:         Create projection matrix P
7:     end if
       Time loop
8:     while code1_obj.run() or code2_obj.run() do
9:         Solve Code 1 system of equations
10:        Retrieve src_fieldC1 from Code 1
11:        if Code 1 == FEMuS then
12:            Compute P_0 src_fieldC1 from P_2 FEMuS field
13:        end if
14:        if moving_mesh then
15:            Update projection matrix P
16:        end if
17:        Compute trg_fieldC2 from interpolation over target mesh
18:        if Code 2 == FEMuS then
19:            Compute P_2 FEMuS field from P_0 trg_fieldC2
20:        end if
21:        Set trg_fieldC2 into Code 2
22:        Solve Code 2 system of equations
23:        Retrieve src_fieldC2 from Code 2
24:        if Code 2 == FEMuS then
25:            Compute P_0 src_fieldC2 from P_2 FEMuS field
26:        end if
27:        if moving_mesh then
28:            Update projection matrix P
29:        end if
30:        Compute trg_fieldC1 from interpolation over target mesh
31:        if Code 2 == FEMuS then
32:            Compute P_2 FEMuS field from P_0 trg_fieldC1
33:        end if
34:        Set trg_fieldC1 into Code 1
35:    end while
36: end procedure
```

At the supervisor level, the main function manages the instantiation and setup of both *Code 1* and *Code 2* classes (`Code1_obj` and `Code2_obj`), ensuring they are correctly configured and ready for interaction. The initialization process, described in Algorithm 2, includes the creation of interface objects for both codes. Thus, the `init_interface()` routine is called to

Algorithm 2 Interfaces initialization.

```
1: function INIT_INTERFACE(code1_obj, code2_obj)
2:     Set interface parameters, i.e. _mesh_name, _n_nodes
3:     Set connectivity of the interface _conn
4:     Set coordinates of the interface _coords
5:     if Code 1 == FEMuS or Code 2 == FEMuS then
6:         Convert to linear connectivity
7:     end if
8: end function
9: function CREATE_MESH(code1_obj.interface,code2_obj.interface)
10:    Set _conn, _coords and parameters into the MED mesh structure
11:    Create MED mesh copies from Code 1 and Code 2 meshes
12: end function
13: function INIT_MED_FIELD_ON_CELLS()
14:    Create MED field src_fieldC1 and trg_fieldC1 for Code 1
15:    Create MED field src_fieldC2 and trg_fieldC2 for Code 2
16: end function
```

initialize interfaces of the `femusMED` and `foamMED` classes using information on mesh connectivity and coordinate nodes. This information is retrieved with `set_mesh_connectivity()` and `set_mesh_coordinates()` routines. It is worth noting that the FEMuS code mesh is composed of biquadratic elements. However, the FEM interface to the MED coupling must use a linear mesh due to the compatibility requirements with OpenFOAM, which uses linear meshes, and the broader range of functionalities available in the MED library for handling linear meshes. Therefore, we extract data from the original FEMuS interface biquadratic elements and project them into linear elements using the `femusMED` routine `convert_to_linear_mesh()`.

The supervisor also manages the instantiation of the coupling object, which is responsible for numerically coupling the two codes. The `MED_obj` initialization involves creating structures to store mesh data from the two interfaces, setting up storage for the MED fields, and defining other parameters

that characterize the physical problem, such as problem dimension. As shown in Algorithm 2, in the initial steps of the coupling process, both *Code 1* and *Code 2* are required to generate a mesh copy in MED format corresponding to the computational domain (or a portion of it). The information retrieved to create the interfaces is passed to the MED framework to recreate the MED format mesh with `create_mesh()` method.

Following the creation of meshes, the MED field structures must be initialized within the MED framework. In strongly coupled problems, the input parameters of *Code 2* depend on the output of *Code 2* and vice versa. Configurations of this type require the creation of at least four fields: the output field of *Code 2* (`src_fieldC1`) that influences the input field of *Code 2* (`trg_fieldC2`), and the output field of *Code 2* (`src_fieldC2`) that influences the input field of *Code 2* (`trg_fieldC1`). For the FEMuS-OpenFOAM coupling, we choose a piecewise field representation. The FEMuS interface converts its native biquadratic fields into P_0 fields, aligning the representation of the data exchanged between the two codes. As a result, the data structures for both codes are created as P_0 fields using the `init_med_field_on_cells()` routine.

Once interface data are available, the projection operator $\mathbf{P}$ is computed for each of the interfaces involved in the simulation by the `get_remapper()` function. This procedure can be executed once before the time loop starts, provided the mesh does not change over time. However, the algorithm also allows the projection matrix to be recomputed at every time step if the simulation involves a moving mesh.

Both codes have completed their initialization at the supervisor level through their respective dedicated methods. The supervisor function can initiate the time loop for the coupled simulation since the data transfer interfaces have been fully configured, including their corresponding MED mesh copies and MED fields. The supervisor manages the time step synchronization between *Code 1* and *Code 2*. At each time step, the supervisor coordinates the field data exchange and monitors convergence criteria to ensure accuracy and stability. The time loop begins with the solver function run within *Code 1*, which is responsible for solving the system of governing equations of the specific physics being modeled. Once the *Code 1* has completed its computations and obtained a solution, the field named `src_fieldC1` is transferred to the corresponding MED fields. The transfer process involves a sequence of functions. Firstly, the interface class of *Code 1* uses the

method `get_field_from_Code1()` to retrieve the solution of the field from *Code 1*. Next, within the `MEDclass` class, the `set_field()` routine is set to the `src_fieldC1` into the corresponding MED field structure.

At this stage, the projection function `interpolate_field()` is called when a source field from *Code 1* needs to be interpolated from its source mesh onto a target grid compatible with *Code 2*. The field interpolation uses the projection matrix calculated in the previous step only if the mesh is not moving. Otherwise, the **P** matrix must be computed at each time step using `get_remapper()` routine before calling the `interpolate_field()` method. In the FEMuS-OpenFOAM coupling application, a P_0 to P_0 interpolation scheme requires that both fields of FEMuS and OpenFOAM codes are represented as cell-wise fields. When *Code 2* is FEMuS, it becomes necessary to convert its solution into a cell-wise field. After the FEMuS solver is executed, the algorithm interpolates P_2 nodal solutions into P_0 cell-wise solutions. The resulting `trg_fieldC2`, obtained from the interpolation process, is now available in memory as a MED object. Through an inverse process, this field can be stored as the solution for *Code 2* using the interface function `set_field_to_Code2()`. In cases where *Code 2* is FEMuS, an additional interpolation algorithm from P_0 to P_2 is applied before invoking the `set_field_to_Code2()` function.

With the solution provided by *Code 1*, *Code 2* can proceed to solve its specific physics. Once the solution of the system of equations in *Code 2* is obtained, it provides the field to be exchanged back to *Code 1* using a mechanism analogous to the previous one. This procedure is necessary for the Block Gauss-Seidel algorithm, which requires that the most recent solution be written into *Code 2* as soon as it becomes available. The interface method `get_field_from_Code2()` is called to extract solutions from *Code 2*. Then, `set_field()` is employed to set the solution of the *Code 2* into the corresponding MED field. The interpolation function is used to interpolate the MED field from *Code 2*, `src_fieldC2`, to the target MED field associated with *Code 1*, `trg_fieldC1`. Finally, this interpolated field is written into *Code 1* using the `set_field_to_Code1()` routine. Once *Code 1* receives the solution from *Code 2*, the data exchange between the two codes is completed. With both codes now equipped with the necessary fields, the time loop can proceed to the next time iteration at the supervisor level.

The implemented algorithm includes the option to apply a sub-iteration process to enhance the strength of the coupling. This possibility allows for the

specification of a maximum number of sub-iterations along with the definition of suitable convergence criteria before proceeding to the computation of the next time step. This iterative process is repeated at each time step until the end of the simulation.

2.4 Validation

This section presents two numerical examples to validate the implemented algorithm, focusing on volume and boundary field transfer. The first example involves a buoyancy-driven cavity and is used to test the volume data transfer algorithm. The two subsystems, FEMuS and OpenFOAM, solve the velocity and temperature fields independently, while the coupling application ensures the exchange of these fields across the entire domain. The boundary coupling is evaluated by solving a conjugate heat transfer problem, where temperature and heat flux are exchanged at the interface of solid and fluid domains.

This section describes the two problems, defining the governing equations, the boundary conditions, and the solution process of the coupling algorithm. The numerical results are compared to reference data for the same application setting.

2.4.1 Differentially Heated Cavity (DHC)

Natural convection, also referred to as buoyancy-driven convection, is a common phenomenon driven by temperature differences with significant practical engineering applications. A key area of study is the natural convection phenomenon occurring within the Differentially Heated Cavities (DHC) configuration. In DHC, fluid is heated along one boundary and cooled along another, creating a temperature difference between vertical walls that drives flow through buoyancy forces.

This phenomenon plays an important role in several industrial processes. For instance, it occurs in double-glazed windows, where two or more glass window panes are separated by a space filled with inert gas or air. This system is designed to enhance energy efficiency by minimizing heat transfer across windows [92, 93]. Another application of natural convection in enclosures can be found in solar collectors [94, 95, 96]. In these systems, uneven heating of a fluid occurs as the absorber plate is warmed by solar radiation, creating a temperature gradient between the absorber and the surrounding air or

glass cover. Studying the physical phenomenon of the air gap between the absorber plate and the glass cover allows us to minimize heat losses and enhance energy absorption. A similar application is found in gas-filled cavities where buoyancy-driven convection is used for cooling nuclear reactor cores [97]. These cavities can be used as cooling mechanisms in advanced reactor designs, including gas-cooled fast reactors (GFRs) and high-temperature gas-cooled reactors (HTGRs).

In addition to its numerous applications in engineering processes, the Differential Heated Cavity configurations are benchmark cases for developing and validating numerical algorithms and computational codes for thermal problems. The thermal cavity problem has been widely studied, particularly for a Prandtl number of $Pr = 0.71$, and several reference works are available in the literature. De Vahl Davis et al. [98], in the first years of the 1980s, solved this problem to address the stream-function-vorticity form of the governing equations. Their work provides a comprehensive set of flow data results for several DHC configurations. Massarotti et al. [99] addressed the same problem using a semi-implicit version of the characteristic-based split scheme (CBS) with equal-order interpolation functions for all variables. In [100], Manzari's research group analyzed the air-filled cavity employing an artificial compressibility (AC) method to couple the pressure and velocity fields. Additionally, they used an artificial dissipation (AD) method to enhance the stability of the numerical solution. More recently, in [101], Manzari et al. extended the scheme used in their earlier work to address incompressible flow problems with heat transfer, using the DHC problem as a validation benchmark. Mayne et al. [102] employed an h-adaptive finite-element method to ensure a very accurate solution for thermal cavity problems. In [103], Wan et al. used two methods to solve the Navier-Stokes and energy equations in the DHC configuration. The first is a highly accurate quasi-wavelength-based discrete singular convolution (DSC) method. The second one is a standard Galerkin finite-element method. In their work, they conducted a detailed analysis of the numerical simulation, providing extensive benchmark data for laminar natural convection problems.

A Differentially Heated Cavity typically consists of a rectangular or square enclosure where opposite vertical walls are maintained at different temperatures, while the horizontal ones are adiabatic. Inside this enclosure, fluid flow arises along the hot wall and descends along the cold wall. The temperature difference between the two vertical walls generates buoyancy forces within

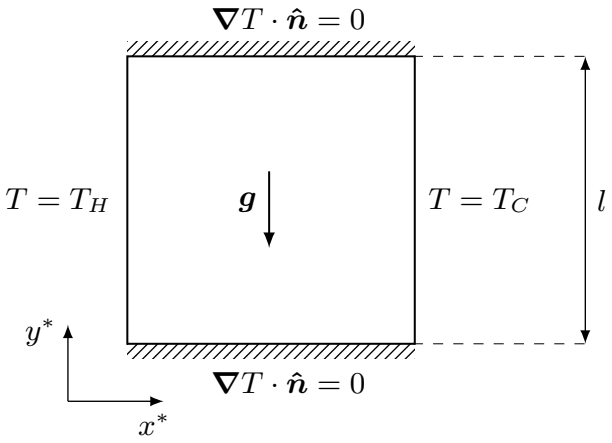

Figure 2.10: Geometry of the buoyant cavity problem with boundary conditions for the temperature field.

the fluid, driving its motion, which would otherwise remain steady.

Figure 2.10 represents the bidimensional square cavity used as the numerical domain for validating the present coupling application. As seen in Figure 2.10, the aspect ratio of the enclosure, defined by the height-to-width ratio of the cavity, is unitary. Variations in aspect ratio can result in different levels of complexity in flow structures and thermal performance. In this study, we consider an incompressible Newtonian fluid. In particular, named the velocity $\boldsymbol{u}$, the pressure p, and the temperature T, the governing equations for the natural laminar convection are as follows

$$\begin{aligned} &\frac{\partial u_i}{\partial x_i} = 0, \\ &\frac{Du_i}{Dt} = -\frac{1}{\rho}\frac{\partial p}{\partial x_i} + \frac{\partial}{\partial x_j}(\nu S_{ij}) + g_i\beta(T - T_0), \\ &\frac{DT}{Dt} = \frac{\partial}{\partial x_i}\left(\alpha\frac{\partial T}{\partial x_i}\right), \end{aligned} \tag{2.10}$$

where ν is the kinematic viscosity, ρ the density, β the coefficient of thermal expansion, α the thermal diffusivity, T_0 the reference temperature, set to the mean value between T_H and T_C. Since the coupling term is explicitly represented by the buoyancy force, natural convection phenomena are a well-suited problem for validating the coupling algorithm. Regarding the boundary condition, no-slip boundary conditions are imposed for the velocity field at each boundary edge. For the energy equation, Dirichlet and Neumann boundary

conditions are used. In particular, uniform and homogeneous temperatures are applied to the two opposite vertical edges, with the two Dirichlet boundary conditions creating a hot and cold wall. An insulation condition has been imposed on the remaining edges, according to Figure 2.10. Furthermore, the volumetric thermal source Q is set to zero for every numerical simulation.

This configuration provides a framework for analyzing the effects of different parameters on fluid flow and heat transfer efficiency. In particular, three main parameters govern the behavior of the solution in this setup. The Rayleigh number (Ra) is a dimensionless parameter that quantifies the strength of the thermal buoyancy against the viscous and thermal diffusion in a fluid. Higher Ra values typically lead to more intense convection within the system. Alongside the Rayleigh number, the Prandtl number represents the ratio of momentum diffusivity (viscosity) to thermal diffusivity. The variations in Pr can significantly influence flow stability and heat transfer behavior. For example, as Pr increases, flow stability tends to increase, while the heat transfer efficiency generally decreases at a constant Ra. The third parameter is the Grashof number (Gr), defined as a combination of the other two parameters. This dimensionless quantity represents the balance between buoyancy and viscous forces and controls natural convection within the system. The definitions of the aforementioned parameters are as follows

$$Ra = \frac{g\rho\beta L^3(T_H - T_C)}{\nu\alpha}, \tag{2.11}$$

$$Pr = \frac{\nu}{\alpha}, \tag{2.12}$$

$$Gr = \frac{g\rho\beta L^3(T_H - T_C)}{\nu^2} = \frac{Ra}{Pr}, \tag{2.13}$$

where L is the reference length of the domain.

Volume data transfer algorithm

This section details the solution method for the specific case of a volume data transfer algorithm in the coupling application. Two different approaches are employed to solve the Differentially Heated Cavity problem described in 2.4.1. The first coupling case, referred to as c_1, includes that the FEM code, FEMuS, solves for the temperature field, and the FVM code, OpenFOAM, solves for the velocity field. In the second coupling case, referred to as c_2, the reverse coupling is applied: FEMuS solves the Navier-Stokes equations, while OpenFOAM handles the energy equation. In both procedures, the coupling

between the codes occurs through two equations: the buoyancy term, which requires the temperature field in the momentum equation, and the advection term in the energy equation, which is computed via the velocity field coming from the momentum equation. This configuration is a necessary condition for the cases c_1 and c_2 to satisfy the problem described in (2.10). The field transfer between the codes is performed considering the volumetric value of the specific field. For each cell of the target mesh, the field is interpolated from the source mesh by using the MED structures described in the previous sections. The entire volumetric field is transferred between the two codes, adopting the same discretization for the domain, even if the FEM codes consider biquadratic quadrilateral elements, as the FVM code uses linear quadrilateral elements.

Algorithm 3 Volume data transfer algorithm.

procedure SUPERVISOR CONTROL
    Instantiate class objects
    Initialize volume interfaces
    Create MED meshes and fields
    Create projection matrix `_remapper`

Time loop

    while `OpenFOAM_obj.run()` **do**
        Solve T with FEMuS solver
        Retrieve `source_P2_T` from FEMuS
        Compute `source_P0_T` from `source_P2_T`
        Compute `target_P0_T` from interpolation over OpenFOAM mesh
        Set `target_P0_T` into OpenFOAM
        Solve **u** with OpenFOAM solver
        Retrieve `source_P0_v` from OpenFOAM
        Compute `target_P0_v` from interpolation over FEMuS mesh
        Compute `target_P2_v` from `target_P0_v`
        Set `target_P2_v` into FEMuS
    end while
end procedure

Following Algorithm 3, the procedure employed for the coupling application involving volume data transfer is described. Firstly, both coupling cases c_1 and c_2 use `init_interface()` and `create_mesh()` routines to create

a MED mesh object representing the entire domain. Through the FEMuS interface class, the quadratic mesh is converted into MED format, and the structure of the interface domain is created. In this specific application, the FEMuS code employs a 2D computational grid with QUAD 9 Lagrangian elements. This mesh is then translated into a MED-format mesh object, and a corresponding linear mesh is generated to facilitate the field interpolation process. Regarding the OpenFOAM framework, a 3D domain is used to represent the two-dimensional cavity, as OpenFOAM can only handle 3D meshes, even for 2D problems. The OpenFOAM class is used to clone the OpenFOAM mesh and the resulting MED mesh consists of hexahedral linear elements (HEX 8).

The initialization of MED fields over the MED meshes involves calling the routine `init_med_field_on_cells()`. Both c_1 and c_2 cases create a cell-wise temperature field and a cell-wise velocity field over the whole domain for FEMuS and OpenFOAM MED interfaces. We can refer to these MED fields as `source_P0_T` and `target_P0_T` for the temperature field, and `source_P0_v` and `target_P0_v` for the velocity field.

This application does not involve mesh movement, so the projection matrix `_remapper` remains constant over time and can be computed before the time loop starts.

The employed coupling method c_1 is described in the following, while the c_2 case can be extrapolated since it is entirely similar to the first one. As the time loop starts, FEMuS solves the temperature equation described in (2.10). The temperature solution over the entire domain is extracted from FEMuS code using the function `get_field_from_femus()`. It is worth noting that the FEMuS code solves for a biquadratic (P_2) temperature field, requiring an interpolation from P_2 to P_0 to enable the $P_0 - P_0$ interpolation managed by the MED routines. This P_2 to P_0 interpolation is handled internally by the FEMuS interface class. The resulting P_0 temperature field is assigned to the MED field named `source_P0_T` through the routine `set_field()`. For the case c_1, this source field is defined over the cloned MED mesh of FEMuS representing the internal volume of the computational domain.

At this point, the `interpolate_field()` routine performs the $P_0 - P_0$ interpolation to transfer the `source_P0_T` to the OpenFOAM interface. This operation computes the target MED field, `target_P0_T`, over the MED target mesh using the interpolation matrix calculated by the MED routines. The function `set_field_to_OpenFOAM()` set the interpolated field into the

OpenFOAM temperature field, used to compute the buoyancy term as in (2.10). Once OpenFOAM has solved the Navier-Stokes equation, the velocity field is extracted using `get_field_from_OpenFOAM()` routine and set to the `source_P0_v` field. The interpolation function computes the `target_P0_v` to be transferred to the FEMuS velocity field with `set_field_to_femus()` routine. To compute the advection contribution in the FEMuS energy equation, this volumetric field must be first interpolated using the P_0 to P_2 interpolation scheme. The $P_0 - P_2$ interpolation is performed by computing a weighted distribution over the quadratic nodes of the piece-wise field. FEMuS use this updated velocity field to solve the temperature equation in the following time iteration.

Simulations Results

In this section, the coupling application results obtained for the DHC problem are presented and compared with the literature data referenced in Section 2.4.1. Additionally, two simulations have been performed by solving the system of equations (2.10) considering a monolithic solution with the FEMuS code and OpenFOAM. These two solutions are labeled by F and OF, respectively. The results from the monolithic approaches are compared to the coupling application results and used as references as the literature benchmark.

The problem is described in the previous sections, referring to Figure 2.10 and additional information about OpenFOAM configuration parameters is provided in Appendix B. The simulations performed are obtained by varying the governing parameters of the problem, the Prandtl number and the Rayleigh number. In particular, two different Prandtl numbers are considered, corresponding to an air-filled cavity ($Pr = 0.71$) and a water-filled cavity ($Pr = 7$). Regarding the Rayleigh number, four cases are simulated, ranging from 10^3 to 10^6. The numerical fields resulting from the computation have been non-dimensionalized with the following expressions

$$\boldsymbol{x}^* = \frac{\boldsymbol{x}}{L}, \quad \boldsymbol{u}^* = \frac{\boldsymbol{u}\, L}{\alpha}, \quad \Theta = \frac{T - T_C}{\Delta T}, \quad \Psi = \frac{\psi}{\alpha} \tag{2.14}$$

where T_C represents the Dirichlet boundary condition for the temperature on the cold wall, and Ψ represents the non-dimensional stream function. Naturally, by considering variables such as the non-dimensional temperature Θ, the non-homogeneous Dirichlet boundary conditions change their specific

values: on the cold wall, we now have $\Theta = 0$, while on the hot one, we have $\Theta = 1$.

Air filled cavity - $Pr = 0.71$. In this paragraph, natural convection for an air-filled square cavity is studied. The numerical tests for the validation of the algorithm have been performed for different Ra numbers, ranging from 10^3 up to 10^6 and compared with reference data [98, 99, 101, 102, 103]. In Figure 2.11, the grid convergence analysis is reported for the maximum

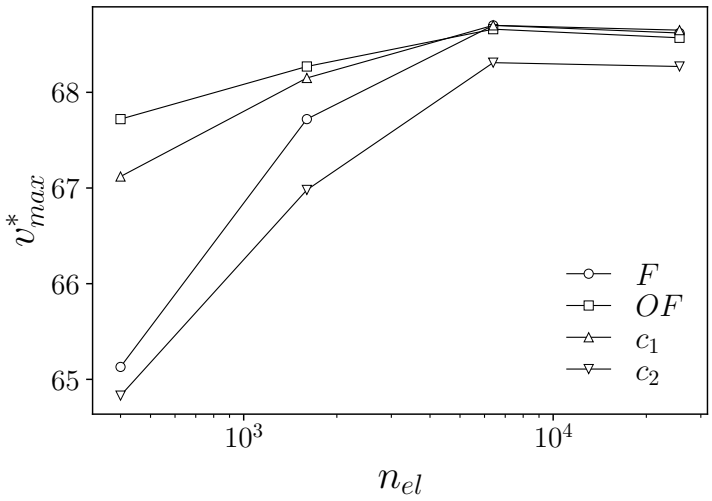

Figure 2.11: Grid convergence of the v^*_{max} value at $y^* = 0.5$ for the case with $Ra = 10^5$, for both the monolithic solutions and the coupling applications.

value of the v^* component evaluated at $y^* = 0.5$, for the case of $Ra = 10^5$. In particular, four types of grid size have been investigated, corresponding respectively to 400, 1600, 6400, and 25600 elements (n_{el}) for each of the four simulation setups. The four simulations show the same convergence behavior, with the optimal refinement determined to be an 80×80 grid.

Figures 2.12-2.15 analyzes the natural convection patterns of the square-cavity problem varying with the Ra number. The enclosure with differentially heated vertical walls is characterized by two different flow patterns: the variation in the thickness of the boundary layer along the wall and the recirculating motion in the core region. The boundary layer growth is predominant in simulations with higher Rayleigh numbers. In contrast, the recirculating motion in the core region has more strength in simulations with lower Rayleigh numbers. The variation of the flow patterns is depicted in Figure 2.12, where

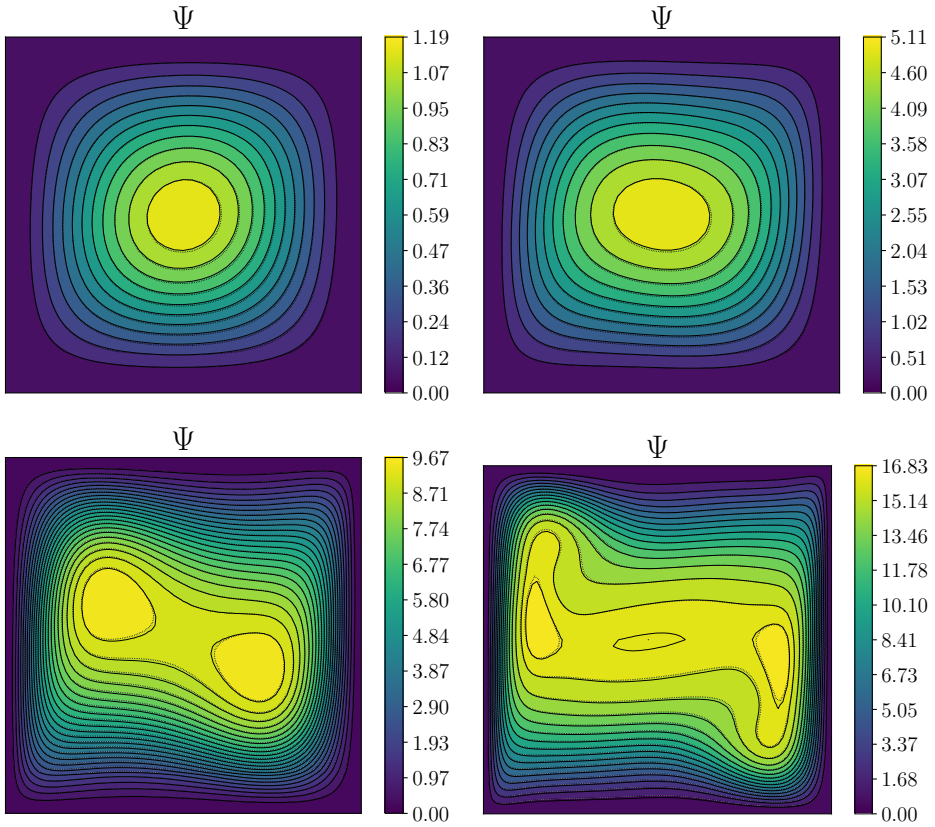

Figure 2.12: Non-dimensional stream function contour, Ψ, for low Rayleigh number on the top, $Ra = 10^3$ (left) and $Ra = 10^4$ (right), and for high Rayleigh number on the bottom, $Ra = 10^5$ (left) and $Ra = 10^6$ (right). Coupling algorithm c_1 (solid line) and c_2 (dotted line).

the contour maps of the stream function Ψ are reported. This Figure shows that the simulations with $Ra = 10^3$ and 10^4 generate only a single circulating eddy in the core region. In contrast, a higher Rayleigh number creates a more complex flow pattern. In particular, for $Ra = 10^5$, the main eddy splits into two smaller counter-rotating eddies, which are stretched toward the top left and bottom right corners. As the Rayleigh number increases, the fluid flow patterns undergo additional transformation, with the inner secondary eddies moving closer to the hot and cold walls.

The streamlines are reported for both the coupling simulations (c_1 and c_2), showing a very similar behavior in all four simulations. The streamlines for c_1 are indicated using the solid lines, while the c_2 results are depicted using dotted lines. The results reported in Figure 2.12 display a good agreement

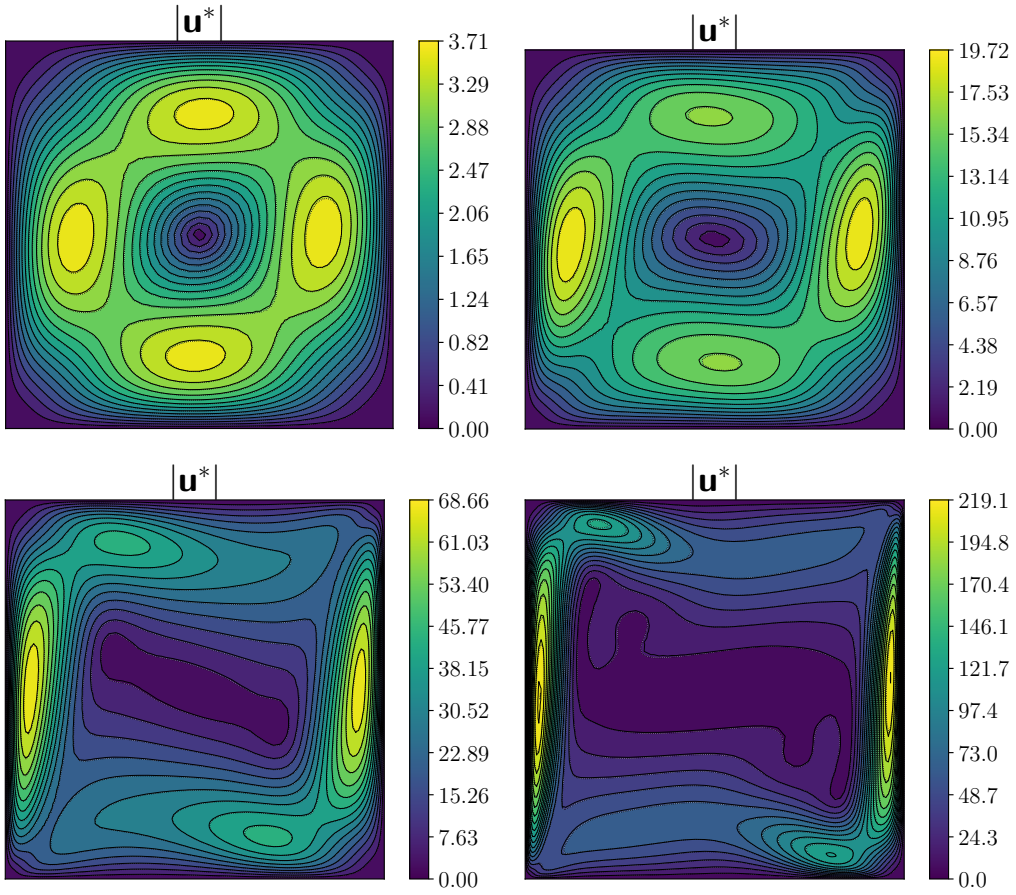

Figure 2.13: Non-dimensional velocity magnitude contour, $|\mathbf{u}^*|$, for low Rayleigh number on the top, $Ra = 10^3$ (left) and $Ra = 10^4$ (right), and for high Rayleigh number on the bottom, $Ra = 10^5$ (left) and $Ra = 10^6$ (right). Coupling algorithm c_1 (solid line) and c_2 (dotted line).

with the streamlines computed in [98] and [103]. Table 2.1 compares the stream function field with the corresponding values reported in [98] and [103]. Here, Ψ_{max} represents the maximum value of the non-dimensional stream function, while Ψ_{mid} refers to the value of the stream function at the midpoint of the cavity. As observed, simulations with lower Rayleigh numbers yield identical values for both Ψ_{max} and Ψ_{mid}. This equivalence occurs because, at lower Rayleigh numbers, the inner eddy reaches its maximum intensity at the midpoint of the cavity. For the other two simulations, Ψ_{max} values are not located at the center of the cavity but are instead shifted toward the upper left side of the cavity. In particular, for the case with $Ra = 10^5$ the maximum

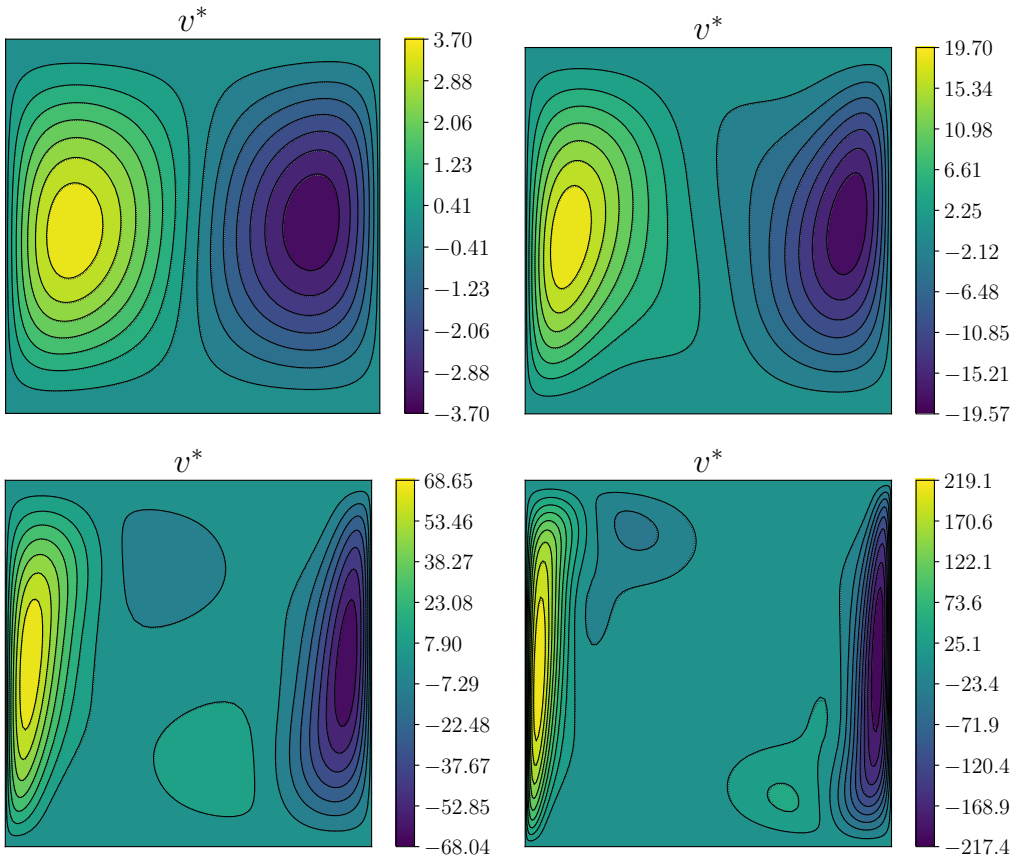

Figure 2.14: Non-dimensional vertical velocity contour, v^*, for low Rayleigh number on the top, $Ra = 10^3$ (left) and $Ra = 10^4$ (right), and for high Rayleigh number on the bottom, $Ra = 10^5$ (left) and $Ra = 10^6$ (right). Coupling algorithm c_1 (solid line) and c_2 (dotted line).

value of Ψ occurs at $x^* = 0.291$ and $y^* = 0.601$, for both c_1 and c_2 simulations. The literature results reported in [98] locates the maximum at $x^* = 0.285$ and $y^* = 0.601$. The values of Ψ_{max} computed for the simulations with $Ra = 10^6$ are located at $(x^*, y^*) = (0.146, 0.551)$ in c_1 and $(x^*, y^*) = (0.153, 0.537)$ in c_2, while the corresponding benchmark solution is located at $x^* = 0.151$ and $y^* = 0.547$. Consequently, both simulations can predict the maximum values of Ψ and their locations with a good agreement comparing them with the one calculated in [98].

The velocity isocontours are reported in Figures 2.13-2.15 for all the simulations performed. Here, the isolines of the velocity magnitude ($|\boldsymbol{u}^*|$) and the non-dimensional velocity components (u^*, v^*) are reported for four Ra

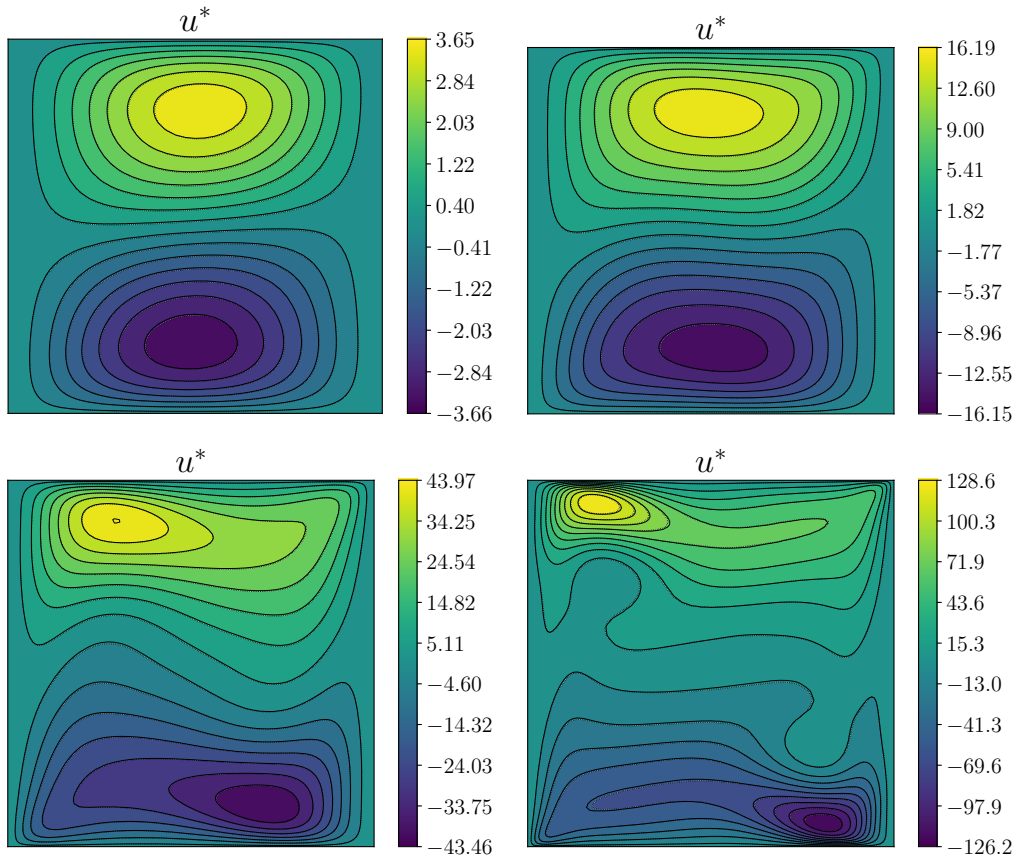

Figure 2.15: Non-dimensional horizontal velocity contour, u^*, for low Rayleigh number on the top, $Ra = 10^3$ (left) and $Ra = 10^4$ (right), and for high Rayleigh number on the bottom, $Ra = 10^5$ (left) and $Ra = 10^6$ (right). Coupling algorithm c_1 (solid line) and c_2 (dotted line).

numbers (from 10^3 up to 10^6) for the two coupling algorithms (c_1 and c_2). The solid line tracks the isolines of the case c_1 and the dotted one is the simulation of the case c_2. Figure 2.15 shows the evolution of the u^* contours as the Rayleigh number increases. The simulations performed with $Ra = 10^3$ and $Ra = 10^4$ generate two horizontal eddies positioned one below the other. As the Rayleigh number increases, these two eddies shift closer to the adiabatic walls. In a similar way, in Figure 2.14, the non-dimensional v^*-component exhibits two dominant circulations near the left and right walls that move closer to the hot wall and the cold wall as the Ra number increases. This behavior produces a visible reduction of the boundary layer. Reference results of the physical field contours can be widely found in the literature for

Ra	Ψ_{max}			Ψ_{mid}			
	c_1	c_2	[98]	c_1	c_2	[98]	[99]
10^3	1.181	1.186	1.174	1.181	1.186	1.174	1.167
10^4	5.088	5.109	5.071	5.088	5.109	5.071	5.075
10^5	9.630	9.643	9.612	9.106	9.173	9.111	9.153
10^6	16.83	16.82	16.75	16.39	16.42	16.32	16.49

Table 2.1: Non-dimensional maximum values of the stream function, Ψ_{max}, and non-dimensional values of the stream function at the midpoint of the cavity, Ψ_{mid}. Simulation results (c_1 and c_2) compared to literature data ([98] and [99]).

this problem. For this reason, the interested reader can refer to [103] and references therein. For every case of Ra number, the contour isolines agree with the data found in the literature. In Tables 2.2 and 2.3, the maximum value of the non-dimensional velocity components, u^* and v^* evaluated are reported respectively at the planes $x^* = 0.5$ and $v^* = 0.5$ with different Ra numbers and compare them with the same data taken from the literature. A

Ra	F	OF	c_1	c_2	[98]	[101]	[102]	[103]
10^3	3.658	3.646	3.660	3.662	3.63	3.68	3.65	3.49
10^4	16.178	16.224	16.218	16.208	16.18	16.10	16.18	16.12
10^5	34.791	34.961	34.670	34.497	34.81	34.00	34.77	33.39
10^6	64.808	65.180	64.625	64.450	65.33	65.40	64.69	65.40

Table 2.2: Maximum values of u^*-component at $x^* = 0.5$, for different Ra numbers and comparison with literature data.

good agreement can be seen for the maximum value of the non-dimensional velocity components.

The evolution of the pattern in the non-dimensional temperature field (Θ) is shown in Figure 2.16. Here, the isotherms of the non-dimensional temperature are presented for both c_1 (solid line) and c_2 (dotted line). The vertical temperature distribution at lower Ra evolves into a horizontal temperature distribution in the core of the cavity at higher Ra. In the immediate neighborhood of the hot and cold walls, the contours remain parallel to them.

Ra	F	OF	c_1	c_2	[98]	[101]	[99]	[102]	[103]
10^3	3.71	3.70	3.70	3.71	3.68	3.73	3.69	3.70	3.69
10^4	19.63	19.64	19.69	19.71	19.51	19.90	19.63	19.62	19.76
10^5	68.66	68.70	68.79	68.31	68.22	70.00	68.85	68.69	70.63
10^6	220.4	219.9	221.4	221.0	216.8	228.0	221.6	220.8	227.1

Table 2.3: Maximum value of v^*-component at $y^* = 0.5$, for different Ra numbers and comparison with literature data.

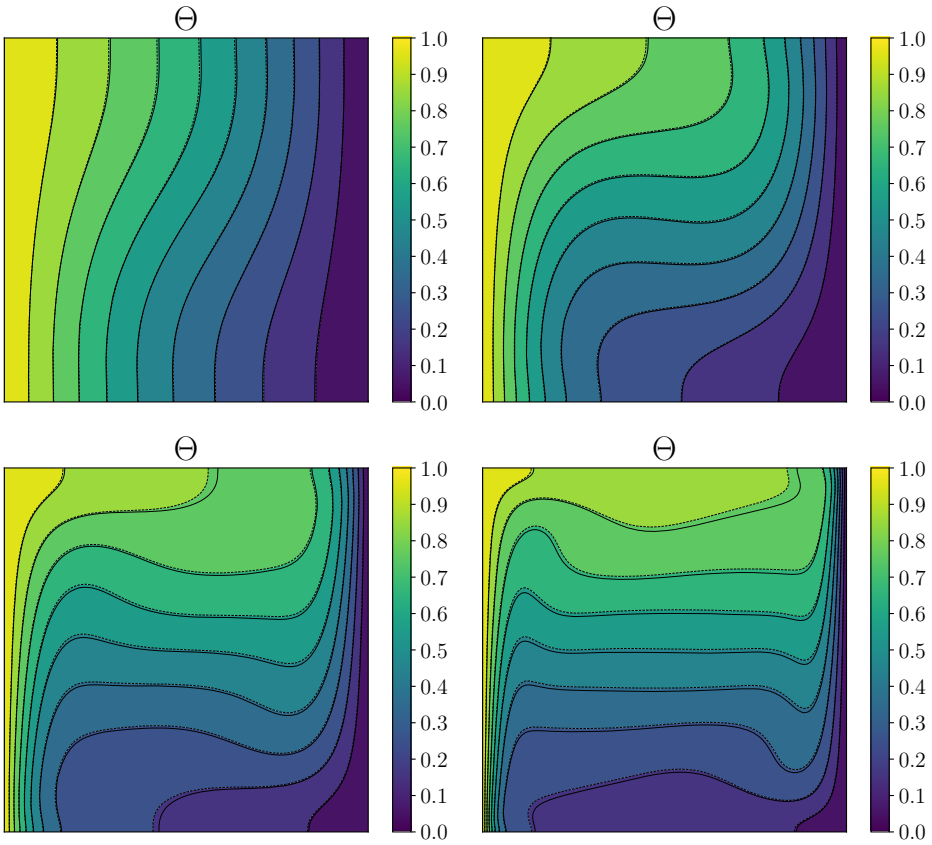

Figure 2.16: Non-dimensional temperature contour, Θ, for low Rayleigh number on the top, $Ra = 10^3$ (left) and $Ra = 10^4$ (right), and for high Rayleigh number on the bottom, $Ra = 10^5$ (left) and $Ra = 10^6$ (right). Coupling algorithm c_1 (solid line) and c_2 (dotted line).

In Figures 2.17-2.18, the non-dimensional velocity components and the non-dimensional temperature plots are reported for every type of the four

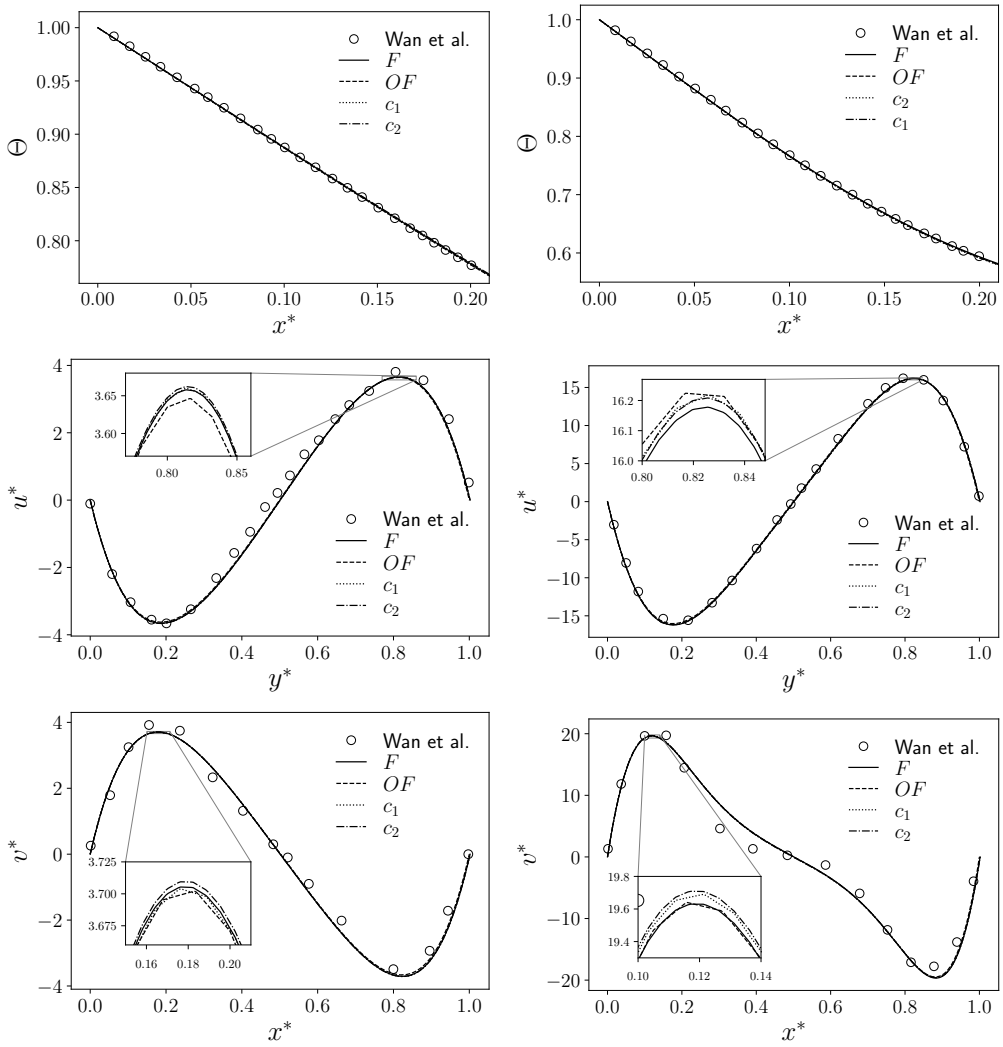

Figure 2.17: Non-dimensional temperature Θ (at $y^* = 0.5$, top) and non-dimensional components u^* (at $x^* = 0.5$, middle) and v^* (at $y^* = 0.5$, bottom) for the four types of simulations (F, OF, c_1 and c_2) with a comparison with literature data from [103] (circular markers). Case with $Ra = 10^3$ on the left and $Ra = 10^4$ on the right.

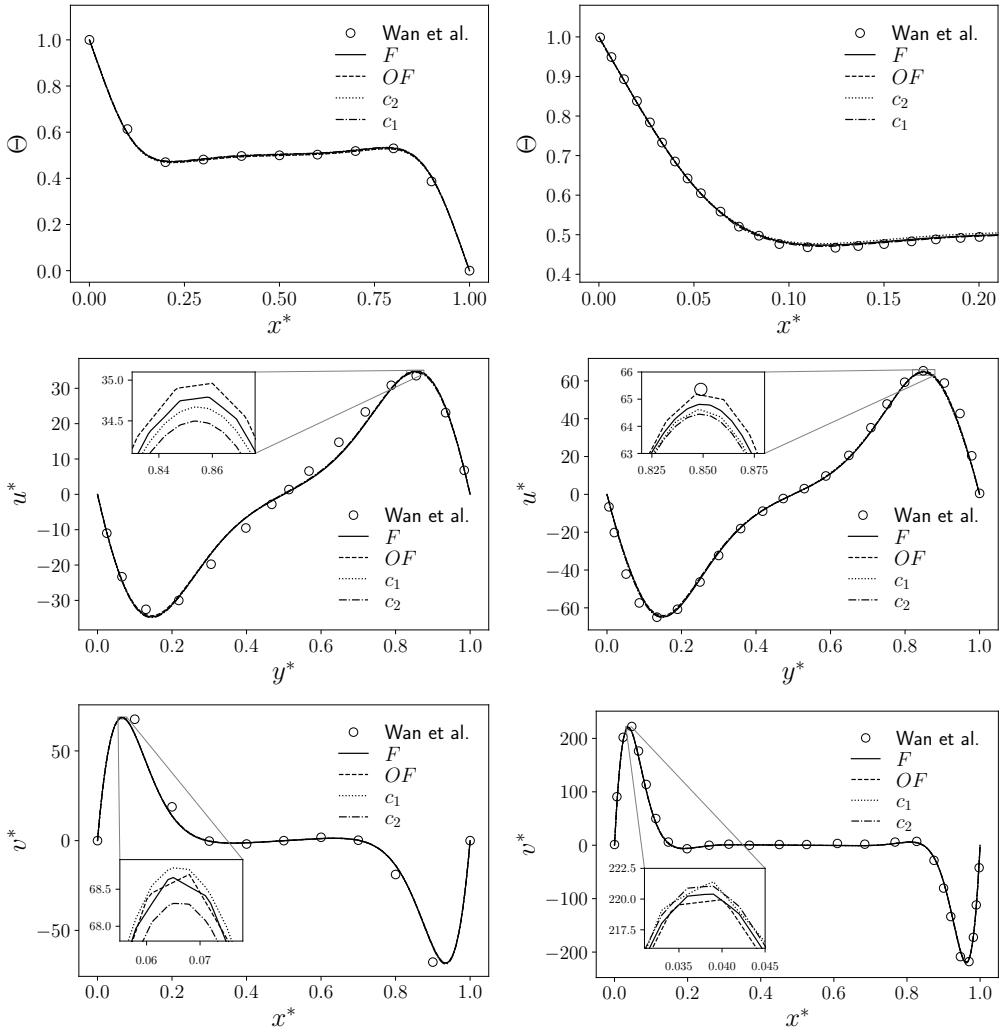

Figure 2.18: Non-dimensional temperature Θ (at $y^* = 0.5$, top) and non-dimensional components u^* (at $x^* = 0.5$, middle) and v^* (at $y^* = 0.5$, bottom) for the four types of simulations (F, OF, c_1 and c_2) with a comparison with literature data from [103] (circular markers). Case with $Ra = 10^5$ on the left and $Ra = 10^6$ on the right.

simulations (F, OF, c_1, c_2) and for every Ra number. A comparison with literature data from [103], symbolized with circular markers, is also highlighted. Specifically, these plots refer to the variables' behavior at specific points in the domain: the line $x^* = 0.5$ for the u^* component and the line $y^* = 0.5$ for the v^* component and the temperature Θ.

Regarding the latter variable, the plotted domain is restricted to $x^* \in [0, 0.2]$ (apart from $Ra = 10^5$) since the literature data can be found only in this interval. Moreover, for the same Θ, a good agreement with reference data published in [103] is present for every case and every type of simulation, including both coupled algorithms. For this reason, a zoom on specific regions of the non-dimensional temperature plot is not provided since the lines of the four simulations are almost overlapping. The same trend can also be seen for the v^* component and for the u^* component. Each of the simulations seems to produce the same numerical solution, confirming the goodness of the simulations and coupling procedure. A zoom of the plot is provided in the region close to the maximum/minimum of the velocity components to highlight better the slight differences between the four simulations and the literature results.

One of the key quantities for designers and engineers is the quantification of heat transfer, and in this context, the heat transfer coefficient along the hot and cold walls plays an important role. The Nusselt number, which expresses the ratio of convective heat transfer to conductive heat transfer at the wall, is commonly used to quantify the heat transfer along the walls. The local Nusselt number is defined as

$$Nu_{loc} = \pm \frac{\partial \Theta}{\partial x}|_{wall}, \tag{2.15}$$

where the negative sign refers to heat transfer from the wall to the fluid (hot wall), while the positive sign indicates heat transfer from the fluid to the wall (cold wall). In Figure 2.19, the local Nusselt number computed along the hot wall is presented for the coupling simulations c_1 and c_2. These plots are compared to the reference data from [99], [101], and [103]. It can be observed that for low Rayleigh numbers, both coupling simulations show good agreement with all the reference data. For $Ra = 10^5$, the results align with the references in [101] and [103], while the solution from [99] shows slight differences when compared to the others. These differences become more emphasized for $Ra = 10^6$, where all three reference datasets produce distinct behaviors. Both c_1 and c_2 simulations appear to closely match the

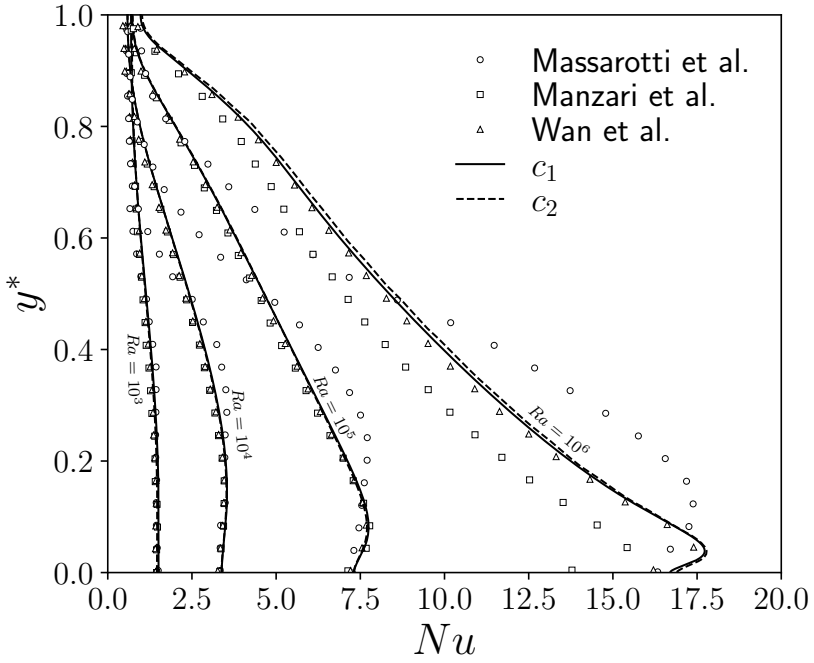

Figure 2.19: Comparison between reference data [99, 101, 103] and coupling simulation results of the Nusselt number profile along the hot wall for the four Rayleigh numbers.

solution provided by [103].

In Table 2.4, the average Nusselt values are reported for all the cases simulated and compared to the literature references. The average Nusselt number, which is a design parameter of interest, is obtained from the following expression

$$Nu = \int_0^1 Nu_{loc} dy. \tag{2.16}$$

Table 2.4 shows that all the simulations are in good agreeement with the references for all the four Raileigh number.

Water filled cavity - $Pr = 7$ Many studies examine the relationship between Nu, Re, and Ra. However, a significant gap remains in the literature concerning the effects of Pr on flow and heat transfer in the DHC configuration. Most investigations have used air ($Pr = 0.71$) as the working fluid in the cavity to construct benchmarks. Furthermore, research on the relationship between Reynolds and Prandtl numbers has predominantly focused on Rayleigh-Bénard convection (RBC) [104]. Only a few studies have addressed

Ra	F	OF	c_1	c_2	[98]	[99]	[101]	[102]	[103]
10^3	1.117	1.142	1.121	1.113	1.118	1.074	1.117	1.115	1.117
10^4	2.241	2.300	2.255	2.245	2.243	2.084	2.243	2.259	2.254
10^5	4.480	4.639	4.521	4.514	4.519	4.30	4.521	4.483	4.598
10^6	8.824	9.066	8.842	9.008	8.800	8.743	8.806	8.881	8.632

Table 2.4: Average values of Nusselt number on the hot wall for different Ra numbers and comparison with literature data.

whether solutions validated for $Pr = 0.71$ apply to significantly higher or lower Prandtl numbers. Thus, the influence of Pr on the behavior of the DHC configuration remains mostly an open problem. This analysis is particularly important because higher Pr values are relevant for many thermal engineering applications. For example, water has a Prandtl number equal to 7, and oils have values reaching up to 10^5.

One of the few studies examining the impact of the Prandtl number is by Kennelly et al. [105], which analyzed its influence on temperature and velocity fields. They found that the Prandtl number significantly affects the flow and heat transfer characteristics within the DHC configuration. For a constant Ra, substantial differences were observed in the velocity and temperature fields between low and high Pr fluids, particularly near the cavity corners, where inertial effects are significant. In particular, at constant Ra, increasing Pr reduces the number of eddies within the cavity since the fluid flow becomes more stable. Moreover, as the Pr number increases, the heat transfer performances increase, resulting in higher Nusselt numbers.

In Figure 2.20, the u^*- and v^*-components of the velocity pattern are shown for the $Pr = 7$ case. These results were obtained using the c_1 and c_2 simulations for the four values of Rayleigh number cases, from 10^3 to 10^6. Solid lines represent the c_1 simulation results, while the c_2 simulation results are depicted using dotted lines. As shown in Figure 2.20, the primary flow characteristics are similar to those of an air-filled cavity. In particular, as the Rayleigh number increases, the eddies move closer to the cavity walls: the horizontal eddies shift, and the vertical eddies migrate closer to the vertical walls, causing a sensible reduction of the boundary layer. However, the results differ from those observed for $Pr = 0.71$, and these differences become more pronounced with increasing Rayleigh number.

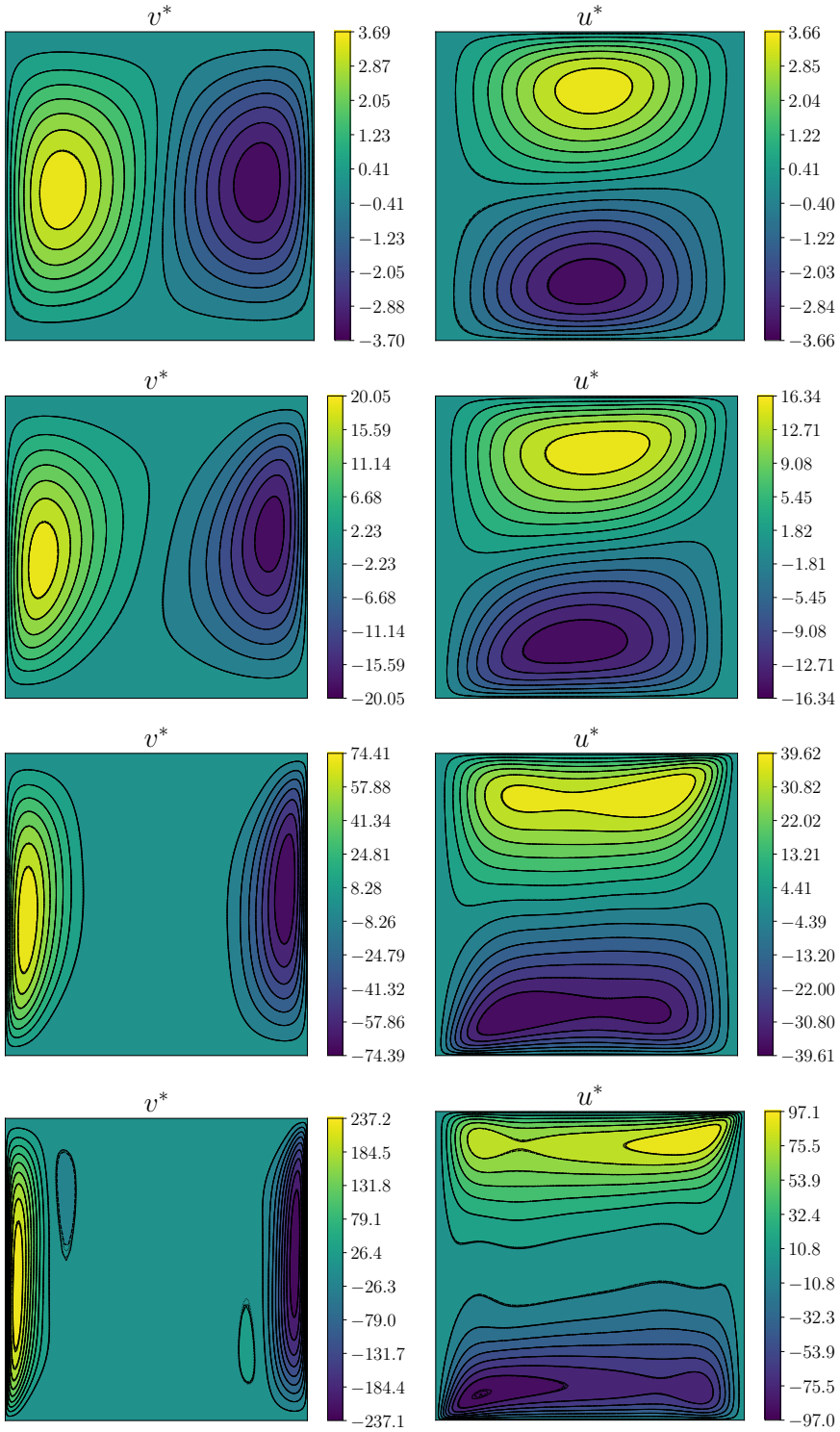

Figure 2.20: Contour lines of non-dimensional v^*-component on the left and non-dimensional u^*-component on the right for the four cases of Rayleigh numbers, from $Ra = 10^3$ (top) to $Ra = 10^6$ (bottom). Coupling algorithm c_1 (solid) and c_2 (dotted).

Ra	v^*				u^*			
	F	OF	c_1	c_2	F	OF	c_1	c_2
10^3	3.70	3.69	3.69	3.71	3.67	3.66	3.66	3.67
10^4	19.84	19.82	19.86	19.93	16.28	16.25	16.27	16.32
10^5	73.82	73.65	74.11	74.25	35.79	35.76	35.70	35.74
10^6	236.35	235.69	237.85	237.67	81.17	81.04	80.93	80.87

Table 2.5: Maximum values of v^*-component at $y^* = 0.5$ and u^*-component at $x^* = 0.5$ for different Ra numbers in water-filled cavity.

Table 2.5 presents the corresponding maximum values for the two velocity components. These values are comparable to those observed in air-filled cavities, but the gap between the results widens as the Rayleigh number increases. For lower Rayleigh numbers, the maximum values remain similar across all Prandtl number cases, highlighting minimal variation in flow dynamics under these conditions. For higher Ra the maximum values of both u^* and v^* considerably differ from those of Table 2.2 and 2.3.

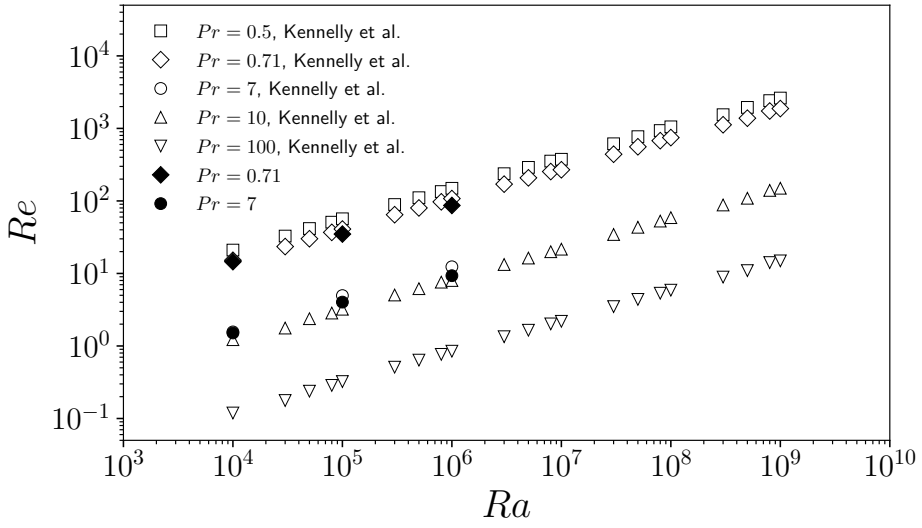

Figure 2.21: Reynolds numbers against Rayleigh numbers (10^4 to 10^9) for different Pr (0.5 to 100), as reported in [105]. Simulated cases are indicated with black-filled markers: $Pr = 0.71$ (diamond) and $Pr = 7$ (circle).

In their study, Kennelly et al. [105] plotted the Reynolds number as a function of varying Rayleigh and Prandtl numbers, as shown in Figure 2.21. Their results indicate that the Reynolds number increases with decreasing Prandtl number and increasing Rayleigh number. In Figure 2.21, the results

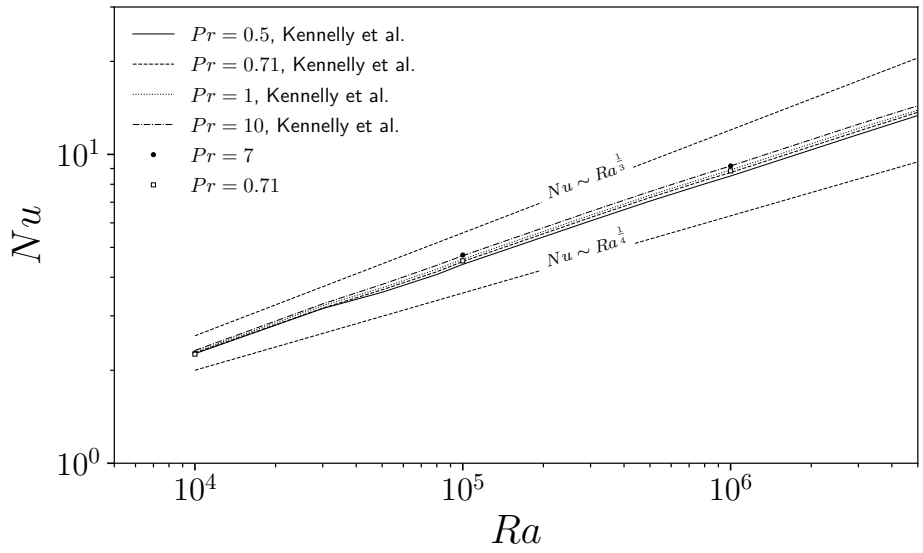

Figure 2.22: Nusselt number against Rayleigh number, as reported in [105]. Simulated cases are represented with circular markers: black-filled for $Pr = 7$ and empty for $Pr = 0.71$.

from the c_1 case are represented using filled markers (black). The results from the c_2 case are not reported as they do not differ from the c_1 simulation. It can be observed that, for both Prandtl numbers, the simulated results slightly underestimate the Reynolds number compared to the reference data. In contrast to the Reynolds number, the Nusselt number demonstrates a much weaker dependence on the Prandtl number. The variation in Nusselt values across different Prandtl numbers is relatively small, indicating that the heat transfer is influenced by the Rayleigh number rather than the fluid's Prandtl number. Figure 2.22 shows the behavior of the Nusselt values varying the Rayleigh and Prandtl numbers as in [105]. The observed dependence of the Nusselt on the Rayleigh number was found to be between a scaling behavior of $Ra^{1/3}$ and $Ra^{1/4}$. Furthermore, the simulated cases agree with the experimental and analytical results reported by the authors. This consistency supports the reliability of the numerical methods and assumptions used in the simulations.

Evaluation of the DHC Coupling Application Performance

For multiphysics simulations or CFD simulations involving a large number of mesh elements, computational cost becomes significant, and execution time is relevant. In coupling applications, the overhead introduced by data exchange needs to be considered in the total computation time of the simulation, as it plays a non-negligible role in such assessments. Implementing data transfers that do not rely on reading from files can improve computational cost, as reading/writing into output files is computationally expensive. For this reason, the data exchange procedures are performed online to reduce the overall computational effort. In this way, the coupling scheme does not rely on a code's generic output format but instead exploits direct access to memory data stored in MED structures.

Figure 2.23 shows the execution time per iteration throughout the entire simulation for the c_1 coupling case. For each time step, the figure displays the execution time used by FEMuS to solve the temperature equation, OpenFOAM to solve the Navier–Stokes equations, and the coupling application to perform data exchange between the two codes. The simulations are conducted with increasing mesh resolution: 20×20 (top left), 40×40 (top right), 80×80 (center left), 160×160 (center right) and 320×320 (bottom). As can be observed, even though the energy equation is relatively simple, the FEMuS solver requires an execution time much larger than the time that OpenFOAM requires in all five cases. On the other hand, the data exchange time is of the same order of magnitude as the OpenFOAM execution time.

As confirmation, Table 2.6 and Figure 2.24 report the total simulation times up to the convergence of the solution. For lower mesh resolution, most of the time ($\sim 50 - 60\%$) is dedicated to solve the energy equation using FEMuS, while less time is required for data exchange and the OpenFOAM solution. As the mesh resolution increases, the time required by the solvers to perform the simulation grows exponentially, whereas the data exchange time shows a linear trend with respect to the number of DOFs. The execution time referenced in Figure 2.23-2.24 and Table 2.6 corresponds to the application running in serial mode. However, both FEMuS and OpenFOAM support parallel execution via the MPI library, and the developed platform allows both codes to run in parallel. As a result, solver execution time can be significantly reduced through parallel computation. Nevertheless, as previously mentioned, data exchange between the solvers is currently handled in serial mode. Once each code completes its parallel execution, control is

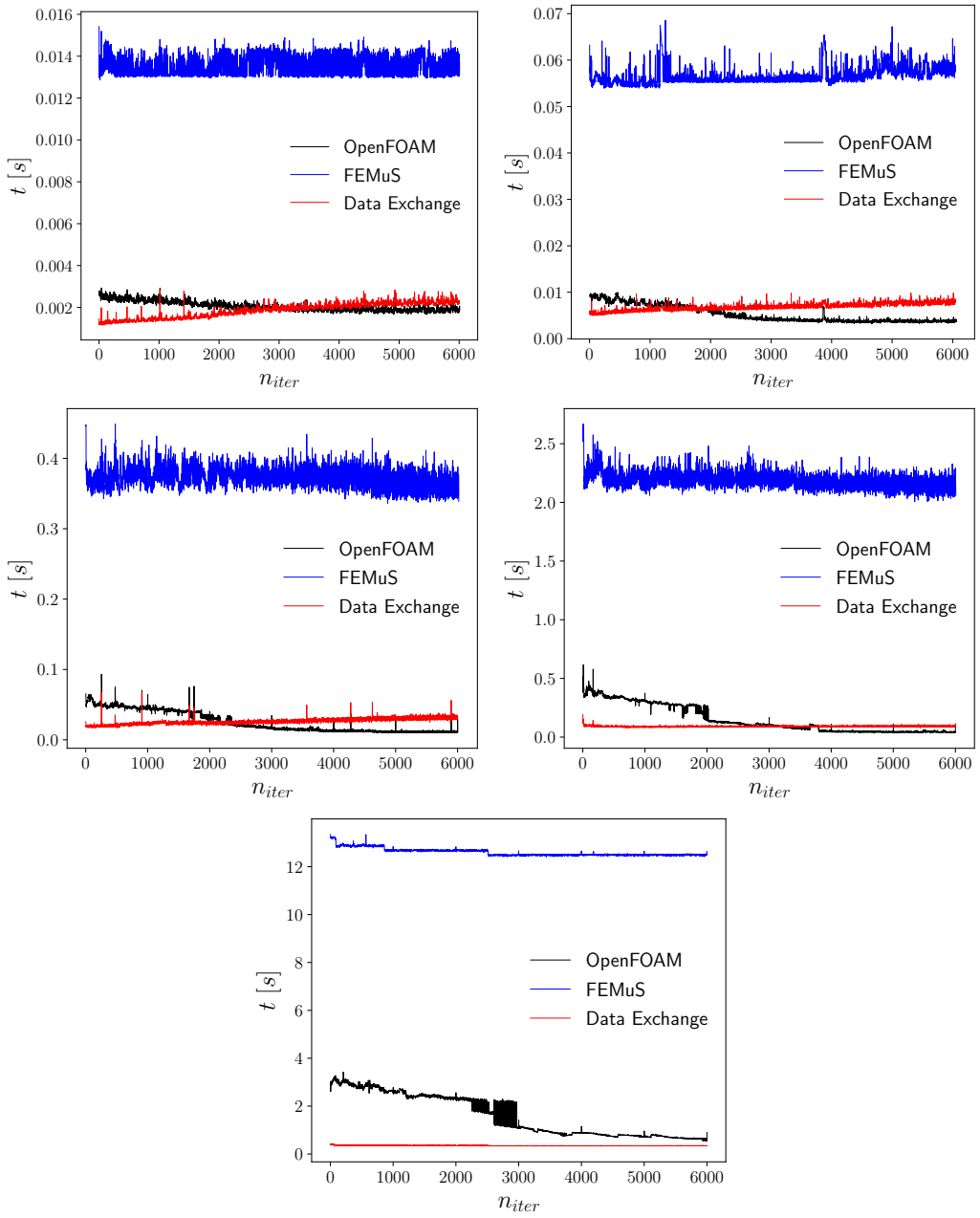

Figure 2.23: Execution time for c_1 coupling with varying mesh resolutions. OpenFOAM, FEMuS, and Data Exchange timings are shown for meshes of: 20×20 (left) and 40×40 (right) in the top row, 80×80 (left) and 160×160 (right) in the central row and 320×320 in the bottom row.

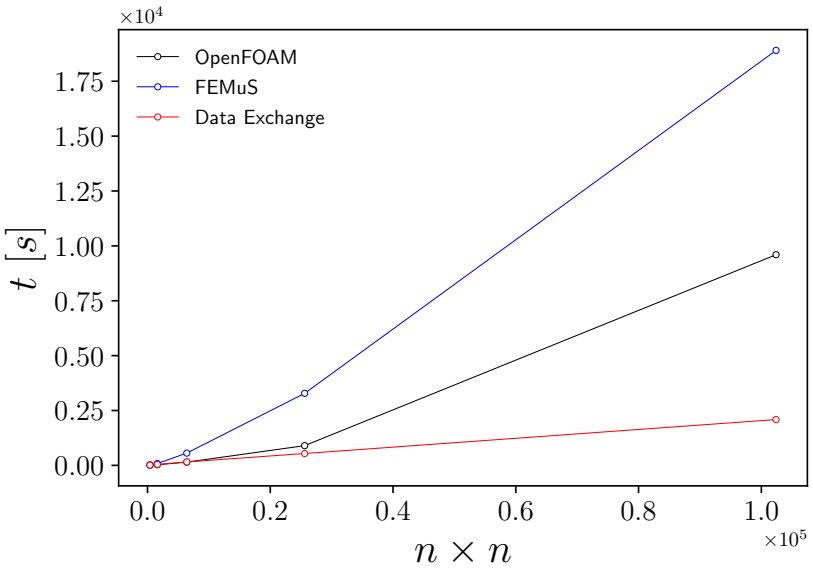

Figure 2.24: Total execution time to reach the converged solution for the c_1 coupling case simulation, varying mesh resolution.

	Time				
$n \times n$	20×20	40×40	80×80	160×160	320×320
FEMuS	41%	54%	65%	70%	62%
OpenFOAM	25%	20%	17%	19%	31%
Data Exchange	34%	26%	18%	11%	7%

Table 2.6: Time percentage over the total execution time, for the c_1 coupling case simulation.

returned to the supervisor, which manages the data exchange sequentially. Parallelizing the data transfer could also significantly reduce the coupling process's computational cost.

With regard to the convergence of the solution, Figure 2.25 shows the residual behavior of the two velocity components as a function of the number of iterations for both the coupled application c_1 and the monolithic OpenFOAM code, in the case of the air filled cavity with $Ra = 10^5$ and an 80×80 elements mesh. The coupling algorithm improves the solution's convergence

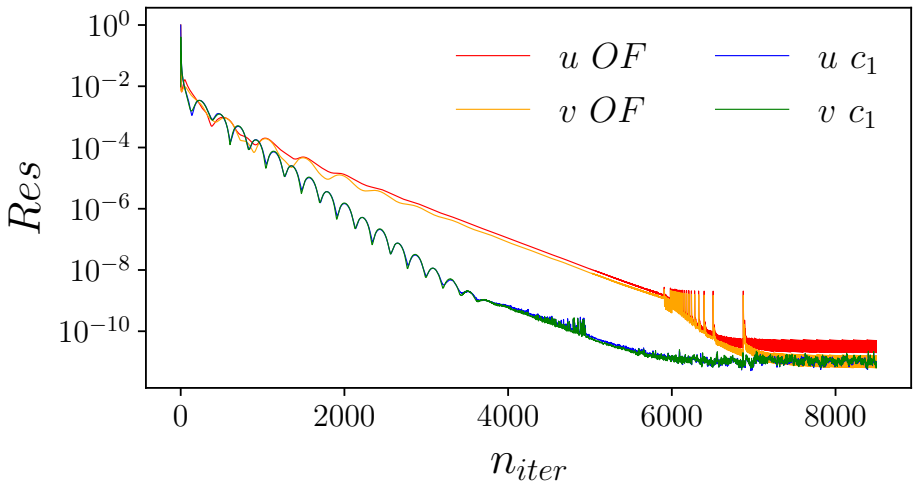

Figure 2.25: Residuals of the u- and v-velocity components as a function of the number of iterations for the OpenFOAM case and the c_1 coupling case.

behavior, achieving the convergence criteria after approximately 6000 iterations, compared to around 7000 iterations required by the standalone OpenFOAM simulation.

2.4.2 Conjugate Heat Transfer (CHT)

In this section, the second application implemented to test the data transfer between a FVM code and a FEM code is described. This test investigates the data transfer through the interface connecting two separate physical domains. In the context of heat transfer, these interfaces are classified as conjugate problems, where the so-called conjugate boundary condition can be applied [106, 107]. In the following, a Conjugate Heat Transfer (CHT) problem is analyzed, focusing on the thermal exchange between two regions composed of different materials. Specifically, the analysis considers heat transfer across a physical boundary between a solid and fluid regions.

Many practical applications are based on conjugate heat transfer problems whenever heat conduction in a solid region is closely coupled with convection heat transfer in an adjacent fluid region. Typical examples include heat transfer in cavities with thermally conducting walls, enhanced heat transfer with finned surfaces, or heat dissipation in high-performance devices. Many studies have focused on heat transfer between solid and fluid regions in the context of solar collectors and thermal energy storage systems, particularly

in concentrated solar power plants [108, 109]. Solar collectors and thermal energy storage systems are the two core components of solar power plants, making them a primary focus of research and development efforts in the field. The interaction between the solid collector and the working fluid is a critical aspect of collector design [110, 111, 112, 113], since the solar irradiation is absorbed by the solar collector and the heat is transferred to the working fluid through the domains interface. This thermal exchange is a classic example of a CHT configuration, where heat conduction in the solid and convection in the fluid are strongly coupled. Similarly, CHT plays an important role in optimizing heat extraction processes within thermal storage systems, where the main objective is to obtain an excellent heat transfer rate (absorb and release heat at the required speed) [114, 115]. Another key application area is heat exchanger systems, where numerous studies have adopted different methods to simulate heat transfer processes [116, 117, 118, 119, 120]. CHT is also crucial in nuclear energy production, particularly in safety analyses and operational assessments of nuclear reactors. The coupling of conduction and convection mechanisms is fundamental in ensuring reactor safety and efficient operation [121, 122, 123].

From the scientific computing perspective, the primary challenge in solving a conjugate heat transfer problem lies in addressing the strongly coupled interface domain. This interface is subject to both Dirichlet and Neumann boundary conditions simultaneously, reflecting the complex interaction between the solid and fluid regions. Different approaches have been developed over the years in order to tackle this kind of problem [124, 125, 126, 127]. Numerical solutions for conjugate conduction-convection problems are typically carried out within a unified computational domain that encompasses both the solid and fluid regions. In this context, SIMPLE-like algorithms are commonly used to solve the governing equations. However, the main difficulty is ensuring that the numerical solutions accurately capture the physical reality of the problem, particularly at the coupled interfaces. Proper representation of the boundary interactions and careful validation of the numerical methods are essential for achieving reliable and realistic results. For this reason, this section focuses on validating the multi-physics application developed to couple different domains through their interface. This approach enables the treatment of the CHT problem using two separate domains while applying a coupling algorithm to numerically model solid-fluid heat exchange.

The heat transfer in CHT systems is governed by two distinct physical

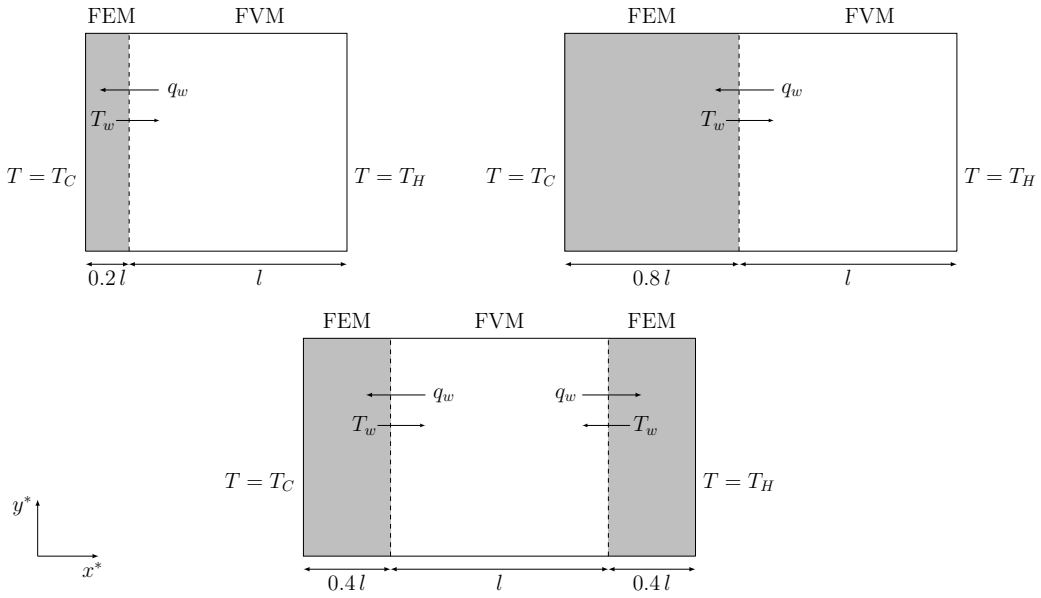

Figure 2.26: Geometrical configurations of the CHT problem: on the left the domain with the solid wall thickness equal to t_1, on the right equal to t_2 and on the bottom equal to t_3.

mechanisms. In the solid domain, conduction dominates, where the heat flows through the material driven by temperature gradients. In fluids, convection becomes the primary mode of heat transfer, which may occur naturally through buoyancy-driven forces or through forced movement caused by external drivers like pumps or fans. The configurations investigated for testing the boundary data transfer algorithm are reported in Figure 2.26 and refer to the cases described in [128]. The three configurations rely on two different domains connected where different equations are solved. Within the first region (white), the momentum and temperature equations are solved for a buoyant fluid, employing the same system of equations described in (2.10). The second region (gray) represents a solid domain in which only the temperature equation has been solved. The solid is modeled as a two-dimensional isotropic material with constant material properties where the only parameter that describes the temperature distribution is the thermal diffusivity α. This parameter is defined as

$$\alpha = \frac{k_s}{\rho\, c_p}\,, \tag{2.17}$$

where k_s is the thermal conductivity and c_p is the thermal capacity of the

solid domain. Therefore, the heat equation in the solid reads as

$$\frac{\partial T}{\partial t} = \alpha \Delta T \,. \tag{2.18}$$

The peculiarity of this kind of physical setup is the mutual exchange of the boundary conditions values at the interface between the two regions. At the interface, the fluid problem is defined by a non-homogeneous Dirichlet boundary condition, while the solid problem is defined by a non-homogeneous Neumann boundary condition. Specifically, the temperature field in the solid at the boundary is used as the boundary condition for the fluid region, while the heat flux computed at the same interface for the fluid region is the boundary condition for the temperature equation of the solid. This setup is commonly adopted for CHT simulations, as detailed in [129]. A schematic representation of the exchange of physical quantities can be seen in Figure 2.26. Here, the grey-colored solid region is graphically separated from the fluid one with a dashed line, through which the flux (wavy line, q_w) and the temperature (solid line, T_w) are exchanged.

For the solid subdomain, the other boundary conditions include two homogenous Neumann boundary conditions imposed at the top and bottom boundaries, while a fixed temperature is assigned on the external sides of the domain. The physical temperature field has been non-dimensionalized, as in the DHC problem. Thus, the temperature field T is transformed into the corresponding Θ by using (2.14). For the configurations shown at the top of Figure 2.26, a fixed temperature of $\theta = 0$ is imposed on the cold wall, which corresponds to the left vertical wall. In contrast, for the third case, both external vertical walls of the solid domains are treated with homogeneous Dirichlet boundary conditions. Specifically, the left wall is assigned a fixed cold temperature of $\theta = 0$, while the right wall is set to a fixed temperature of unity, $\theta = 1$.

In the fluid subdomain, the boundary conditions for the velocity field are the same as the ones described for the cavity in the previous section. For the temperature field, the top and bottom walls are adiabatic. In the configurations depicted at the top of Figure 2.26, the non-dimensional temperature Θ is fixed at 1 on the hot wall. The remaining walls serve as exchange interfaces with the solid subdomain.

Boundary data transfer algorithm

This section provides a description of the solution methods employed to implement a boundary data transfer algorithm within the coupling application framework. In the CHT problem, the boundary data transfer algorithm is necessary to replicate numerically the setups where the heat exchange between the fluid and solid subdomain is not uniform. As stated in the introductory part of the chapter, a possibility would be to solve the whole domain with a single code that implements both physics. However, the strategy of using well-established codes is adopted in this thesis for robustness and accuracy, so a suitable strategy must be developed to exchange data between FVM and FEM codes when they share a boundary region.

The method used for the coupling application involving boundary data transfer follows a Algorithm 4. As illustrated in Figure 2.26, for all the configurations simulated, the FVM code is employed to solve the natural convection flow within the cavity. On the other hand, the FEM code is used to solve the energy equation within the solid domain. Each code initializes its own mesh independently: the FEMuS code generates the mesh for the solid domain, while OpenFOAM creates the mesh for the fluid domain. The coupling class provides `init_interface()` and `create_mesh()` routines to create a MED mesh object representing the boundary interface for both codes. In this specific application, the goal is to couple the two regions within a 2D problem. Thus, the FEMuS interface structure consists of a 1D mesh of biquadratic Lagrangian edges. Similarly, the OpenFOAM interface class provides the information needed to construct a copy of the 2D boundary mesh. The 2D representation of the boundary mesh is required since OpenFOAM handles 3D meshes, even for 2D problems.

After the creation of the MED objects representing the interfaces and the meshes, the numerical fields to be exchanged are initialized. The temperature at the boundary and the wall heat flux through the same boundary are initialized as MED fields using `init_med_field_on_cells()`. Consequently, the MED fields `source_P0_T` and `target_P0_q` are created to store FEMuS data: a cell-wise temperature field for storing the computed temperature from FEMuS and a heat flux field for storing the interpolated data from OpenFOAM. Similarly, corresponding MED fields are initialized for OpenFOAM, namely `source_P0_q` and `target_P0_T`, serving the same purposes. The application of conjugate heat transfer does not require any mesh movement and the projection matrix can be computed before the time loop begins. However, other

Algorithm 4 Boundary data transfer algorithm.

procedure SUPERVISOR CONTROL
    Instantiate class objects
    Initialize boundary interfaces
    Create MED meshes and fields
    Create projection matrix `_remapper`

Time loop

    while `OpenFOAM_obj.run()` **do**
        Solve **u** and T with OpenFOAM fluid solver
        Retrieve `source_P0_q` from OpenFOAM boundary
        Compute `target_P0_q` from interpolation over FEMuS mesh
        Compute `target_P2_q` from `target_P0_q`
        Set `target_P2_q` as Neumann BC into FEMuS
        Solve T with FEMuS solid solver
        Retrieve `source_P2_T` from FEMuS boundary
        Compute `source_P0_T` from `source_P2_T`
        Compute `target_P0_T` from interpolation over OpenFOAM mesh
        Set `target_P0_T` as Dirichlet BC into OpenFOAM
    end while
end procedure

data transfer boundary applications, such as fluid–structure interaction, may involve changing meshes. In such cases, the interpolation matrix must be recomputed at every time step, as described in Algorithm 1.

The time loop start with OpenFOAM solving the governing equation for the fluid and the temperature equation using its own solver routines. The wall heat flux within the boundary representing the interfaces solid-fluid is computed and stored in `source_P0_q` MED field. Therefore, this field is first extracted from the solution of OpenFOAM using the routine `get_field_from_OpenFOAM()` and then stored in a MED field over the corresponding boundary mesh using `set_field()` routine. The heat flux provided by OpenFOAM is interpolated onto the target mesh to obtain the target field `target_P0_q` using the `interpolation()` function. This interpolation routine performs a $P_0 - P_0$ interpolation from the linear 2D mesh copy of the OpenFOAM mesh to the 1D linear mesh derived from the bi-quadratic FEMuS mesh. The `target_P0_q` field is then transferred to the

FEMuS solver as a non-homogeneous Neumann boundary condition using the `set_field_to_femus()` routine. It is important to note that a $P_0 - P_2$ interpolation is required before the solution is applied to the boundary, as the field provided by OpenFOAM uses a cell-wise approximation. The updated boundary condition is then used by FEMuS to solve the temperature equation within the solid domain, as described in (2.18). The temperature solution, `source_P0_T`, computed by the FEMuS solver in the solid domain, is retrieved at the boundary interface using the `get_field_from_femus()` routine. Through inverse mapping, this solution is first converted into a P_0 field and then interpolated onto the OpenFOAM boundary to create the `target_P0_T` field. The projection from OpenFOAM interface to FEMuS interface of the coupled temperature is computed using the already mentioned `interpolation()` routine. This field is applied as a Dirichlet boundary condition in OpenFOAM, where the non-homogeneous boundary temperature is updated using the `set_field_to_OpenFOAM()` routine. At this stage, control is turned back to OpenFOAM, where it continues the task of solving its equations in the following time step.

Simulations Results

In this section, the boundary data transfer algorithm is tested by using a Conjugate Heat Transfer problem. The physical and geometrical configurations are based on the work of Basak et al. [128], which is used as the reference for validation. Several physical and geometrical configurations were analyzed in [128], varying parameters such as the Prandtl number, the Rayleigh number, the conductivity ratio (K), the solid wall thickness (t), and its geometric position (on the hot or cold side). For this validation, the numerical simulations were conducted by varying the wall thickness, the conductivity ratio, and the Rayleigh number.

A schematic representation of the physical configurations used for the simulations is provided in Figure 2.26. Three geometric configurations are analyzed to account for different solid region layouts. The first configuration features a solid region with thickness $t_1 = 0.2l$. The second has a width of $t_2 = 0.8l$, where l is the side length of the square cavity. The third configuration includes two solid regions on opposite sides of the cavity, each with a thickness of about $0.4l$, resulting in a total combined width of $0.8l$. This third case is referred to as having a thickness of $t_3 = 0.4l \times 2$. The conductivity ratio K is defined as the ratio between solid and fluid thermal

conductivity as

$$K = \frac{k_s}{k_f}\,, \tag{2.19}$$

where the subscripts s and f refer to solid and fluid regions, respectively. In this study, three distinct values of K have been investigated: $K_1 = 0.1$, $K_2 = 1$, and $K_3 = 10$. These values allow us to analyze systems where the solid is significantly less conductive than the fluid (K_1), equally conductive (K_2), or significantly more conductive (K_3). Regarding the Rayleigh number, which quantifies the natural convection within the fluid, two representative values have been selected for the simulations: 10^3 and 10^5. These values correspond to different flow regimes, with $Ra = 10^3$ representing weaker convection and $Ra = 10^5$ indicating stronger convective flows. In contrast, the Prandtl number has been kept fixed at $Pr = 0.015$. This value corresponds to a low-Prandtl fluid, such as liquid metals. By systematically varying K, Ra, and wall thickness, this study provides a comprehensive assessment of the boundary data transfer algorithm under diverse thermal and flow conditions.

In Figures 2.27-2.30, the contour of the non-dimensional temperature Θ and the non-dimensional velocity stream function Ψ are reported for the simulated cases. The non-dimensional temperature can be set as follows

$$\Theta = \frac{T - T_0}{\Delta T}\,, \qquad \Psi = \frac{\psi}{\alpha}\,. \tag{2.20}$$

Specifically, Figure 2.27 illustrates the solutions for $Ra = 10^3$ with a wall thickness of t_1, showing the effect of the conductivity ratio variation on the temperature distribution and flow patterns. For the case with $K = 0.1$, as shown at the top of Figure 2.27, the isotherms are parallel to the side walls, indicating conduction-dominated heat transfer within the fluid. The lower thermal conductivity of the solid (grey-colored domain) leads to a steeper temperature gradient within the solid wall. Thus, we can observe a squeeze of the lines toward the solid region, which results in a temperature variation spanning from 0 at the left wall (cold wall) to 0.6 within the solid. In contrast, within the fluid phase, the temperature gradient is much smaller, ranging from 0.7 to 1 near the hot wall. This smaller gradient reduces the buoyancy force, resulting in weaker fluid flow. Therefore, the flow strength within the cavity is very weak, with the maximum magnitude of the streamfunction reaching only $\Psi = 0.4$. The circulation intensity is high near the center of the cavity. It decreases toward the walls due to the no-slip boundary conditions, as shown in the function contours in the upper right images of Figure 2.27.

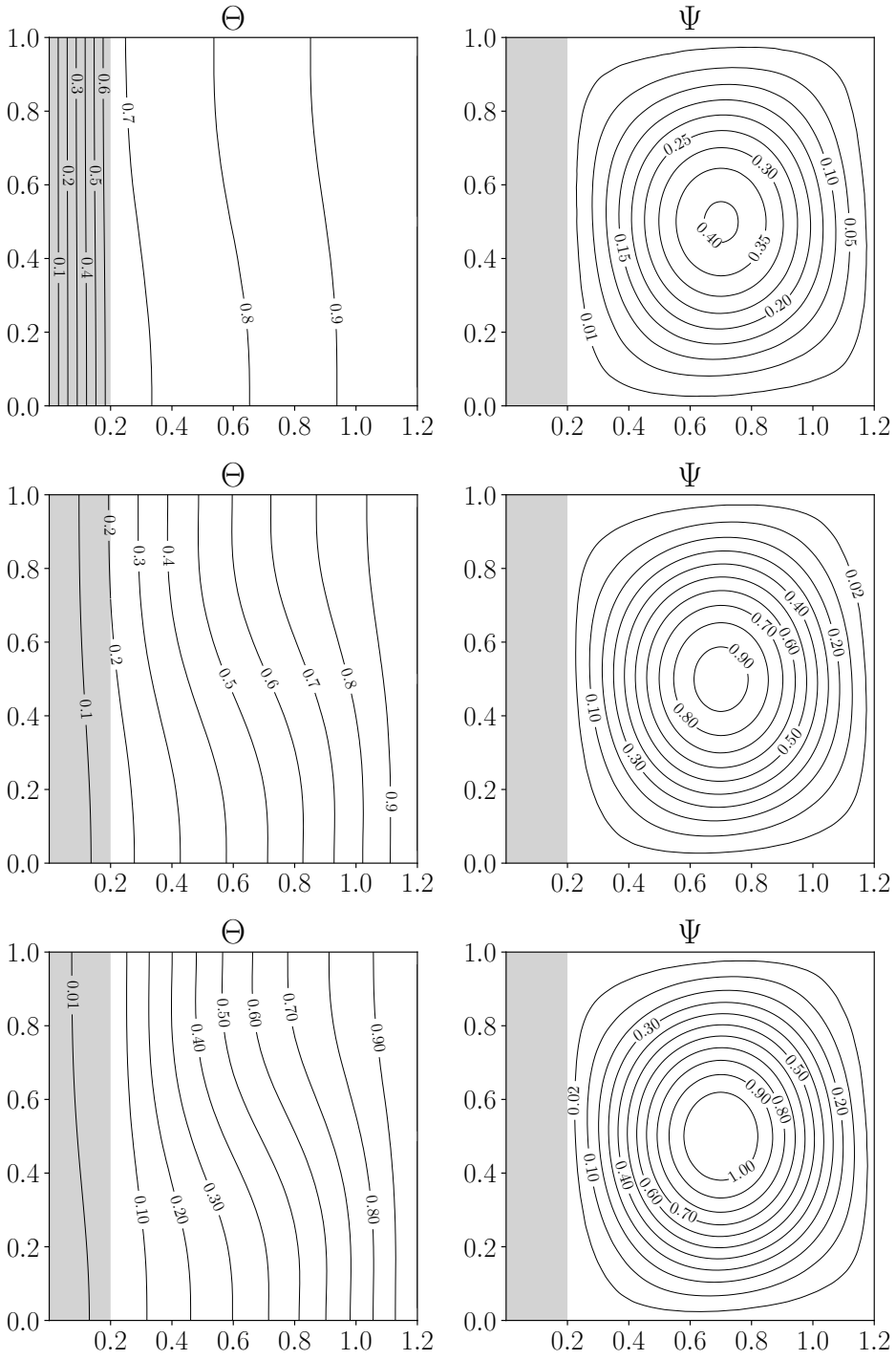

Figure 2.27: Simulations with solid thickness t_1 and $Ra = 10^3$. Contour of non-dimensional temperature Θ (left) and velocity stream function Ψ (right) for $K = 0.1$, 1, 10 (from top to bottom).

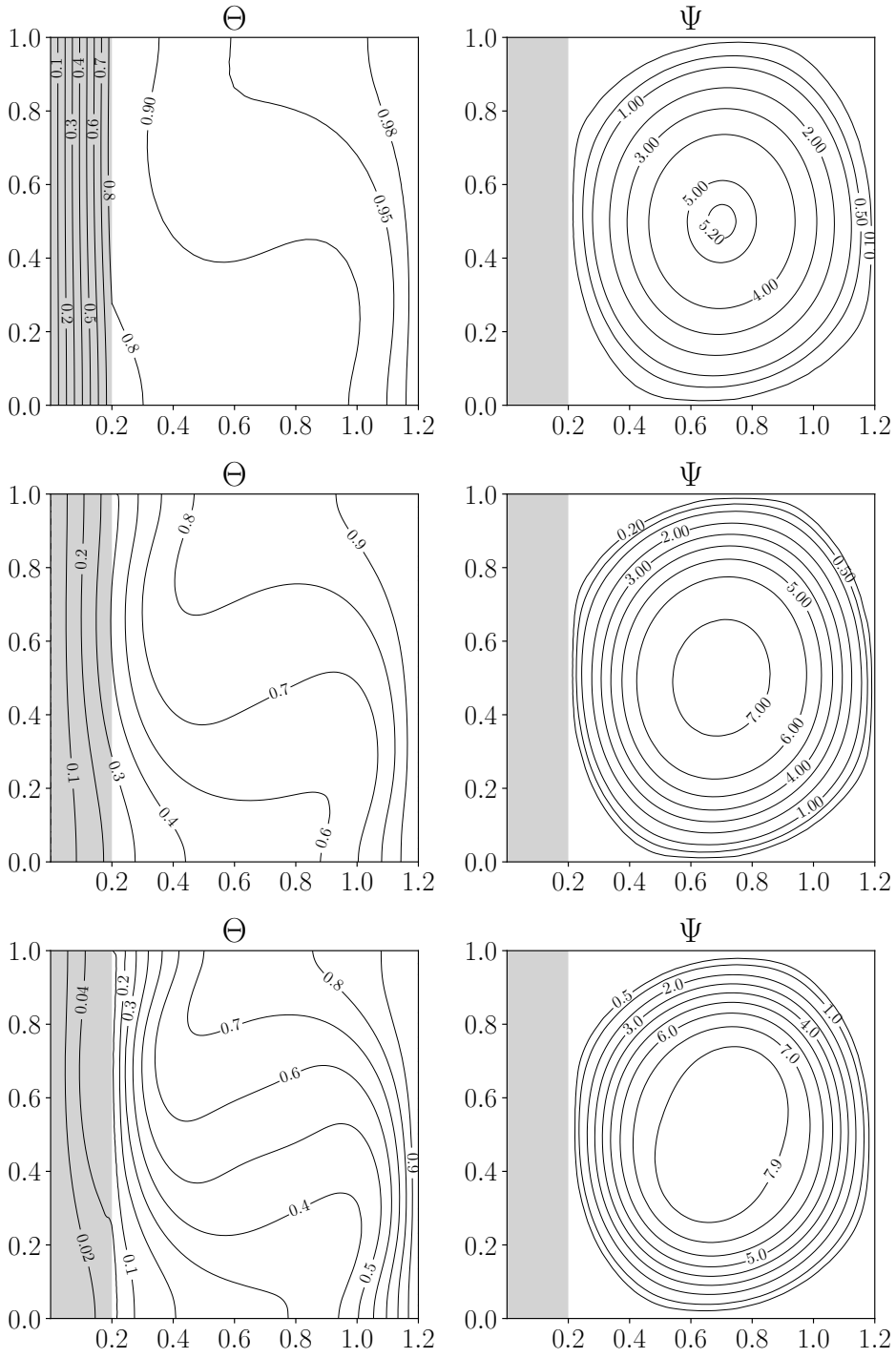

Figure 2.28: Simulations with solid thickness t_1 and $Ra = 10^5$. Contour of non-dimensional temperature Θ (left) and velocity stream function Ψ (right) for $K = 0.1$, 1, 10 (from top to bottom).

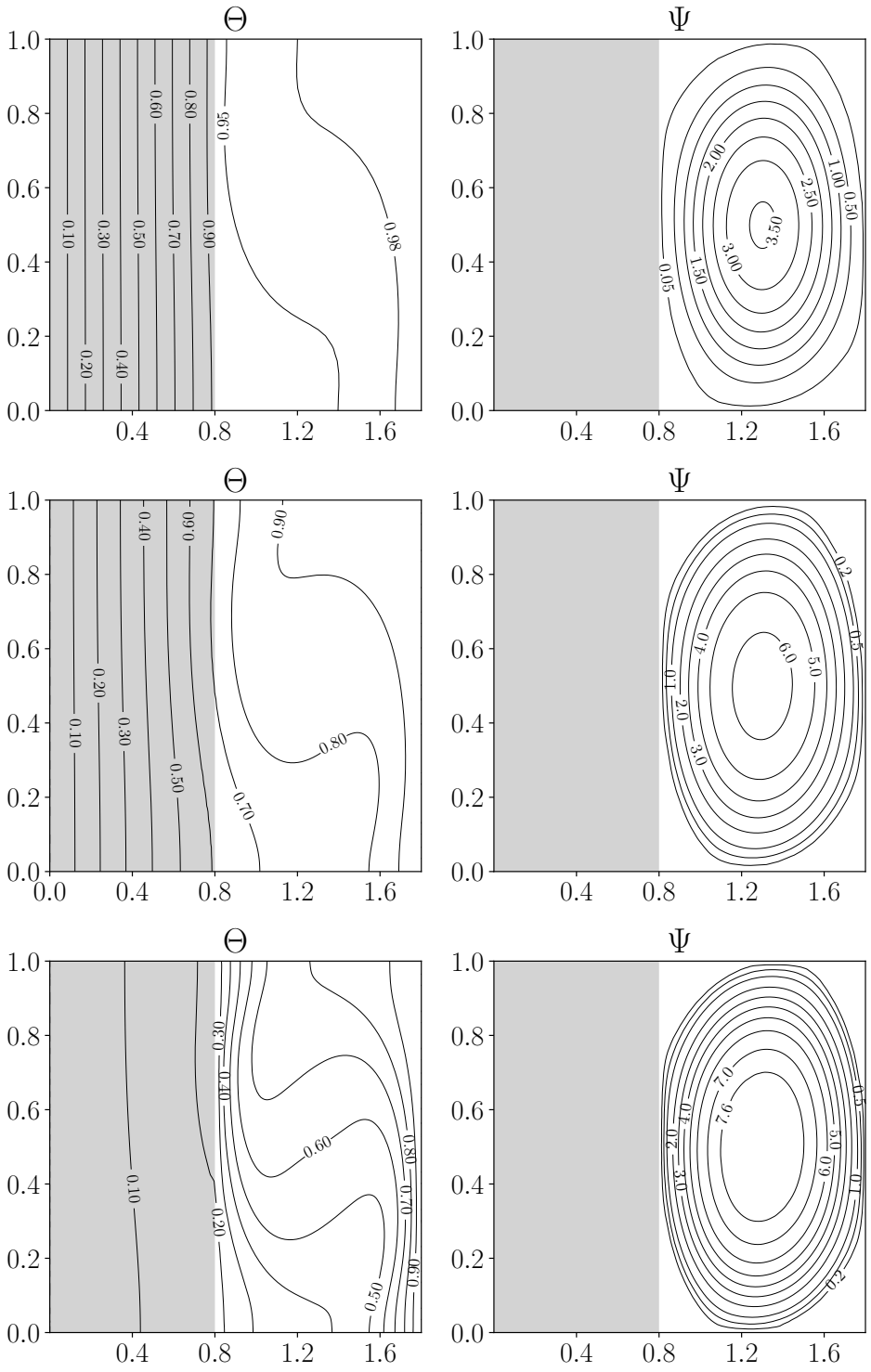

Figure 2.29: Simulations with solid thickness t_2 and $Ra = 10^5$. Contour of non-dimensional temperature Θ (left) and velocity stream function Ψ (right) for $K = 0.1$, 1, 10 (from top to bottom).

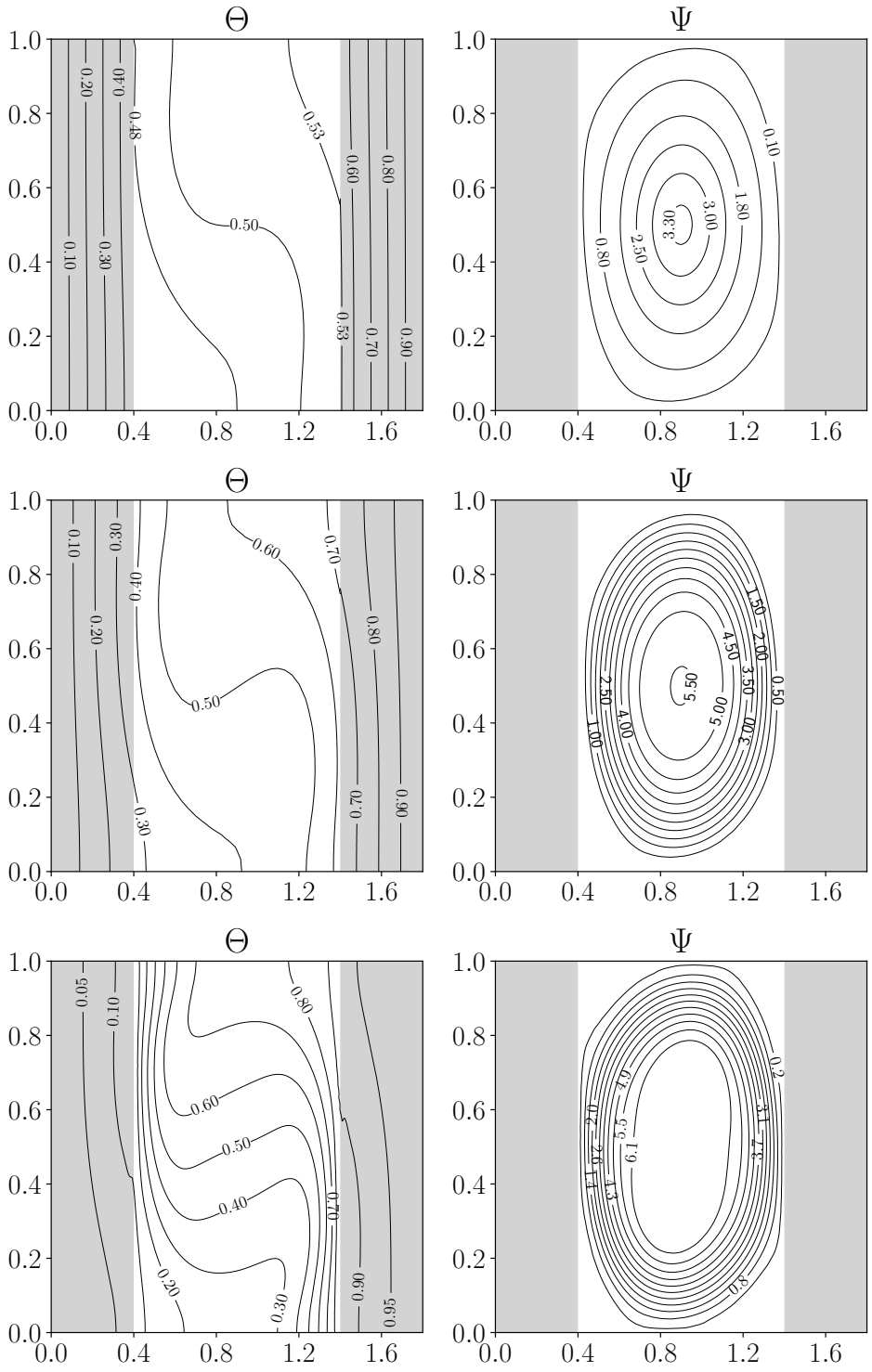

Figure 2.30: Simulations with solid thickness t_3 and $Ra = 10^5$. Contour of non-dimensional temperature Θ (left) and velocity stream function Ψ (right) for $K = 0.1$, 1, 10 (from top to bottom).

The circulation induced by the temperature difference is anticlockwise. This is consistent with the hot wall positioning on the right and the cold wall on the left. The buoyancy forces pull the fluid to rise along the hot right wall. It descends along the cold left wall, forming the observed circulation pattern.

Considering the non-dimensional temperature, the isolines behave depending on the conductivity ratio K. As already discussed, for $K < 1$, the temperature gradient is concentrated in the solid region. As the conductivity ratio increases, the temperature gradient shifts to the fluid region, where the higher conductivity enhances more efficient heat transfer. The middle section of Figure 2.27 illustrates a case with the same parameters previously described but with a conductivity ratio of $K = 1$. In this scenario, the non-dimensional temperature distribution appears uniform across the solid and fluid regions. This behavior occurs because both materials have equal thermal conductivities, resulting in an identical temperature gradient across the solid and fluid domains. Therefore, at the interface, the temperature gradient is continuous, unlike in the previous case. This temperature distribution leads to a stronger intensity of buoyancy forces within the fluid causing the isotherms to be slightly more curved. In this setup, the temperature difference in the fluid increases from 0.3 to approximately 0.8, resulting in an increase of the maximum value to 0.9. As the conductivity ratio increases, this effect becomes more evident, as shown in the patterns at the bottom of Figure 2.27. Specifically, for a conductivity ratio of 10, the temperature gradient shifts almost entirely to the fluid region, increasing from 0.8 in the previous case to approximately 0.95. This results in a corresponding intensification of the buoyancy effect.

Figure 2.28 presents the simulation results for $Ra = 10^5$ with a wall thickness of $0.2l$. Similar to the previous case, the figure illustrates the behavior of the non-dimensional temperature and stream functions as the conductivity ratio varies. The increase in the Rayleigh number to 10^5 results in a significant strengthening of the flow, causing greater distortion of the isotherms compared to the $Ra = 10^3$ case. In fact, as the Rayleigh number increases, the solution moves from temperature stratification towards a recirculation cell. Increasing K produces a similar effect to the previous case. Specifically, as the thermal conductivity of the solid increases, the temperature gradient within the solid region decreases, causing the gradient to shift further into the fluid region. This steeper temperature gradient in the fluid enhances the flow strength, which in turn leads to greater distortion of the isothermal lines.

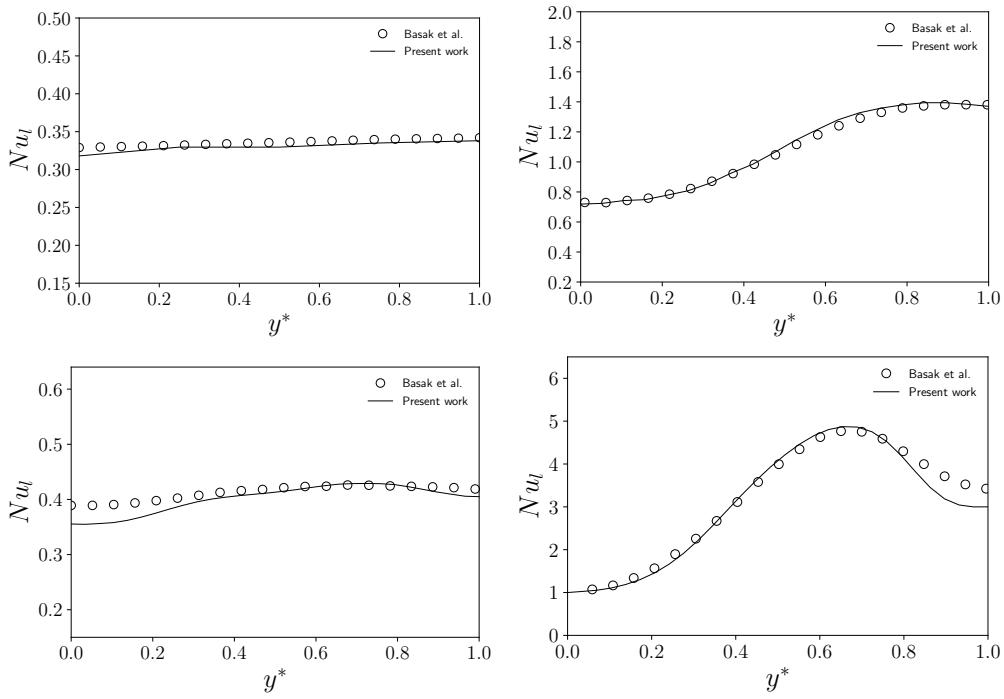

Figure 2.31: Local boundary Nusselt number for the cases with solid thickness t_1: solid lines are the simulations with $Ra = 10^3$ on top ($K = 0.1$ on the left and $K = 10$ on the right), $Ra = 10^5$ on the bottom ($K = 0.1$ on the left and $K = 10$ on the right). A comparison with data from [128] is reported (white circular markers).

Similar considerations apply to the variation of K in the simulations with t_2 wall thickness and $Ra = 10^5$, reported in Figure 2.29. As already observed in Figure 2.28, the temperature isolines are distorted in the fluid core, indicating convection-dominated heat transport. The variation in K results in a shift of the temperature gradient from the solid region to the fluid. The main difference compared to the cases with the same Rayleigh number but a wall thickness of t_1 is that, as the total wall thickness increases, the flow strength decreases. This reduction is due to lower buoyancy forces caused by a smaller thermal gradient. The behavior remains consistent when the total wall thickness is unchanged but applied to the third configuration. The solutions for the three values of K ($K = 0.1$, $K = 1$, and $K = 10$), with $Ra = 10^5$ and t_3, are shown in Figure 2.30.

In Figure 2.31, the local Nusselt number on the interface is reported for the case of the solid wall thickness equal to t_1. The Nusselt number has

Ra	t_1					
	K_1	[128]	K_2	[128]	K_3	[128]
10^3	0.332	0.335	0.898	0.890	1.08	1.08
10^5	0.412	0.412	1.897	1.907	3.269	3.162
Ra	t_2					
	K_1	[128]	K_2	[128]	K_3	[128]
10^5	0.118	0.117	0.850	0.851	2.556	2.578
Ra	t_3					
	K_1	[128]	K_2	[128]	K_3	[128]
10^5	0.117	0.117	0.859	0.855	2.575	2.586

Table 2.7: Average Nusselt numbers for different conductivity ratios K, varying the Ra number and wall thickness t. A comparison with results from [128] is also reported.

been computed as the normal gradient of the non-dimensional temperature on the interface. It represents the total ratio between the convective and the conductive heat transfer over the boundary interface. In particular, it can be expressed as

$$Nu_l = \frac{\partial \Theta}{\partial \boldsymbol{n}} \,. \tag{2.21}$$

A comparison with data from the reference literature is shown using circular white markers for the data from [128] and using a black solid line, which represents results obtained with the boundary data algorithm presented in this work. Overall, the estimation of the Nusselt number shows good agreement with the literature data. However, for $Ra = 10^5$ and both values of the conductivity ratio, there is a slight deviation in the results compared to the reference data.

In Table 2.7 the average Nusselt number $\overline{Nu_l}$ on the shared boundary between the two regions is reported, with a comparison of the same parameter presented in [128]. A good agreement with the literature data is achieved for every simulation.

Chapter 3

Turbulent Natural Convection of Liquid Metals

Natural convection of liquid metals has increased interest due to their wide range of applications, especially as excellent heat transfer fluids. Heat transfer in liquid metals is fundamental in many engineering processes, including high-performance cooling systems in next-generation nuclear reactors and solar energy collectors. For example, during the shutdown of a liquid metal nuclear reactor, heat transfer from the reactor core to the coolant occurs primarily through natural convection. Natural convection in liquid metals also occurs when the absorber plate in solar collectors is warmed by solar radiation, creating a temperature gradient between the absorber and the surrounding fluid. Therefore, understanding the flow and heat transfer behavior in the natural convection regime is paramount in liquid metal applications.

In recent years, there has been increasing interest in thermally driven flows in enclosures. Buoyancy-driven convection in cavities occurs in two fundamental forms. The first is Rayleigh-Bénard convection, where convective flow arises due to heating at the bottom and cooling at the top of an enclosure. The second is vertical convection, which occurs when the sidewalls of an enclosure are heated differentially, resulting in a configuration known as the differentially heated cavity. In this scenario, a non-zero temperature difference creates a circulation, with fluid rising along the hot wall, moving

across the top, descending along the cold wall, and returning across the bottom. The higher the temperature drop between the sidewalls, the higher the fluid's convection regime inside the cavity.

This chapter focuses on the low-Prandtl-number natural convection study within a square differentially heated cavity. First, a brief description of the available literature data is reported. This is followed by highlighting the key differences between the behavior of standard fluids (e.g., water and air) and non-conventional fluids characterized by low Prandtl numbers. The results obtained using the turbulence models described in Chapter 1 are then presented and compared with literature data. Simulations conducted with various OpenFOAM turbulence models are also discussed and evaluated against literature benchmarks. Finally, the chapter introduces simulations performed using the coupled approach, where the volume data transfer algorithm is applied as described in Chapter 2.

3.1 Literature Overview

The natural convection regime is determined by the Grashof number, $Gr = Ra/Pr$, which is the buoyancy ratio to viscous forces acting on a fluid. The Rayleigh number, Ra, is the dimensionless number that defines the ratio between diffusive and convective thermal transport phenomena, while Pr, the Prandtl number, is defined as the ratio of momentum diffusivity to thermal diffusivity. For a given Pr number, below a critical Ra, conduction dominates heat transfer. As Ra increases, buoyancy strengthens the flow, and the convection term becomes more important. Once Ra exceeds the critical threshold, the flow becomes unsteady, exhibiting periodic motion, and eventually transitions to turbulence. However, it is evident that the critical Rayleigh number alone does not define the transition limit, as the natural convection regime is closely dependent also on the Prandtl number.

In [105], Kennelly et al. analyzed the effect of Reynolds number on varying Prandtl numbers. As shown in Figure 2.21, the highest Reynolds number calculated for high-Prandtl-number fluids is approximately 2600, obtained for $Pr = 0.5$ and $Ra = 10^9$. These results are confirmed in [130] and [131], where Henkes et al. demonstrate that for ordinary fluids with a Prandtl number of 1, the flow becomes turbulent only at high Rayleigh numbers. For instance, in the case of natural convection for air-filled cavities, the critical Rayleigh is above 10^8. According to Incropera et al., the laminar natural

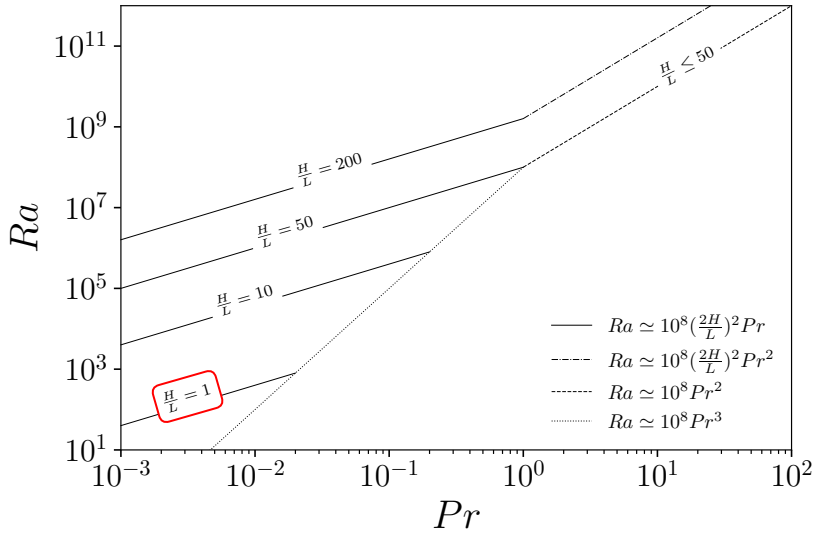

Figure 3.1: Transition boundaries from laminar (below the lines) to turbulent (above the lines) flow in cavities, as identified by Lage et al. [134]. The $Ra - Pr$ plot illustrates critical Rayleigh number as a function of Prandtl number for different aspect ratios.

convection at a local Rayleigh number larger than 10^9 may be promoted to turbulent transition [132]. In their DNS simulation, Paolucci et al. [133] detected the existence of a critical Rayleigh number between 10^8 and 2×10^8 for an airflow in a square cavity. The flows under these conditions undergo a periodic unsteady flow. In [133], they suggested that beyond the critical value, a Reynolds-averaged form of the Navier-Stokes and energy equations should be employed together with a turbulence model.

A study conducted by Lage et al. [134] demonstrates that the transition to turbulence depends not only on the Prandtl number and Rayleigh number but also on the aspect ratio of the cavity, defined as the ratio of cavity height (H) to base (L). In [134], the authors derived a correlation that identifies the critical Rayleigh number as a function of the Prandtl number and the aspect ratio of the cavity. This correlation is illustrated in Figure 3.1, presented as a $Ra - Pr$ plot. The lines delineate two regions: the area below the lines corresponds to the laminar natural convection regime, while the area above is characterized by turbulent flow. As the cavity aspect ratio decreases, the critical Rayleigh number also decreases, reaching its lowest values for square cavities ($H = L$). The boundary lines marking the transition from

laminar to turbulent flow for square cavities are represented by a dashed line for high Prandtl numbers and a dotted line for Prandtl numbers < 1. For lower Prandtl numbers fluids ($Pr < 0.025$), the threshold is represented by the solid line labeled with $H/L = 1$ and highlighted in red. According to this study, high-Prandtl-number fluids begin the transition to turbulence at a Rayleigh number exceeding 10^8, whereas low-Prandtl-number fluids start the transition at much lower Rayleigh numbers $Ra \leq 10^5$. As a result, maintaining laminar flow in liquid metal enclosures is challenging, as the flow often becomes turbulent even at relatively low Rayleigh numbers.

However, the turbulent regime in this kind of fluid has received limited attention and remains a subject of ongoing research. High-fidelity studies of turbulent heat transfer in low-Prandtl liquid metals have primarily focused on Rayleigh-Bénard convection. Notable examples include works by Zwirner et al., Zürner et al., and Vogt et al. [135, 136, 137]. On the contrary, studies on differentially heated cavities with liquid metals at high Rayleigh numbers are very limited, and most available data are restricted to laminar flow conditions, with few studies addressing the transition to turbulence. The available literature data for the specific case of a square cavity at high Rayleigh numbers with low Prandtl number fluids is limited to the studies [138], [139], [140], [141], [142] and [143].

Bawazeer et al. in [138] conducted a detailed analysis of the flow structure in low-Pr regimes using the multi-relaxation time lattice Boltzmann method. Their study mapped flow regimes in the $Ra - Pr$ domain for steady and unsteady flows. In particular, they found that the Prandtl number and the critical Rayleigh number are related by the following correlation

$$Ra = 2.8727 \times 10^8 Pr^{2.0502}, \tag{3.1}$$

which was obtained by interpolating the critical Ra from their simulations results. Figure 3.2 displays the critical Rayleigh numbers identified by Bawazeer et al., along with the correlation from Equation 3.1. This line divides the parameter space into two distinct zones: the region below the critical Rayleigh line represents steady solutions, while the region above the line represents unsteady solutions. As we can see from Figure 3.2, decreasing the Prandtl number results in a considerable reduction of the critical Rayleigh number. In [138], the authors attribute this behavior to the low kinematic viscosity of liquid metals, which causes the advection term in the Navier-Stokes equations to dominate over the viscous term. Consequently, flow instability arises

at relatively low values of the controlling parameter Ra, in contrast to fluids with higher Prandtl numbers.

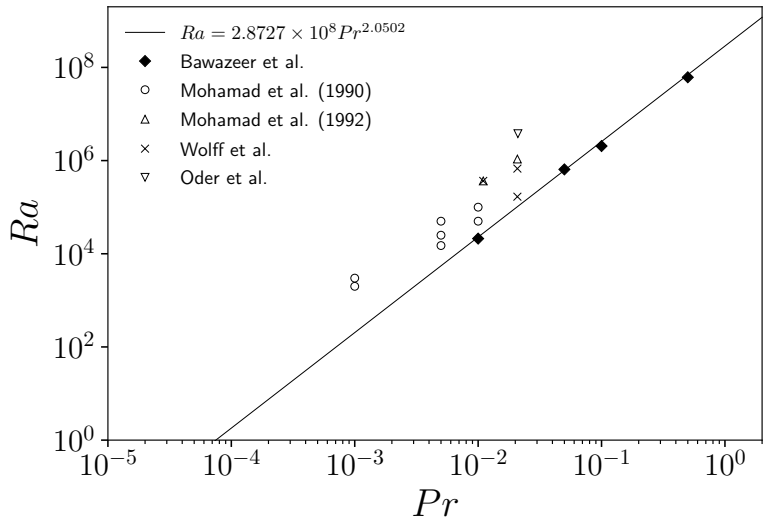

Figure 3.2: Correlation by Bawazeer et al. [138] (Equation 3.1) (black line), marking the transition from steady (below) to unsteady (above) flow regimes in low-Prandtl fluids. Black diamond markers refer to the simulation cases conducted in [138]. Circle markers denote critical values predicted by Mohamad et al. [141]. Triangles point out cases studied by Mohamad et al. [142], and crosses show configurations analyzed by Wolff et al. [139]. DNS case from [143] is indicated using triangle down markers.

The transition to turbulent buoyant flow with low Prandtl number fluids was also explored by Mohamad and Viskanta in [141]. They investigated transient natural convection in a cavity with Pr values ranging from 0.001 to 0.01 and Gr numbers up to 10^7 using finite difference methods. The authors solved the conservation equations without considering turbulence terms, focusing instead on analyzing the oscillatory behavior of the system under different conditions. Specifically, they predicted the critical Grashof numbers for three different Prandtl numbers, 0.001, 0.005, and 0.01, at which the flow begins to exhibit periodic oscillations. The simulations performed by Mohamad et al. are shown in Figure 3.2 as white circle markers. Their findings indicate oscillatory behavior for Grashof numbers greater than 2×10^6, 3×10^6, and 5×10^6 for Prandtl numbers of 0.001, 0.005, and 0.01, respectively. In Figure 3.2, the three critical Rayleigh numbers correspond

to the three circle markers with the lowest Rayleigh number values. As we can see, the critical values predicted by Mohamad et al. are slightly higher than those estimated by Bawazeer et al..

In a recent study, Mohamad et al. [142] analyze the turbulent buoyant flow of low Prandtl number fluids at Rayleigh number values exceeding the critical thresholds identified in their earlier work [141]. In their study, Mohamad et al. employed a low Reynolds number $k-\varepsilon$ model, as described in [144], to predict the behavior of turbulent natural convection for low Prandtl number fluids. They compared their simulation results with experimental data and direct numerical simulation findings. They reported temperature profiles in two different cases, see triangular marker in Figure 3.2. The former case involved a gallium-filled cavity ($Pr = 0.0208$) at a Rayleigh number of $Ra = 1.08 \times 10^6$, where the authors compared their simulation results with those from previous work by Viskanta et al. [140]. The latter case referenced experimental studies conducted by Wolff et al. [139], who measured fluctuating temperature profiles in a liquid tin-filled cavity ($Pr = 0.011$) at a Rayleigh number of $Ra = 3.66 \times 10^5$. Wolff et al. [139] conducted a combined experimental and numerical study for two additional configurations to investigate the influence of the Rayleigh number in a gallium-filled cavity. They performed both simulations and experiments at Rayleigh numbers of $Ra = 1.68 \times 10^5$ and $Ra = 6.73 \times 10^5$, see cross marker in Figure 3.2. Finally, the most comprehensive reference work in the literature was provided by Oder et al. in [143]. They presented results from a direct numerical simulation using a high-order spectral element method to investigate natural convection in a square cavity for low-Prandtl-number fluids. They solved the Navier-Stokes equations using the spectral element method implemented in nek5000 v 19.0 and performed three separate simulations at a constant Grashof number of 1.8×10^8, considering three different Prandtl numbers: 0.021, 0.1, and 0.2.

3.2 Characterization of the Flow

For low-Prandtl-number fluids, reaching the turbulent regime requires a lower Rayleigh number compared to fluids with higher Prandtl numbers. For such fluids, as the Grashof number increases, the fluid motion transitions from steady to unsteady, eventually becoming turbulent. To verify this behavior, the stability of the solution has been investigated by comparing its oscillatory characteristics with the results reported by Mohamad et al. in [141]

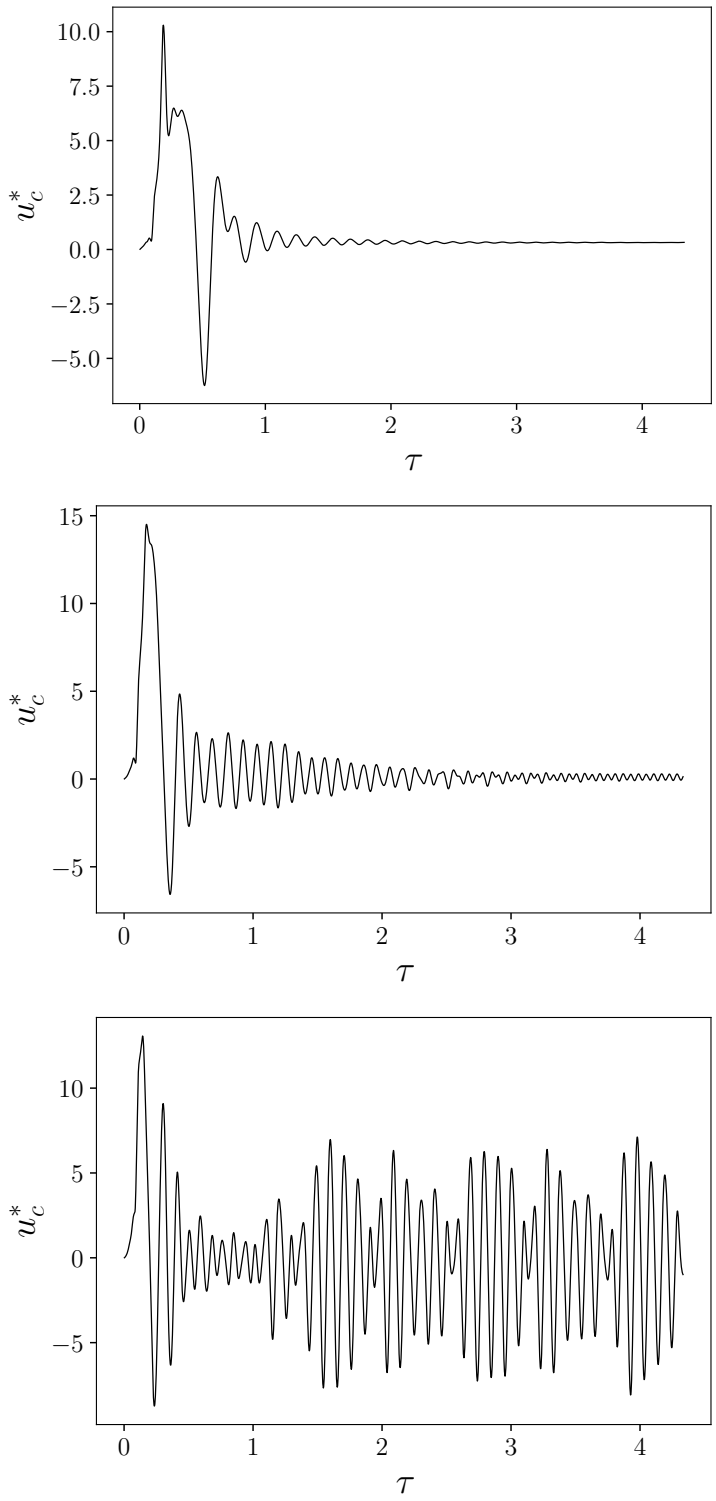

Figure 3.3: Transition to unsteady behaviour of non-dimensional velocity at the center of the cavity over time, varying the Rayleigh number, 2×10^4 (top), 5×10^4 (center), and 10^5 (bottom) with a fixed Prandtl number of 0.01.

and Bawazeer et al. in [138]. The stability analysis has been carried out using both the OpenFOAM and FEMuS laminar buoyant solvers, based on the same configuration shown in Figure 2.10. For details regarding the mesh, system of equations, and boundary conditions, the reader is referred to Section 2.4.1. The fluid properties have been adjusted to represent a liquid metal, while variations in the Rayleigh number have been achieved by modifying the geometry dimensions, keeping the non-dimensional temperature difference between the side walls fixed at 1. Figures 3.3 illustrate the transition to the unsteady solution in a sodium-filled cavity ($Pr = 0.01$) as the Rayleigh number increases. In Figures 3.3, u_c^* represents the dimensionless velocity at the center of the cavity, and τ denotes the non-dimensional time, obtained by dividing the time variable by L^2/α. Oscillatory behavior is observed to begin at a Rayleigh number between 2×10^4 and 5×10^4, corresponding to Grashof numbers of 2×10^6 and 5×10^6, respectively. Our results estimate the critical Rayleigh number at a lower value than that reported by Mohamad et al. in [141], but they show better agreement with the findings of Bawazeer et al.[138], who identified the critical range to be between 10^4 and 5×10^4 (or Gr between 10^6 and 5×10^6). For this reason, to observe turbulent natural convection in low-Prandtl number fluids using FEMuS and OpenFOAM solvers, the Grashof number should be at least greater than 5×10^6.

Case	Prandtl	Rayleigh	Grashof	Reference
Simulation 1	0.0210	3.78×10^6	1.8×10^8	[143]
Simulation 2	0.0208	1.08×10^6	5.2×10^7	[140, 144]
Simulation 3	0.0110	3.66×10^5	3.3×10^7	[139, 144]

Table 3.1: Parameter values (Prandtl, Rayleigh and Grashof numbers) for the simulated cases.

In the following, we present a brief overview of the main characteristics of turbulent flows in square cavities, supported by simulation results from three cases consistent with the limited data available in the literature. The first case corresponds to the study reported in [143], specifically for $Pr = 0.021$ and a Grashof number of 1.8×10^8. The other two cases are taken from [144] and [139]. The first is characterized by $Pr = 0.0208$ and a Rayleigh number of 1.08×10^6, while the second is characterized by $Pr = 0.011$ and a Rayleigh number of 3.66×10^5. The simulations performed are summarized in

Table 3.1 and show three different Grashof number values. All three Grashof numbers are significantly above the unsteady threshold previously identified using the OpenFOAM and FEMuS solvers. This condition ensures the onset of turbulent behavior.

Figure 3.4 shows the streamlines for the three simulated cases, illustrating and comparing the flow patterns of the simulations. As observed, all three cases exhibit a large, concentric, clockwise-rotating convection cell, similar to the laminar simulations presented in Chapter 2. In addition to the primary vortex that dominates the cavity, secondary and tertiary vortices form in the corners and at the center of the main cell. This behavior is due to the peculiar characteristics of low-Prandtl number fluids. In particular, by focusing on the differences between high-Prandtl-number fluids $Pr > 1.0$ and low-Prandtl-number fluids $Pr < 1.0$, we can explain the flow patterns observed in natural convection in enclosures. In the natural convection regime, the thermal boundary layer for high-Prandtl-number fluids has a comparable or smaller thickness compared to the velocity boundary layer. The temperature gradient is higher within the thermal boundary layer, confining buoyancy forces to this region of the domain. On the other hand, the fluid motion outside the thermal boundary layer remains passive. Thus, at high Prandtl numbers, the fluid flow forms only a single main circulating cell inside the enclosure. As the Rayleigh number increases, the thermal boundary layer thickness decreases, the high-velocity region becomes confined near the boundaries, and the flow in the core of the enclosure becomes nearly stagnant. Additionally, the core region of the flow becomes thermally stratified in the vertical direction, as we can see from Figures 2.16, with hot fluid at the top and cold fluid settling at the bottom. This stratification results in a heat transfer primarily driven by conduction rather than convection.

For fluids with $Pr < 1$, the thermal boundary layer can extend significantly away from the heated and cold walls, potentially reaching the core of the enclosure. In this case, the buoyancy effects can influence the behavior of the fluid within the entire cavity. The presence of buoyancy within the center of the cavity can induce the formation of secondary vortices. Figure 3.5 (left) and Figure 3.6 (left) show zoomed-in views of the cavity center, illustrating the streamlines and the rotational direction of the flow for the secondary vortices. As observed in these two simulations, an anti-clockwise recirculating cell forms at the center of the cavity. The third case does not exhibit the same behavior, likely due to its lower Grashof number. In fact, com-

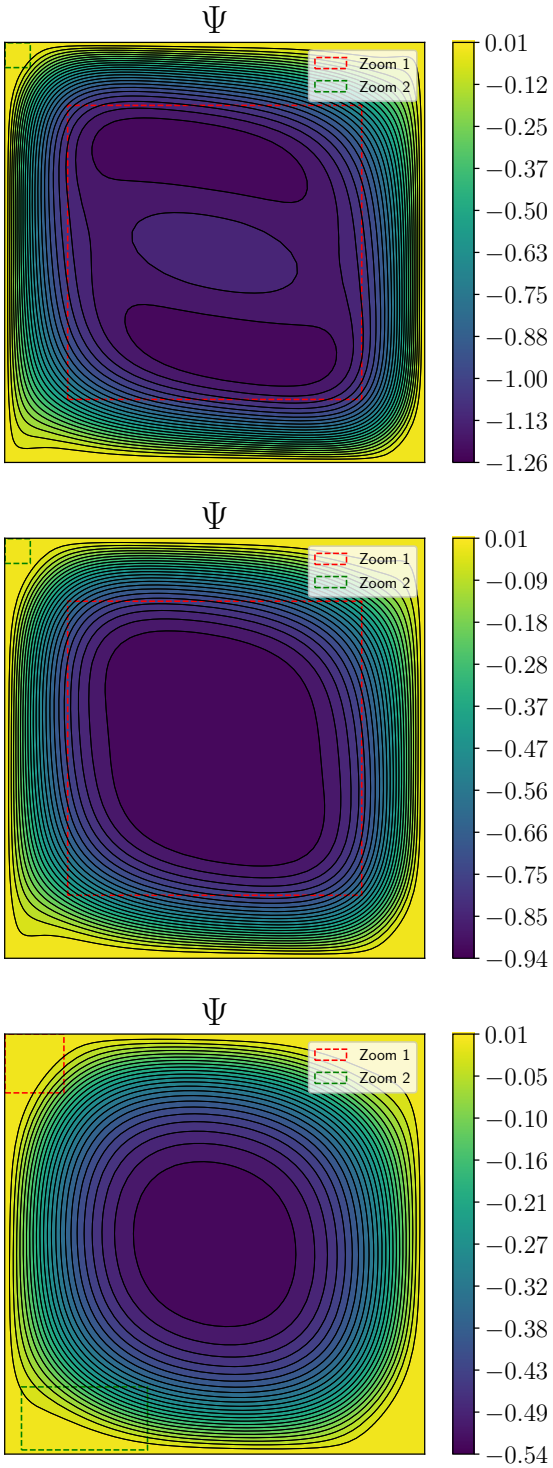

Figure 3.4: Stream functions for the simulations with $Pr = 0.021$ and $Ra = 3.78 \times 10^6$ (top), $Pr = 0.0208$ and $Ra = 1.08 \times 10^6$ (middle), and $Pr = 0.011$ and $Ra = 3.66 \times 10^5$ (bottom).

paring the three cases, it seems that a higher Grashof number enhances the intensity of the inner anti-clockwise circulation: decreasing its value suggests weaker buoyancy forces relative to viscous forces, which limits the formation or intensity of secondary vortices.

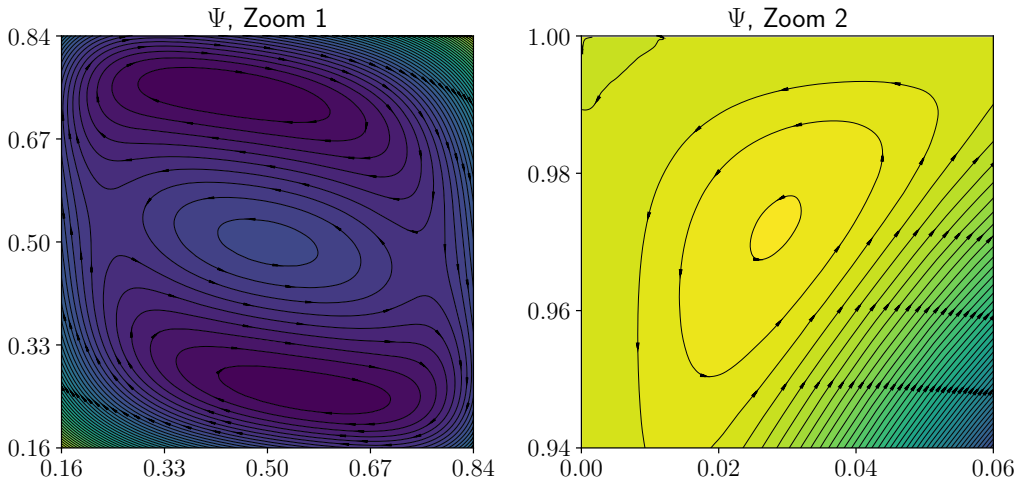

Figure 3.5: Zoomed-in views of Simulation 1: central zone (left) and upper left corner (right) secondary recirculation cells.

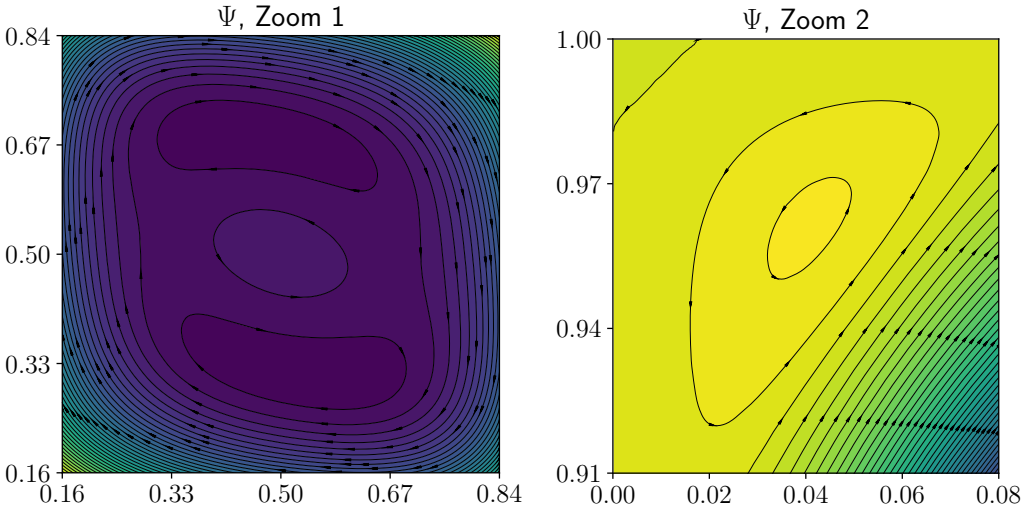

Figure 3.6: Zoomed-in views of Simulation 2: central zone (left) and upper left corner (right) secondary recirculation cells.

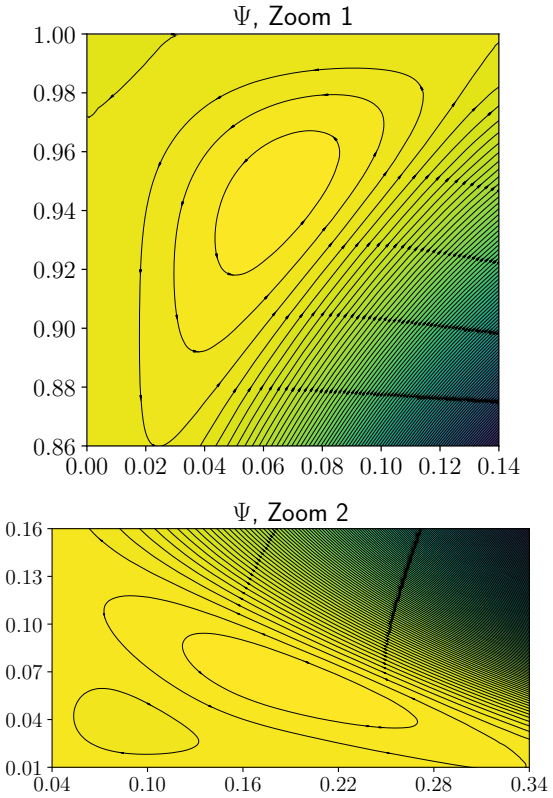

Figure 3.7: Zoomed-in views of Simulation 3: upper left corner (top) and lower left corner (bottom) secondary recirculation cells.

As can be seen from Figures 3.5, 3.6 (right) and Figures 3.7, buoyant flow in liquid metals is also characterized by the formation of weak secondary circulations in the corners of the cavity. In particular, cases with higher Grashof numbers exhibit secondary circulation only in the upper-left and lower-right corners. In contrast, the third simulation, characterized by a lower Grashof number, displays small recirculation cells in all four corners of the enclosure. At the corners, the main flow detaches from the side walls and turns at a 90-degree angle, leading to the development of these localized circulations. Larger recirculation zones develop in the top-right and bottom-left corners of the enclosure, in contrast to the smaller regions in the other corners. This behavior arises because the fluid velocities near the hot and cold vertical walls are higher than those along the horizontal top and bottom walls, causing flow separation to occur earlier in the upper-right and lower-left corners compared to the lower-right and upper-left corners. This is clearly illustrated

in the bottom images of Figure 3.4, where the extension of the zoomed-in view in green, labeled as Zoom 2, is noticeably larger than the red zone in the upper left corner. The larger green-marked zone, shown in detail in Figure 3.7 (bottom), comprises two distinct convection cells, whereas the upper-left and lower-right corners each contain a single, smaller eddy. Due to the presence of secondary recirculation cells, the impact of the main flow on all four walls of the enclosure does not occur next to the corners, as would typically be expected in natural convection of a high-Prandtl-number fluid within the same Rayleigh number range, see Figures 2.12. Instead, the points of collision are shifted vertically along the side walls and horizontally along the top and bottom walls. For the simulated cases, no stream function patterns are available in the literature for direct quantitative comparison. Nevertheless, the streamline from the first two simulations, shown in the top and center panels of Figure 3.4, can be qualitatively compared with those reported in [144]. Both the simulations and the literature reference are characterized by higher Grashof number values and show very similar flow patterns. On the other hand, Simulation 3 with a lower Grashof number can be compared with the streamlines reported in [139]. The reference from the literature for lower Grashof numbers exhibits very similar behavior to that shown in the bottom image of Figure 3.4.

3.3 Simulation Results

This section presents a comparative analysis of the three aforementioned cases. Specifically, the simulation results, obtained using three different approaches, are reported and compared with the available literature data. Firstly, the simulations performed using the FEMuS monolithic solver are presented. The second approach uses the monolithic code OpenFOAM with its built-in turbulence model. Finally, the third approach aims to leverage the strengths of both codes by combining them in a coupled application, using the volume transfer algorithm introduced in Chapter 2.

The configuration and computational domain used in these simulations are the same as described in Section 2.4.1, with physical parameters adjusted for low-Prandtl-number fluids, according to the non-dimensional simulation parameters of Table 3.1. Based on the geometry configuration shown in Figure 2.10, a no-slip condition is applied to all sides of the cavity. Dirichlet boundary conditions are imposed on the vertical walls for the temperature

field, with the left side set to a hot temperature and the right side to a cold temperature. Instead, the top and bottom walls have a homogeneous Neumann boundary condition imposed for the temperature field. The boundary conditions for the other turbulent variables are detailed in the following sections, as each solver uses its own set of boundary conditions for these variables. The simulations have been conducted using a refined mesh of

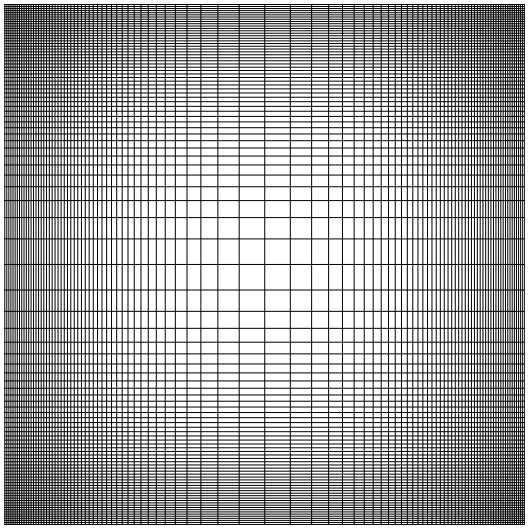

Figure 3.8: Mesh grid of 100×100 elements used in all the simulations, with a refined distribution near the walls.

100×100 elements, as shown in Figure 3.8. This fine resolution ensures that the turbulent parameter y^+ is maintained below 1 for all cases so that the first computational node from the wall lies within the viscous sublayer of the boundary layer (linear region). By maintaining $y^+ < 1$, the mesh captures the steep gradients in velocity and other flow variables near the wall without relying on empirical wall functions.

All the comparisons are presented using dimensionless variables. The non-dimensional magnitude of the velocity field is obtained using the following quantity

$$u_0 = \sqrt{\beta g \Delta T L}, \tag{3.2}$$

where β is the coefficient of thermal expansion, g is the gravitational acceleration, ΔT is the temperature difference between the vertical walls, and L is the side length of the cavity. Thus, the non-dimensional velocity is given by

$$U^+ = \frac{\sqrt{u^2 + v^2}}{u_0}, \tag{3.3}$$

where u is the horizontal component and v is the vertical component of the velocity field. Using this velocity reference value u_0, the dimensionless turbulent kinetic energy can be computed as

$$k^+ = \frac{k}{\frac{u_0^2}{2}} = \frac{2k}{\beta g \Delta T L}\,. \tag{3.4}$$

The non-dimensional temperature field is given by

$$\Theta^+ = \frac{T - T_{cold}}{\Delta T}\,, \tag{3.5}$$

where $\Delta T = T_{hot} - T_{cold}$. Given both the velocity reference value (3.2) and the temperature difference, the components of the turbulent heat flux are non-dimensionalized as

$$\langle \mathbf{u}'T' \rangle^+ = \frac{\langle \mathbf{u}'T' \rangle}{u_0 \Delta T}. \tag{3.6}$$

3.3.1 FEMuS Results

This section presents the results obtained for the three simulated cases using the improved turbulence model described in Chapter 1. The governing equation used in these simulations are the RANS equations with the introduction of the Oberbeck-Boussinesq approximation for modeling the buoyancy forces. In particular, the system of equations solved by FEMuS code is as follows

$$\frac{\partial \langle u_i \rangle}{\partial x_i} = 0\,, \tag{3.7}$$

$$\frac{D\langle u_i \rangle}{Dt} = -\frac{1}{\rho}\frac{\partial \langle p \rangle}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial \langle u_j \rangle}{\partial x_i}\right) - \langle u_i' u_j' \rangle\right] - g_i \beta \langle T \rangle\,, \tag{3.8}$$

$$\frac{D\langle T \rangle}{Dt} = \frac{\partial}{\partial x_i}\left(\alpha \frac{\partial \langle T \rangle}{\partial x_i} - \langle u_i' T' \rangle\right)\,. \tag{3.9}$$

The model used to compute the Reynolds stress tensor and the turbulent heat flux are the EASM and EAHFM presented in Chapter 1. These models use the expressions for $\langle \mathbf{u}'\mathbf{u}' \rangle$ and $\langle \mathbf{u}'T' \rangle$, as detailed in (1.213) and (1.237). The turbulence model includes the logarithmic four-parameter turbulence model, $K - \Omega - K_\theta - \Omega_\theta$, as shown in (1.246), (1.247), (1.249) and (1.250). The boundary conditions for the dynamic turbulent variables over all the four

walls of the cavity are

$$\begin{aligned} \frac{\partial K}{\partial x} &= \frac{2}{\delta}, \\ \Omega &= \ln\left(\frac{2\nu}{C_\mu \delta^2}\right), \end{aligned} \tag{3.10}$$

where δ is the wall distance. On the other hand, the thermal turbulent variables have the following boundary conditions on the vertical walls, where the temperature field is fixed

$$\begin{aligned} \frac{\partial K_\theta}{\partial x} &= \frac{2}{\delta}, \\ \Omega_\theta &= \ln\left(\frac{2\alpha}{C_\mu \delta^2}\right), \end{aligned} \tag{3.11}$$

while homogeneous boundary conditions are imposed over the adiabatic walls.

The results are compared with available literature data and the previous version of the turbulence model. This older model uses the formulations for the Reynolds stress tensor and the turbulent heat flux described in [145]. The results of the improved model are referred to as $New\ A4P$ and the ones of the older version of the model are indicated as $A4P$.

Simulation 1

Simulation 1 refers to the case reported in [143] which provides DNS results for $Pr = 0.0210$ and $Ra = 3.78 \times 10^6$. The benchmark profiles of the variables are provided along the non-dimensional coordinate $x^+ = x/L$ at two different $y^+ = y/L$ positions: at $y^+ = 0.5$, corresponding to the middle of the cavity, and at $y^+ = 0.75$, corresponding to the upper part of the cavity.

Figure 3.9 shows the dimensionless magnitude of the velocity field of Simulation 1 at both y^+ positions. The profile on the left illustrates the velocity at $y^+ = 0.5$, while the plot on the right displays the results at $y^+ = 0.75$. The non-dimensional velocity is reported, and the $A4P$ results are compared with the $New\ A4P$ results. As observed, the new model significantly improves the prediction of the velocity field. The enhancements that the implemented model provides are evident, particularly in the peak values of the velocity magnitude, which are otherwise underestimated in the $A4P$. This behavior can be attributed to the earlier model's overestimation of the components of the Reynolds stress tensor.

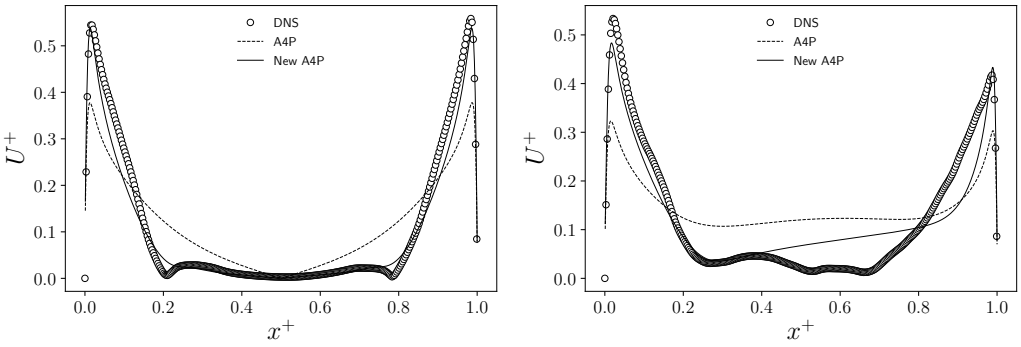

Figure 3.9: Comparison between DNS (circle markers), *A4P* and *New A4P* simulations of the non-dimensional magnitude of velocity field along x^+ coordinates at different heights: $y^+ = 0.5$ on the left, $y^+ = 0.75$ on the right.

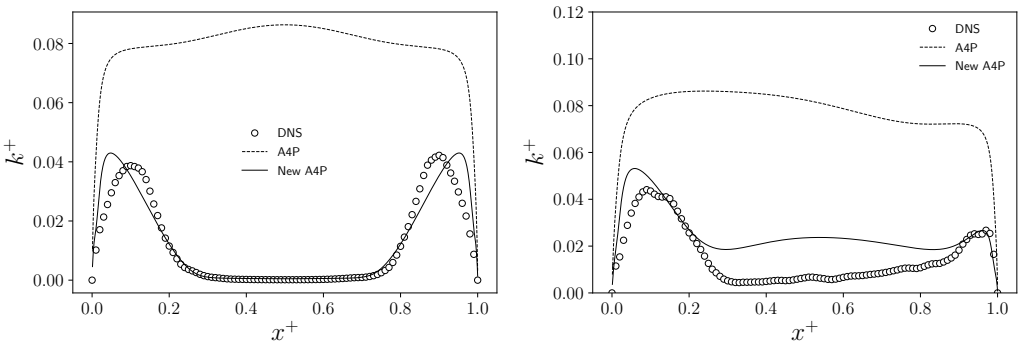

Figure 3.10: Comparison between DNS (circle markers), *A4P* and *New A4P* simulations of the non-dimensional turbulent kinetic energy field along x^+ coordinates at different heights: $y^+ = 0.5$ on the left, $y^+ = 0.75$ on the right.

The aforementioned overestimation is reflected in the values of the turbulent kinetic energy, as shown in Figure 3.10. The left image in the figure shows the plot of the non-dimensional k along x^+ at the center of the cavity, while the right image displays the plot at a height of $y^+ = 0.75$. As observed from the k profiles, the old version tends to overestimate these values at both y^+ locations, particularly in the central region of the cavity. The prediction of turbulent kinetic energy profiles appears to be more accurate when the buoyancy contribution is included in the Reynolds stress tensor and the turbulent heat flux models. However, the *New A4P* FEMuS turbulence model predicts a peak in turbulent kinetic energy that is shifted closer to the vertical walls. This inaccuracy suggests that the modeling of turbulent kinetic

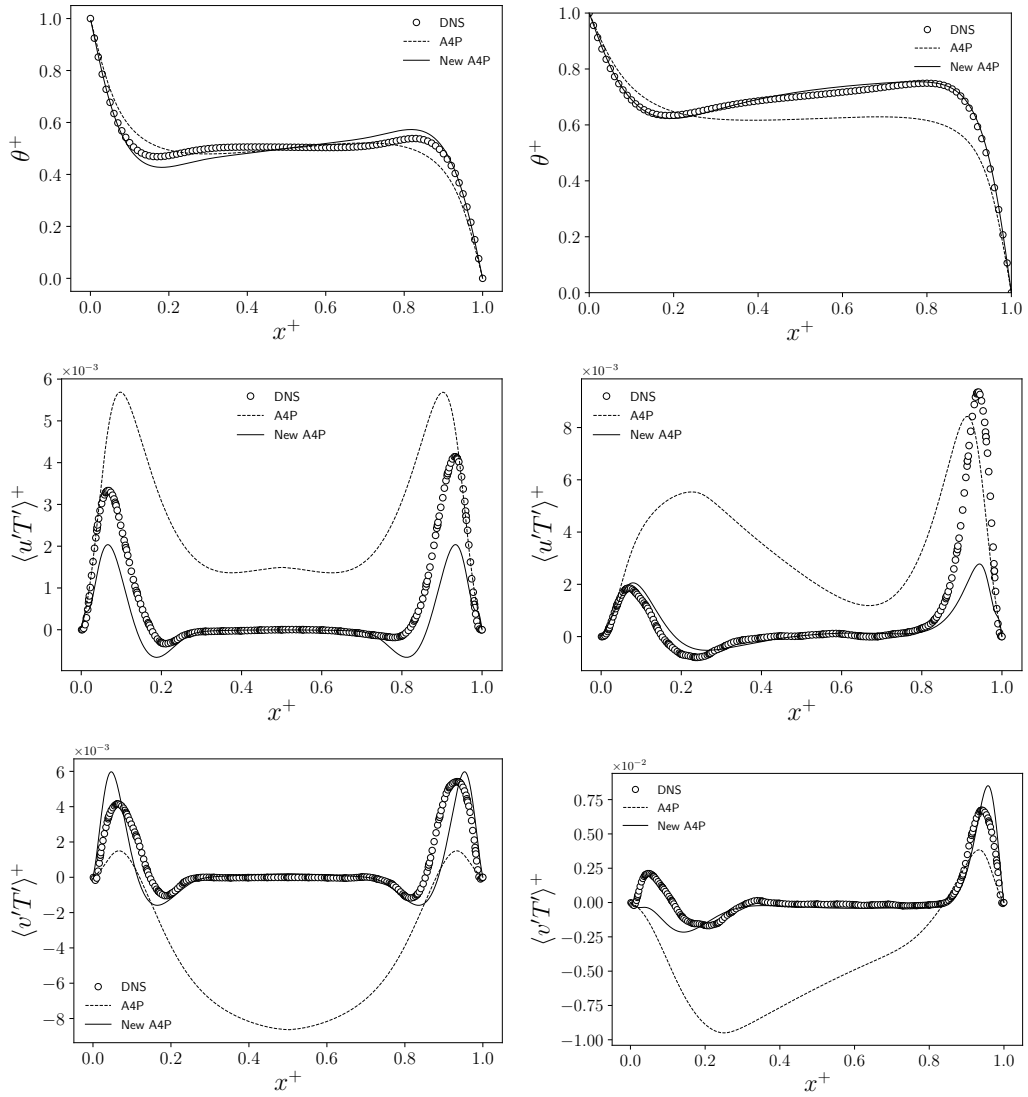

Figure 3.11: Comparison between DNS (circle markers), *A4P*, and *New A4P* simulations of the non-dimensional temperature (top) and turbulent heat flux components (center and bottom) along x^+ coordinates at different heights: $y^+ = 0.5$ on the left, $y^+ = 0.75$ on the right.

energy and its dissipation may require further corrections.

The thermal fields are shown in Figure 3.11, with the left group displaying the values at the middle of the cavity and the right group presenting the plots at three-quarters of the cavity height. The results indicate that the new version of the model provides better predictions for both the temperature field

and the turbulent heat flux components. The temperature field computed by *New A4P* shows evident improvements in predicting the profile in the upper part of the cavity. The additional term in Equations (1.213) and (1.237), introduced to account for buoyancy effects, significantly improves the accuracy of the profile predictions, particularly in capturing the shape and trend of the profiles.

Simulation 2 and 3

In this section, the results obtained for Simulations 2 and 3 are presented. The comparison includes the *New A4P*, the *A4P*, the data obtained by Mohamad and Viskanta [144] using a three-dimensional low Reynolds number $k-\varepsilon$ model, as well as experimental data from Viskanta et al. [140] for Simulation 2 and from Wolff et al. [139] for Simulation 3. Figure 3.12 shows

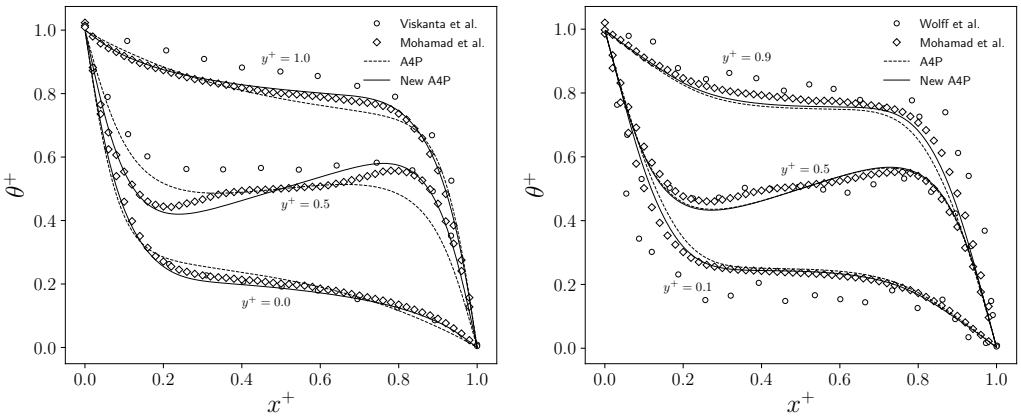

Figure 3.12: Non-dimensional temperature profiles at different cavity heights: comparison between *A4P* and *New A4P* results and reference data from [140] (circle) and [144] (diamond) for Simulation 2 (left) and from [139] (circle) and [141] (diamond) for Simulation 3 (right).

the results for Simulation 2 on the left and for Simulation 3 on the right. The non-dimensional temperature profiles are reported along x^+ coordinates for different cavity heights. According to the available reference data, Simulation 2 shows the temperature profiles at the bottom and top wall, $y^+ = 0.0$ and $y^+ = 1.0$, respectively, and the plot at $y^+ = 0.5$, corresponding to the middle plane of the cavity. On the contrary, Simulation 3 provides results for values of the non-dimensional $y-$coordinate equal to 0.1, 0.5 and 0.9. As observed, in both simulations, the results from both models show better alignment

with the simulation data by Mohamad and Viskanta [144] compared to the experimental data. In Simulation 2, the improvements introduced by the new model are particularly evident along the middle line of the cavity when compared to the reference simulation [144]. The profiles obtained with *New A4P* are more accurate than those of *A4P* also at the top and bottom walls, although the differences are relatively small. For Simulation 3, the differences between the two models are more pronounced in the upper and lower parts of the cavity, while in the middle region, the results from both models appear to be very similar.

3.3.2 OpenFOAM Results

This section provides a comparison between the simulations obtained using the available built-in turbulence models of OpenFOAM. The governing equations solved by OpenFOAM are as follows

$$\frac{\partial \langle u_i \rangle}{\partial x_i} = 0\,, \tag{3.12}$$

$$\frac{D\langle u_i \rangle}{Dt} = -\frac{1}{\rho}\frac{\partial \langle p \rangle}{\partial x_i} + \frac{\partial}{\partial x_j}\left[(\nu + \nu_t)\left(\frac{\partial \langle u_i \rangle}{\partial x_j} + \frac{\partial \langle u_j \rangle}{\partial x_i}\right)\right] - g_i \beta \langle T \rangle\,, \tag{3.13}$$

$$\frac{D\langle T \rangle}{Dt} = \frac{\partial}{\partial x_i}\left[(\alpha + \alpha_t)\frac{\partial \langle T \rangle}{\partial x_i}\right]\,. \tag{3.14}$$

where ν_t is the turbulent viscosity and α_t is the turbulent diffusivity. As can be seen, the Reynolds stress tensor and the turbulent heat flux in OpenFOAM are modeled using the Boussinesq approximation, which assumes a linear relationship between the Reynolds stresses and the mean velocity gradients, and between the turbulent heat flux and the temperature gradient. Moreover, OpenFOAM code adopts the Reynolds analogy to determine the turbulent diffusivity as

$$\alpha_t = \frac{\nu_t}{Pr_t}\,. \tag{3.15}$$

where turbulent Prandtl number is taken as constant. This approximation relates momentum transfer to heat transfer, enabling the modeling of the turbulent heat flux based on the velocity field.

The simulations are conducted by comparing four different built-in turbulence models available in OpenFOAM. Two of these models are based on the turbulent kinetic energy and its dissipation rate, namely `kEpsilon` and `RNGkEpsilon`. The `kEpsilon` model is a standard turbulence model widely

used for general-purpose simulations based on the standard Jones and Launder formulation [22], while `RNGkEpsilon` is derived from the Renormalization Group theory by Yakhot and Orszag [26], enhancing its accuracy for flows with rapid strain and swirl effects. Both models have been presented in the first Chapter in 1.3.3. The other two models are based on the turbulent kinetic energy and its specific dissipation rate, referred to as `kOmega` and `kOmegaSST`. The `kOmega` model is well-suited for flows near walls and provides accurate results in boundary layers, and is based on the Wilcox formulation [28] presented in Section 1.3.3. The `kOmegaSST` model combines the strengths of `kOmega` in near-wall regions and `kEpsilon` in free-stream regions, making it effective across a wide range of flow conditions. This model is an extension of the Wilcox k-ω formulation, incorporating the blending approach introduced by Menter [33] in his Shear Stress Transport (SST) model. The detailed model is presented in 1.3.3.

For turbulent variables, the boundary conditions on the vertical walls of the cavity are the built-in boundary conditions implemented in OpenFOAM. In particular, for the k variable, `kqRWallFunction` has been chosen, while `omegaWallFunction` and `epsilonWallFunction` have been used for ω and ε, respectively. On the other two walls, a `zeroGradient` boundary condition has been applied. The other parameters of the OpenFOAM simulation are discussed in Appendix B.

Simulation 1

The results obtained using the OpenFOAM code for the Simulation 1 case are presented in the following. Figure 3.13 illustrates the non-dimensional velocity magnitude at the top, the non-dimensional temperature field in the center, and the turbulent kinetic energy at the bottom, for both y^+ values: 0.5 on the left and 0.75 on the right. As observed, all four simulations show good agreement with DNS in predicting the velocity magnitude. However, the $k-\omega$-based model slightly overestimates the velocity peak, while the $k-\varepsilon$-based models underestimate it. The temperature profiles present similar trends for all the models. In particular, along the line in the middle of the cavity, the $k-\omega$ models provide better results compared to the $k-\varepsilon$ models. At $y^+ = 0.75$, the temperature profiles show good agreement compared to DNS data. As observed for the FEMuS simulation results, none of the turbulent models employed accurately captures the behavior of the turbulent kinetic energy. All models overestimate the values in the central part of the cavity

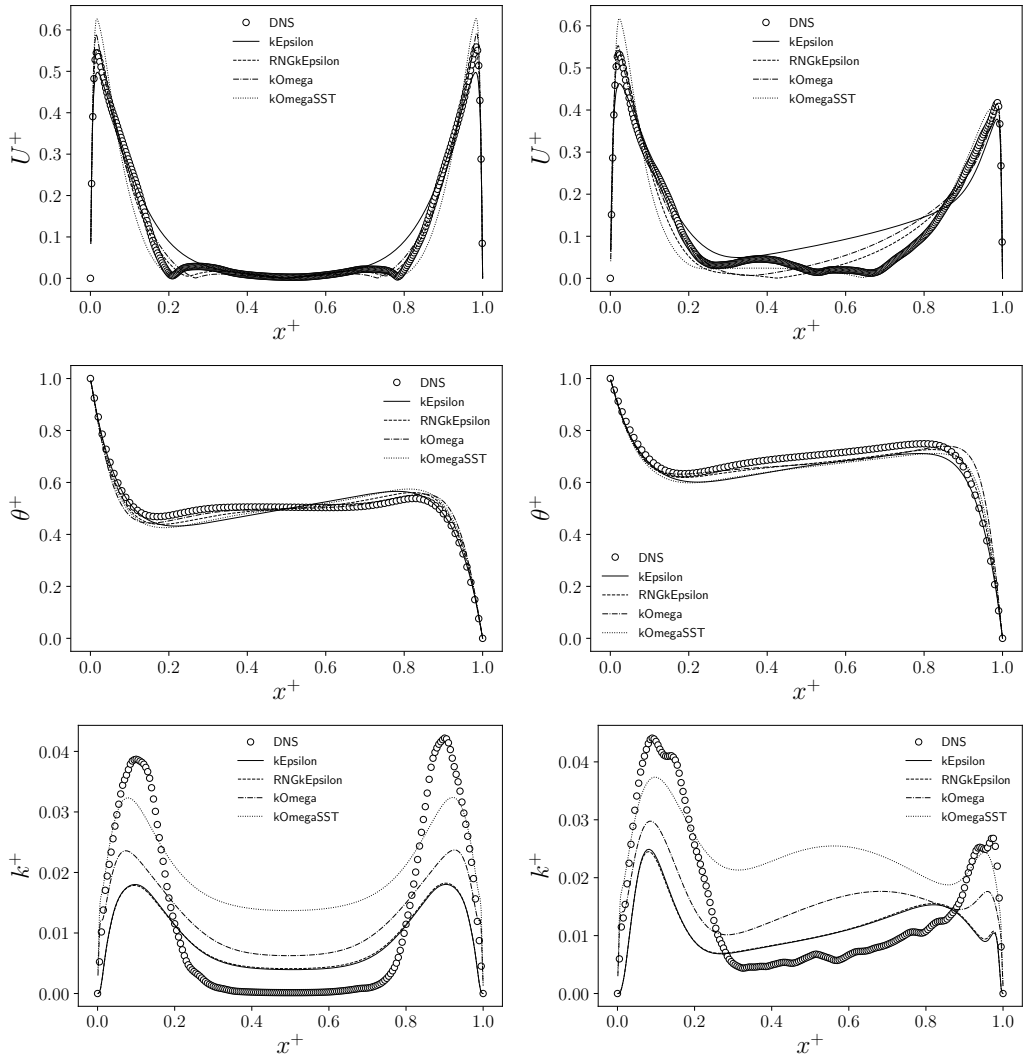

Figure 3.13: Comparison between DNS (circle markers) and the four OpenFOAM simulations of the non-dimensional velocity magnitude (top), temperature (center), and turbulent kinetic energy (bottom) along x^+ coordinates at different heights: $y^+ = 0.5$ on the left, $y^+ = 0.75$ on the right.

for both cavity heights and tend to underestimate the peak values. However, unlike FEMuS, it offers more accurate predictions of the peak's distance from the wall.

Simulation 2 and 3

The results obtained for both Simulation 2 and Simulation 3 are presented in this section. Based on the outcomes of Simulation 1, the `kEpsilon` and `kOmegaSST` models were excluded from further analysis, as they provided less accurate predictions of the main fields. Consequently, Simulations 2 and 3 were carried out using the `RNGkEpsilon` and `kOmega` models only, and the results were compared with the reference data. In Figure 3.14, the non-dimensional temperature profiles are shown for Simulation 2 (left) and Simulation 3 (right). Overall, both models align more closely with the reference

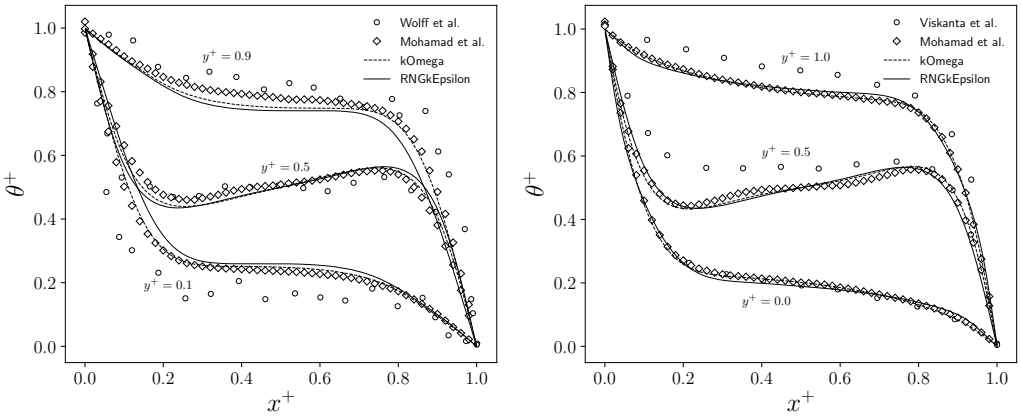

Figure 3.14: Non-dimensional temperature profiles at different cavity heights: comparison between `RNGkEpsilon` and `kOmega` results and reference data from [140] (circle) and [144] (diamond) for Simulation 2 (left) and from [139] (circle) and [141] (diamond) for Simulation 3 (right).

data from [144] than with the experimental data, similar to the observations made with the FEMuS results.

3.3.3 Coupling Application Results

Both FEMuS and OpenFOAM codes show several weaknesses in predicting the variable profiles. FEMuS performs better in simulating the thermal fields, whereas OpenFOAM demonstrates good agreement in simulating the velocity field. These deficiencies in both codes may arise from the imperfect prediction of the strongly coupled fields. Thus, many questions may arise. What would the resulting temperature field be like in FEMuS using the more accurate velocity field provided by OpenFOAM? Could the velocity profile predicted

by OpenFOAM improve by using the more accurate thermal turbulent model implemented in FEMuS? Exploring these possibilities could help address the observed discrepancies and enhance the overall predictive accuracy of the simulations.

For this reason, the third simulation performed for the first case ($Pr = 0.021$ and $Ra = 3.78 \times 10^6$) [143] employs the volume data transfer algorithm described in Chapter 2. Figure 3.15 reports the coupling scheme implemented for this application. FEMuS is used to solve the thermal fields, while OpenFOAM is employed to solve the dynamic fields. In particular, the FEMuS code solves for T, K_θ, Ω_θ, and the turbulent heat flux components, whereas OpenFOAM handles the velocity field, the turbulent kinetic energy, and its specific dissipation using the $k-\omega$ model. The coupling application is used

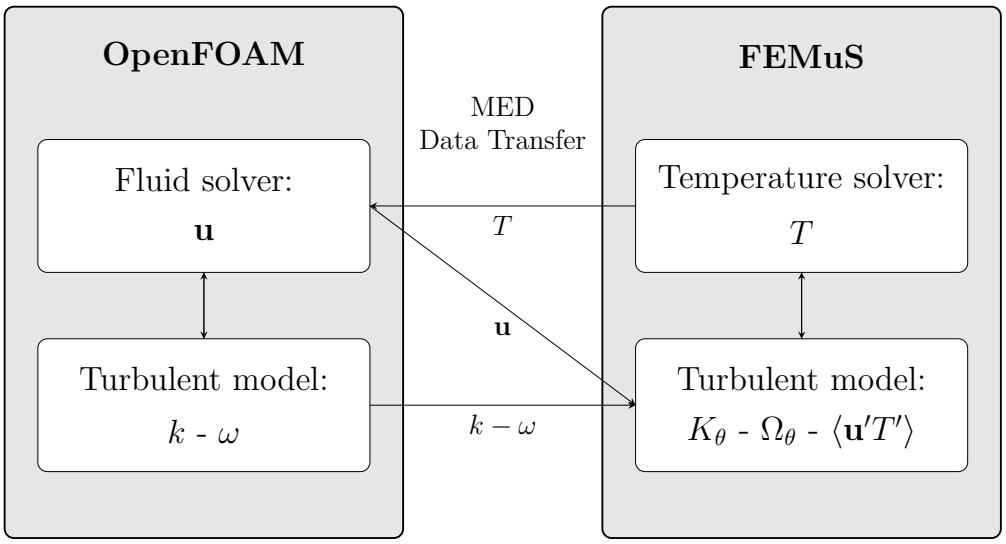

Figure 3.15: Schematic representation of the coupling algorithm for the volume data transfer application in the turbulent cavity case.

to exchange the fields between FEMuS and OpenFOAM following Algorithm 1. FEMuS provides the temperature field to OpenFOAM, which uses it to compute the buoyancy term in the Navier-Stokes equation. In turn, OpenFOAM supplies the velocity field and both dynamic turbulent variables, k and ω.

In the following, the results of the coupling application, here referred to as Coupled $k-\omega$, have been compared with the DNS data and the monolithic solver solutions, the results from the *New A4P* obtained with FEMuS and the

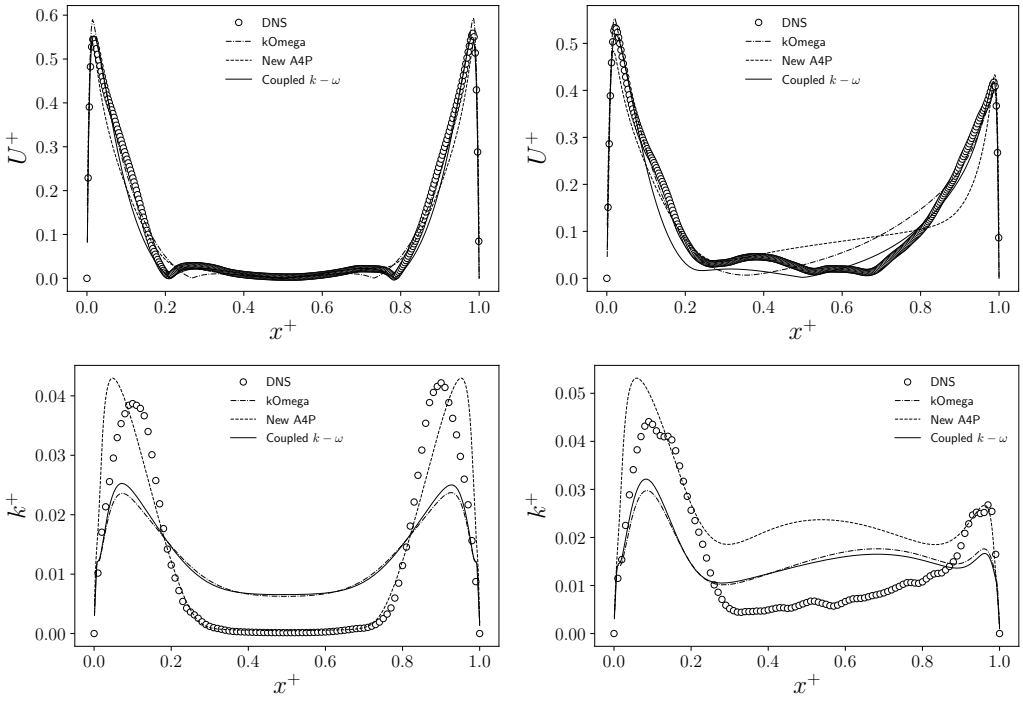

Figure 3.16: Comparison between DNS (circle marker), coupling application and monolithic simulations of the non-dimensional velocity magnitude (top) and turbulent kinetic energy (bottom) along x^+ coordinates at different heights: $y^+ = 0.5$ on the left, $y^+ = 0.75$ on the right.

$k-\omega$ model from OpenFOAM. The profiles of the variables are plotted against the non-dimensional $x-$direction at two different heights of the cavity: $y^+ = 0.5$ and $y^+ = 0.75$. To quantify the discrepancies between the simulation profiles and the DNS data, the Root Mean Square Error (RMSE) is calculated as

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\phi_i - \phi_i^*)^2}, \tag{3.16}$$

where ϕ_i represents the simulation results and ϕ_i^* corresponds to the DNS data at each point i. Additionally, the accuracy of the results are further evaluated using the Normalized RMSE (NRMSE), which is computed by dividing the RMSE by the range of the DNS data

$$NRMSE = \frac{RMSE}{\phi_{max}^* - \phi_{min}^*}, \tag{3.17}$$

where ϕ_{max}^* and ϕ_{min}^* are the maximum and minimum values of the DNS

data.

		Coupled $k-\omega$	kOmega	*New A4P*
$y^+ = 0.5$	RMSE	0.0240	0.0306	0.0327
	NRMSE	0.0435	0.0555	0.0594
$y^+ = 0.75$	RMSE	0.0296	0.0346	0.0484
	NRMSE	0.0556	0.0650	0.0908

Table 3.2: RMSE and NRMSE for dimensionless velocity magnitude, U^+.

Figure 3.16 illustrates the dimensionless velocity profile (top) and the turbulent kinetic energy (bottom). Regarding U^+, both monolithic code solutions produce results consistent with the DNS profiles. The FEMuS solution tends to underestimate the velocity peak, while OpenFOAM slightly overestimates it. The coupling application provides results in good agreement with DNS data, with improvements over monolithic OpenFOAM solver prediction. In particular, the velocity peak and the velocity at the center of the cavity, where the flow is nearly stationary, are accurately predicted. However, in the transition region between the peak and the center of the cavity, the coupled simulation still shows some deviation from the DNS profile. Table 3.2 reports the RMSE and NRMSE values for U^+. The error in the coupled simulations is lower than that of the FEMuS and OpenFOAM solutions for both y^+ values.

		Coupled $k-\omega$	kOmega	*New A4P*
$y^+ = 0.5$	RMSE	0.0085	0.0104	0.0056
	NRMSE	0.201	0.246	0.134
$y^+ = 0.75$	RMSE	0.0083	0.0103	0.0121
	NRMSE	0.189	0.232	0.275

Table 3.3: RMSE and NRMSE for dimensionless turbulent kinetic energy, k^+.

Regarding the turbulent kinetic energy, as illustrated in Figure 3.16, FEMuS and OpenFOAM produce appreciably different results. In particular, FEMuS provides a k profile that better matches the DNS data, especially in the central part of the cavity, while OpenFOAM offers a more accurate prediction of the peak location. In the coupled application, the model used for

k is based on OpenFOAM, and thus, the solution closely follows the OpenFOAM results for both y^+ values. The peak of k is encouragingly maintained in the correct wall distance compared to the FEMuS results, providing a better prediction of its position. Moreover, the value of the peak increases from the OpenFOAM monolithic solution. Table 3.3 reports the overall errors. As expected, the slightest deviation from the DNS data is observed for the FEMuS solution. However, the k values predicted by OpenFOAM show an improvement when using the same turbulent model in the coupling application.

		Coupled $k-\omega$	kOmega	*New A4P*
$y^+ = 0.5$	RMSE	0.0214	0.0254	0.0275
$y^+ = 0.75$	RMSE	0.0116	0.0389	0.0119

Table 3.4: RMSE for dimensionless temperature, θ^+.

In Figure 3.17, the profiles of the simulated thermal field are displayed. The RMSE values for the temperature solutions are provided in Table 3.4, while RMSE and NRMSE for the turbulent heat flux components are reported in Table 3.5. As shown in Figure 3.17 (top) and Table 3.4, the temperature profile is well-estimated by the coupled application. The prediction of the non-dimensional temperature has been improved compared to both monolithic solutions for both y^+ values.

		Coupled $k-\omega$		*New A4P*	
		$\langle u'T' \rangle$	$\langle v'T' \rangle$	$\langle u'T' \rangle$	$\langle v'T' \rangle$
$y^+ = 0.5$	RMSE	0.0010	0.0011	0.0009	0.0012
	NRMSE	0.2330	0.1683	0.1929	0.1842
$y^+ = 0.75$	RMSE	0.0019	0.0010	0.0018	0.0012
	NRMSE	0.1888	0.1210	0.1813	0.1369

Table 3.5: RMSE and NRMSE for dimensionless turbulent heat flux, $\langle \mathbf{u}'T' \rangle$.

The turbulent heat flux profiles are shown in Figure 3.17 (center and bottom), and the corresponding errors are reported in Table 3.5. At $y^+ = 0.5$, both components show a reduced range between their minimum and maximum values. In particular, the FEMuS monolithic solution tends to overestimate the maximum and underestimate the minimum of $\langle v'T' \rangle$, whereas the coupled simulation smooths the profile by reducing the maximum and

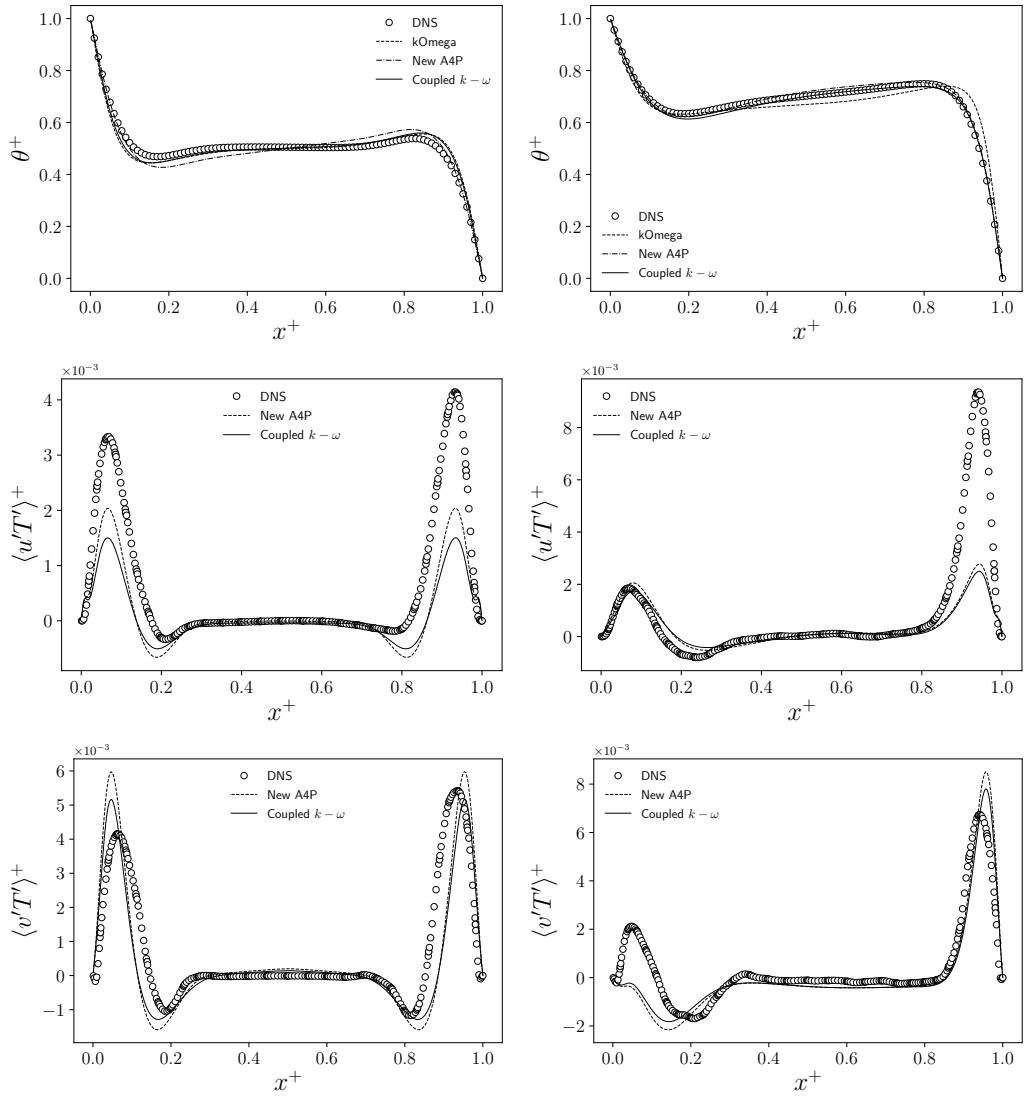

Figure 3.17: Comparison between DNS (circle markers), coupling application and monolithic simulations of the non-dimensional temperature (top) and turbulent heat flux components (center and bottom) along x^+ coordinates at different heights: $y^+ = 0.5$ on the left, $y^+ = 0.75$ on the right.

raising the minimum values. As regard the normal component of the turbulent heat flux, the peak values predicted by the coupling application deviate further from the DNS reference data. In the upper part of the cavity, the results remain largely unchanged compared to the monolithic solutions. As shown in Table 3.5, the errors in the turbulent flux components remain nearly

unchanged compared to the monolithic solutions.

We now introduce the results for the skin friction coefficient and the Nusselt number. The skin friction coefficient C_f is defined as

$$C_f = \frac{1}{2\sqrt{Gr}} \frac{\partial \langle v \rangle}{\partial x}, \tag{3.18}$$

where $\langle v \rangle$ is the vertical component of the mean velocity field. The average Nusselt number has been computed as the integral of the local Nusselt number in (2.21) over the hot wall. Table 3.6 presents the skin friction coefficient and

	Coupled $k-\omega$		kOmega		*New A4P*		[143]
	Value	Error	Value	Error	Value	Error	Value
C_f	0.098	10.90%	0.083	24.92%	0.076	30.81%	0.110
Nu	7.756	2.44%	8.440	6.17%	7.682	3.37%	7.950

Table 3.6: Skin friction coefficient and Nusselt number values and errors.

the Nusselt number for the three simulations, along with their error values. Despite the high error in predicting C_f, the coupled simulation demonstrates a considerable improvement with respect to both the monolithic solutions. Moreover, the Nusselt number error, which was already low has been further reduced to 2.44%. By combining the dynamic solver of OpenFOAM with the more accurate thermal turbulence model of FEMuS, both C_f and Nu predictions are improved.

Chapter 4

Liquid Metal Heat Exchanger

The Ph.D. project, as part of the PON Research and Innovation program, involves a collaborative initiative with Nier Ingegneria S.p.A., a technical consultancy company that offers various services in the field of integral sustainability. The company has expertise in several areas, including Systems Engineering, Sustainability, Occupational Safety, and Software Engineering. The collaboration was carried out with the System Engineering Area team, which provides specialized technical consulting services in various fields, such as fusion energy.

The collaboration takes place in the context of urgent climate change, a challenge that demands a significant transition towards the use of sustainable energy sources. Among these, solar energy stands out as one of the most important renewable resources. Over the years, advances in solar energy technology have boosted the development of increasingly efficient systems to generate heat and electricity. One notable advancement has been inspired by projects such as NEXTower [9], which has led to the use of liquid metals as heat transfer fluids in Concentrated Solar Power systems operating at high temperatures [10, 11]. This development draws on the experience gained in the design of IV-generation nuclear reactors and nuclear fusion reactors. The synergy between nuclear reactor design and CSP technology fields provides a robust foundation for creating high-efficiency energy systems that operate with liquid metals. The use of liquid metals, including lead and its alloys, as

heat transfer fluids and thermal storage fluids could enable energy systems to increase their operating temperature and, thus, plant efficiency. Therefore, it is essential to accurately predict the fluid dynamics and thermal behavior of these fluids, particularly under turbulent flow regimes.

This chapter presents the collaboration project with Nier Ingegneria S.p.A. and focuses on analyzing various approaches for the numerical simulation of a realistic PbLi-air heat exchanger designed by the company. The study aims to improve the accuracy of predictions beyond the zero-dimensional approach used by the company in heat exchanger design. The study begins with a description of the heat exchanger, including its design constraints imposed by the prototype requirements. The analysis proceeds with simulations of the turbulent flow of the lead-lithium alloy (PbLi) using different numerical methods. These approaches are compared to the original design parameters and validated by comparing them with Direct DNS results to assess their accuracy and reliability. In addition, the chapter explores a numerical simulation of the tube-and-fin assembly of the heat exchanger, where the interaction between the fins and the air is modeled as a porous medium. Finally, the study addresses the conjugate heat transfer problem by building a comprehensive simulation framework. This configuration involves coupling fluid and solid domains using the boundary data transfer algorithm as described in Chapter 2.

4.1 Description of the Heat Exchanger

The analyzed PbLi-air heat exchanger is part of the International Thermonuclear Experimental Reactor (ITER) Test Blanket Modules (TBMs), which are designed to evaluate tritium breeding concepts and heat extraction technologies for fusion power plants. The results from the TBM program will directly contribute to the design of DEMO, the demonstration fusion power plant that will succeed ITER [146, 147, 148]. The TBMs in ITER require several ancillary systems to ensure their proper operation, monitoring, and integration within the reactor. These supporting systems handle cooling, tritium production and extraction, neutron diagnostics, and safety functions [149]. Among them, the Lead-Lithium loop is a crucial system in some TBMs and serves as both a breeder material to generate tritium (T) via neutron interactions with lithium and a coolant to extract heat from the TBM. In the normal operational state of the PbLi loop, the liquid metal flow returns from

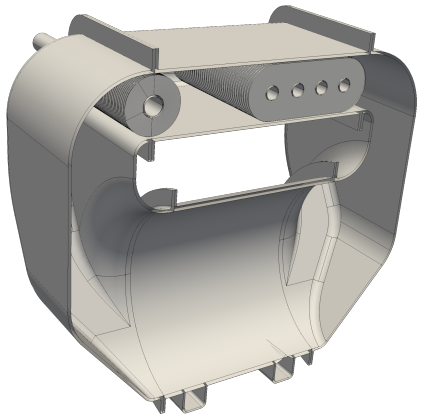

Figure 4.1: Geometry of the PbLi loop cooler.

the TBM module at a temperature of around 673 K, with a mass flow rate in the experimental range of $0.2 - 1.0$ kg/s. The PbLi alloy is then heated to 723 K and enters the TEU (Tritium Extraction Unit), which extracts tritium from the liquid metal into a gas phase. Downwards, a heat exchanger cools down the PbLi to 573 K, this is done through a double cooling system consisting of a closed air circuit (primary coolant) and a water cooling connected to the ITER CCWS (Component Cooling Water System). After that, part of the PbLi flow enters the cold trap (CT) for alloy purification, while the remaining part is directly sent to the storage tank which also acts as a draining system.

The PbLi loop heat exchanger model is shown in Figure 4.1. The system consists of a closed outer casing that directs airflow from the lower chamber, where the fan is located, to the upper section, which houses the tube bundle that forms the heat exchanger. The airflow first passes through the tube bundle on the left, where it extracts heat from the PbLi, cooling the liquid metal. It then flows through a cooling coil, where it is further cooled by the external water circuit.

This cooling system must ensure that the PbLi flow meets the required conditions at the TBM inlet while maintaining a sufficient margin above the PbLi freezing temperature throughout the entire loop. Regarding the lower limit (safety margin), PbLi outlet temperatures below the design constraints can be considered acceptable. In fact, the coldest region in the loop is not the outlet of the heat exchanger but rather the CT outlet pipe, where PbLi

reaches 526 K, which is 16 K above the freezing point [150]. Moreover, to prevent the temperature from dropping further, heating cables will be activated to maintain a safe operating margin. For these reasons, the analysis of the heat exchanger performance serves to verify that the upper limit (TBM inlet temperature) is satisfied, while lower temperatures can be considered acceptable as long as they remain above the solidification temperature.

In the following, the heat exchanger's performance analysis focuses only on its critical section, which is the interaction between the PbLi flow pipe and the air. This section of the heat exchanger involves a non-conventional fluid, the lead-lithium alloy, whose thermal performance requires particular attention for the sizing of the system in Figure 4.1.

4.1.1 Constraints and properties of the PbLi-air heat exchanger

According to the constraints, the cooler system of the PbLi loop is designed to reduce the temperature of the PbLi alloy flow from 723 K to 573 K, achieving a temperature reduction of 150 K. The mass flow rate of the liquid metal is set at 0.63 kg/s, corresponding to the thermal power extraction of 17870 W. This cooling is accomplished using an airflow that externally sweeps the pipe with a mass flow rate of 1.517 kg/s. The air temperature is set to 333 K, and it absorbs the heat from the liquid metal as it flows through the tube. As seen in Figure 4.1, the PbLi-air heat exchanger consists of a single pipe with a hydraulic diameter of $D_h = 0.03$ m and a length of $L = 0.6$ m. The pipe is made of EUROFER (EUROpean FERritic-martensitic steel) [151] with a thickness of 0.004 m and is covered by 120 copper fins, each with an average length of 0.014 m. The schematic of the PbLi-air heat exchanger is shown in Figure 4.2, and the design parameters are summarized in Table 4.1 on the left. The figure shows the internal tube where the liquid metal flows from the inlet (on the left) to the outlet (on the right), the EUROFER pipe confining the liquid metal flow, and the copper fins on the outside. The airflow is in a cross-current configuration against the tube.

Table 4.1 on the right reports the physical properties of the materials used in the following simulations. In particular, the properties of the liquid metal refer to the Pb-rich eutectic alloy Pb-16Li (16 at.% Li) at the operational temperature of 648 K [150]. EUROFER is a low-carbon ferritic-martensitic steel optimized for resistance to neutron irradiation damage [151]. Its thermophys-

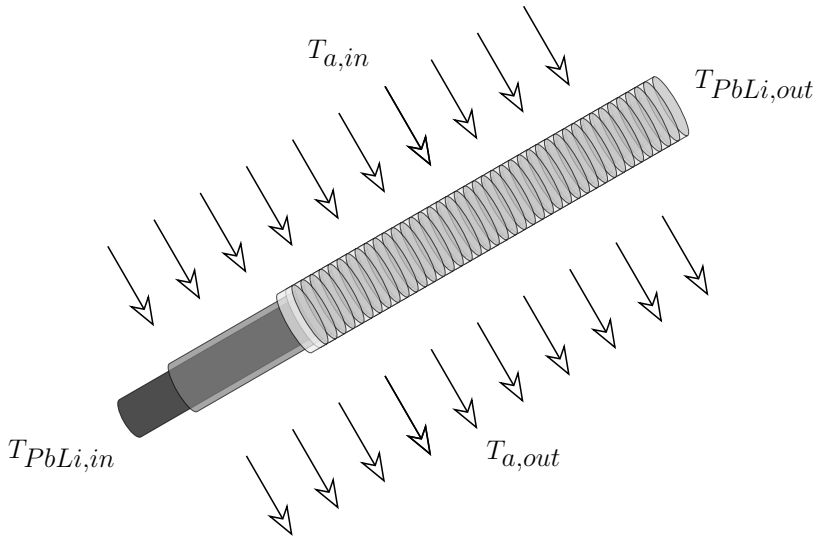

Figure 4.2: Schematic representation of the PbLi-air heat exchanger.

Design Parameter		
$T_{PbLi,in}$	723	K
$T_{PbLi,out}$	573	K
$\dot{m}_{PbLi}$	0.63	kg/s
$T_{air,in}$	333	K
$\dot{m}_{air}$	1.517	kg/s

Properties				
Material	ρ $[\frac{\text{kg}}{\text{m}^3}]$	λ $[\frac{\text{W}}{\text{mK}}]$	c $[\frac{\text{J}}{\text{kgK}}]$	μ [Pa s]
PbLi	9749	21.9	189.1	0.0017
Air	1.225	0.0292	1008.2	0.00002
EUROFER	7798	28.8	512.70	-
Copper	8933	401	385	-

Table 4.1: Design constraints of the heat exchanger and physical properties of the involved material.

ical properties are reported at temperatures corresponding to the operating temperature of the liquid metal.

4.1.2 Zero-dimensional analysis

According to the company's design, the PbLi-air heat exchanger has been modeled using an electrothermal analogy, where thermal resistances correspond to electrical resistor components. Figure 4.3 illustrates the scheme of thermal resistance acting on the heat exchanger system. The model comprises a series of two resistances and a block of resistance in parallel. The first resistance is associated with the thermal convection of the liquid metal

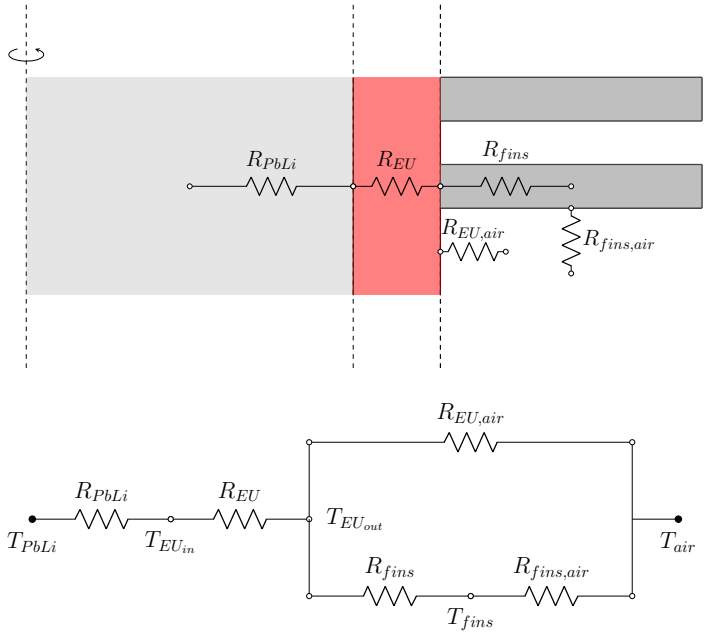

Figure 4.3: Schematic representation of the thermal resistance analogy.

to the pipe wall, labeled by R_{PbLi}. The thermal resistance for convection heat transfer is expressed as

$$R_{PbLi} = \frac{1}{hA}, \tag{4.1}$$

where h is the heat transfer coefficient at the pipe wall, and A is the exchange surface, which corresponds to the internal surface area of the pipe. The liquid metal heat transfer coefficient has been computed as $h = \lambda Nu/D_h$ using the following experimental correlation for the Nusselt number [152]

$$Nu = 6.3 + 0.0167(RePr)^{0.89} Pr^{0.08}. \tag{4.2}$$

The value is equal to $5704\,\mathrm{W/(m^2K)}$, calculated for a Reynolds number of 1.57×10^4 and a Prandtl number of 0.0146. Consequently, the calculated resistance is $0.0031\,\mathrm{K/W}$. The conduction resistance introduced by the EUROFER pipe is determined using the solution of Fourier's equation for a hollow cylinder. The thermal resistance is given by the following relation

$$R_{EU} = \frac{\ln(r_e/r_i)}{2\pi\lambda L}, \tag{4.3}$$

where r_e and r_i are the external and internal radii of the pipe, λ is the thermal conductivity of the material, and L is the pipe length. Given the thickness of

the pipe, 0.004 m, the length of the pipe, 0.6 m, and the conductivity of the EUROFER as in Table 4.1 the value of R_{EU} is equal to 0.0022 K/W. The block of parallel resistances consists of three distinct resistances. $R_{EU,air}$, is associated with convection between air and the EUROFER pipe and it assumes the value of 0.123 K/W. It can be calculated using the same formula as in (4.1), considering a value of 140.4 W/m^{2}K for the heat transfer coefficient and a value of 0.058 m^2 for the surface area of the pipe in contact with the air. The heat transfer coefficient of the air sweeping the pipe has been calculated as $h = \lambda Nu/D_h$ using an empirical correlation for the Nusselt number, known as Gnielinski correlation [153]. Thus, the Nusselt number can be computed as

$$Nu = \frac{(f/2)(Re - 1000)Pr}{1 + 12.7(f/2)^{1/2}(Pr^{2/3} - 1)}, \tag{4.4}$$

where the Reynolds number for the air flow is 6.28×10^4 and f is the Darcy friction factor. This parameter has been calculated using the following experimental relationship

$$f = A + BRe^{-1/m}, \quad \text{with} \quad A = 0.00128,\ B = 0.1143,\ m = 3.2154. \tag{4.5}$$

$R_{EU,air}$ is in parallel with two other resistances connected in series. The first is R_{fins}, which corresponds to conduction within the fins. It has been calculated with experimental correlation and assumes the value of 0.0022 K/W. The second resistance, $R_{fins,air}$, corresponds to convection between air and the surface of the fins. The external surface of the fins in contact with airflow is 0.630 m^2 and the heat transfer coefficient is 140.4 W/m^{2}K. As a result, the value of $R_{fins,air}$ is equal to 0.011 K/W.

The total thermal resistance calculated by the company's design process is expressed as follows

$$R_{tot} = \frac{1}{\frac{1}{R_{fins,air}+R_{fins}} + \frac{1}{R_{EU,air}}} + R_{EU} + R_{PbLi} = 0.0173\,\text{W/K}, \tag{4.6}$$

and the total thermal power extracted is

$$\dot{Q} = \frac{\Delta T}{R_{tot}} \simeq 17873\,\text{W}, \tag{4.7}$$

where ΔT is the temperature difference between the mean temperature of the liquid metal, T_{PbLi}, and the mean temperature of the air, T_{air}. These

mean temperatures are calculated as the averages of the inlet and outlet temperatures of liquid metal and air.

The use of experimental relationships and analytical expressions results in a total extracted thermal power that exactly matches the design constraint. This outcome leaves no margin for uncertainty or safety factors, resulting in non-conservative predictions. For these reasons, a CFD analysis has been requested to validate or revise the current heat exchanger design. Compared to the zero-dimensional approach, CFD simulations can offer more accurate insights, particularly in estimating the heat transfer coefficient of the liquid metal. By employing an appropriate numerical model for turbulent liquid metal flow, it is possible to achieve a more precise prediction of the heat transfer behavior than the algebraic value calculated in this section.

4.2 Lead-Lithium Simulation

The simulation of the lead-lithium alloy has been performed using both FEMuS and OpenFOAM monolithic turbulent codes. Considering the sym-

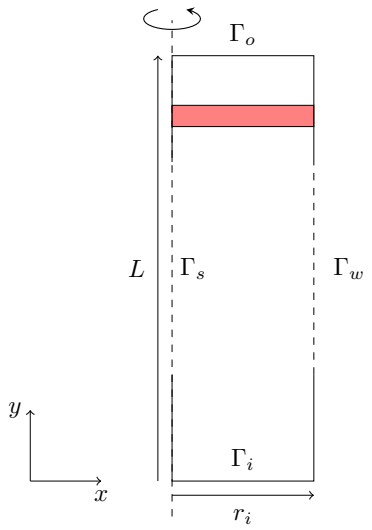

Figure 4.4: Schematic representation of the computational domain for Lead-Lithium flow simulation.

metry of the problem, the computational domain can be simplified. The Lead-Lithium simulation was performed over the entire length of the pipe

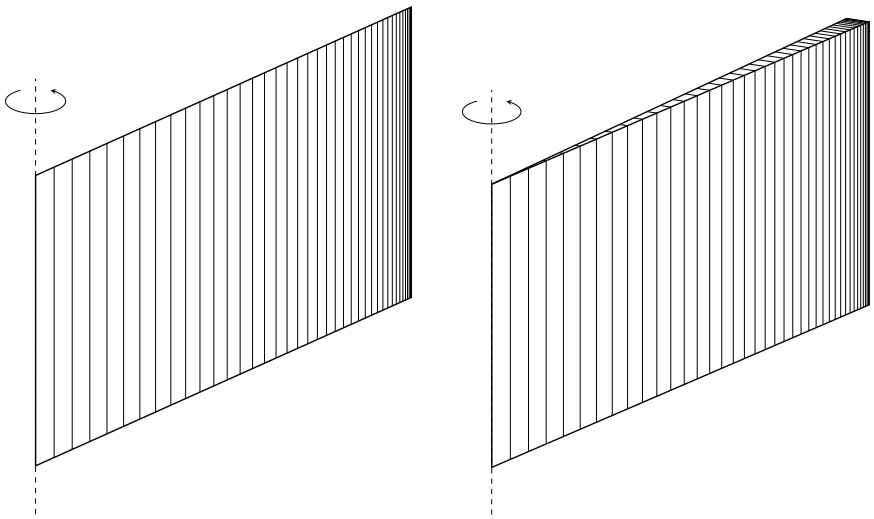

Figure 4.5: Refined mesh of a single cell along the length of the pipe, used for the liquid metal flow simulation by FEMuS (left) and OpenFOAM (right).

but employed an axisymmetric configuration in FEMuS code and a wedge configuration in OpenFOAM. Both strategies effectively represent the radial symmetry of the simulation. The schematic representation of the geometry has been reported in Figure 4.4. The simplification results in a 2D domain for FEMuS defined as a rectangle with a base of $r_i = 0.015\,\mathrm{m}$ and a length of $L = 0.6\,\mathrm{m}$. Similarly, OpenFOAM adopts a wedge geometry with a radius of $0.015\,\mathrm{m}$ and a length of $0.6\,\mathrm{m}$. Figure 4.5 shows a single cell in y direction to provide a clearer view of the radial mesh. It presents a cross-sectional slice of the pipe, also highlighted in red in Figure 4.4. A refinement near the wall has been adopted in both setups to guarantee that y^+ remains lower than 1. In particular, the wall refinement ensures that the cell closest to the wall has a width of $1 \times 10^{-4}\,\mathrm{m}$. The computational mesh consists of 14792 cells with 172 cells along the y direction and 86 along the radial direction for both simulations.

Considering the operational temperature range, the thermal variation of the physical properties is not expected to impact significantly on their values. Thus, the liquid metal flow operates in a forced regime and the buoyancy terms are neglected in all the governing equations. In particular, FEMuS code uses the following RANS system of equation

$$\frac{\partial \langle u_i \rangle}{\partial x_i} = 0\,, \tag{4.8}$$

$$\frac{D\langle u_i\rangle}{Dt} = -\frac{1}{\rho}\frac{\partial\langle p\rangle}{\partial x_i} + \frac{\partial}{\partial x_j}\left[\nu\left(\frac{\partial\langle u_i\rangle}{\partial x_j} + \frac{\partial\langle u_j\rangle}{\partial x_i}\right) - \langle u_i' u_j'\rangle\right], \tag{4.9}$$

$$\frac{D\langle T\rangle}{Dt} = \frac{\partial}{\partial x_i}\left(\alpha\frac{\partial\langle T\rangle}{\partial x_i} - \langle u_i' T'\rangle\right). \tag{4.10}$$

The Reynolds stress tensor and the turbulent heat flux are computed using EASM and EAHFM presented in Chapter 1. These models use the expressions for $\langle \mathbf{u}'\mathbf{u}'\rangle$ and $\langle \mathbf{u}'T'\rangle$, as detailed in (1.213) and (1.237). The model closure is obtained with the logarithmic four-parameter turbulence model, $K-\Omega-K_\theta-\Omega_\theta$, as shown in (1.246), (1.247), (1.249) and (1.250).

OpenFOAM solves for the RANS system of equations using the Boussinesq hypothesis

$$\frac{\partial\langle u_i\rangle}{\partial x_i} = 0\,, \tag{4.11}$$

$$\frac{D\langle u_i\rangle}{Dt} = -\frac{1}{\rho}\frac{\partial\langle p\rangle}{\partial x_i} + \frac{\partial}{\partial x_j}\left[(\nu+\nu_t)\left(\frac{\partial\langle u_i\rangle}{\partial x_j} + \frac{\partial\langle u_j\rangle}{\partial x_i}\right)\right], \tag{4.12}$$

$$\frac{D\langle T\rangle}{Dt} = \frac{\partial}{\partial x_i}\left[(\alpha+\alpha_t)\frac{\partial\langle T\rangle}{\partial x_i}\right]. \tag{4.13}$$

The turbulent viscosity is computed with the `kOmegaSST` model, while the turbulent diffusivity is derived through the Reynolds analogy with $Pr_t = 0.85$.

We base the simulation approach on the assumption that the lead-lithium alloy flows through a pipe before entering the heat exchanger and reaches the fully developed flow condition. With respect to the thermal field, the simulation is meant to model a developing flow. In a realistic scenario, the flow enters the pipe at its maximum temperature and begins to cool as it progresses along the pipe. This setup leads to the following boundary conditions: for both codes, a no-slip boundary condition is applied to the velocity field on the wall of the pipe Γ_w. To simulate fully developed flow, FEMuS adopts pressure values as boundary conditions at both inlet and outlet (Γ_i and Γ_o). These constraints create a sufficient pressure drop to achieve the correct mass flow rate required by the setup. OpenFOAM, on the other hand, employs mapped boundary conditions, where the outlet velocity is mapped to the inlet boundary. Moreover, to guarantee a fully developed flow regime, OpenFOAM imposes a constraint on the mean bulk velocity. In particular, the imposition of a `meanVelocityForce` constraint and the setting of the bulk velocity to 0.0914 m/s create the condition for the fully developed flow.

In FEMuS, the symmetry condition imposed on the boundary along the rotation axis, Γ_s, is a homogeneous Neumann boundary condition. In contrast, OpenFOAM uses a `wedge` boundary condition for the front and back faces of the wedge geometry.

For the temperature field, both codes set the inlet temperature to a uniform value of 723 K, representing the maximum temperature of the lead-lithium alloy. At the pipe wall, a homogeneous heat flux boundary condition is applied, with a value of $\dot{q} = -315998\,\mathrm{W/m^2}$. This value corresponds to the design specification provided by the company and represents the total thermal power per unit surface area. A homogeneous Neumann boundary condition is applied at the outlet for both codes, meaning that no heat escapes the simulation domains through these boundaries. Symmetry conditions are imposed on the remaining boundaries.

Regarding the turbulent variables, FEMuS applies zero-gradient boundary conditions at both the inlet and outlet for all dynamic and thermal turbulent quantities. On the wall boundary Γ_w, we impose the conditions specified in 3.10 for k and ω, and in 3.11 for k_θ and ω_θ. In contrast, OpenFOAM applies mapped boundary conditions at the inlet and outlet, and on Γ_w it uses the built-in `kqRWallFunction` and `omegaWallFunction` for k and ω, respectively. The other parameters of the OpenFOAM simulation are discussed in Appendix B.

4.2.1 Numerical Results

In this section, the results obtained with the CFD codes are presented. Considering the axisymmetry of the problem computed with FEMuS code, the bulk velocity can be calculated as

$$u_b = \frac{\int_0^{r_i} \langle u(r) \rangle r dr}{\int_0^{r_i} r dr}\,, \tag{4.14}$$

where $u(r)$ is the streamwise component of the velocity vector, the only non-zero velocity component. For the OpenFOAM code, u_b can be computed as an integral on any cross-sectional surface normal to the stream direction. The pressure drop imposed in FEMuS code and the `meanVelocityForce` constraint in OpenFOAM are set to achieve a bulk velocity of 0.0914 m/s. This velocity corresponds to a mass flow rate of 0.63 kg/s, as required by the design constraint.

With respect to the thermal field, the heat flux applied to the wall boundary is responsible for cooling the flow temperature from the inlet value to the required outlet temperature. The bulk temperature at the outlet can be calculated for FEMuS by considering the cylindrical configuration of the pipe as follows

$$T_{b,out} = \frac{\int_0^{r_i} \langle T(r) \rangle \langle u(r) \rangle r dr}{\int_0^{r_i} \langle u(r) \rangle r dr} . \tag{4.15}$$

For OpenFOAM, the outlet bulk temperature can be calculated by applying an integral on the outlet surface. Table 4.2 compares the simulation results with the predicted analysis provided by the company. The temperature at the pipe outlet, calculated using the FEMuS solver, is 573.1 K, while the temperature obtained from the OpenFOAM solution is 572.6 K. These bulk temperatures result in thermal power extraction values of 17860 W for FEMuS and 17916 W for OpenFOAM. The table reports also the mean wall

	OpenFOAM	FEMuS	Prediction
$T_{b,out}$ $[K]$	572.6	573.1	573
$\dot{Q}$ $[W]$	17916	17860	17870
T_w $[K]$	594.42	589.6	-
T_b $[K]$	647.6	647.7	648
R_{PbLi} $[K/W]$	0.0030	0.0032	0.0031
h $[W/(m^2K)]$	5940	5431	5704
Nu	8.11	7.45	7.81

Table 4.2: Comparison between OpenFOAM and FEMuS results with the predicted quantities.

temperature, T_w, the bulk temperature throughout the entire domain, T_b, the convective thermal resistance, R_{PbLi}, the heat transfer coefficient, h and the Nusselt number, Nu. The thermal resistance has been computed using (4.1), where h is calculated as $h = \dot{q}/(T_w - T_b)$.

4.2.2 DNS Comparison

This section presents a more detailed analysis of the dynamic fields. The discussion of the dynamic behavior is feasible because the flow is fully developed. Moreover, given the forced convection nature of the flow, the non-fully developed temperature profile has no impact on the velocity-related fields.

As a result, we can provide a comprehensive comparison with the DNS data in the literature. Several DNS reference databases have been created for pipe flows, varying the friction Reynolds number ($Re_\tau = u_\tau \delta/\nu$). Specifically, the available DNS data correspond to the cases listed in Table 4.3. Each value of Re_τ is associated with a bulk Reynolds number (Re_b), which is also detailed in Table 4.3. The graph in Figure 4.6 presents the friction Reynolds numbers

Re_b	Re_τ	DNS
5300	181	[154, 155, 156]
5500	186	[157]
11700	360	[154]
19000	550	[154]
24580	685	[156]
37700	1000	[154]
44000	1140	[155, 156]

Table 4.3: Bulk Reynolds numbers and the corresponding friction Reynolds numbers of the DNS cases in literature.

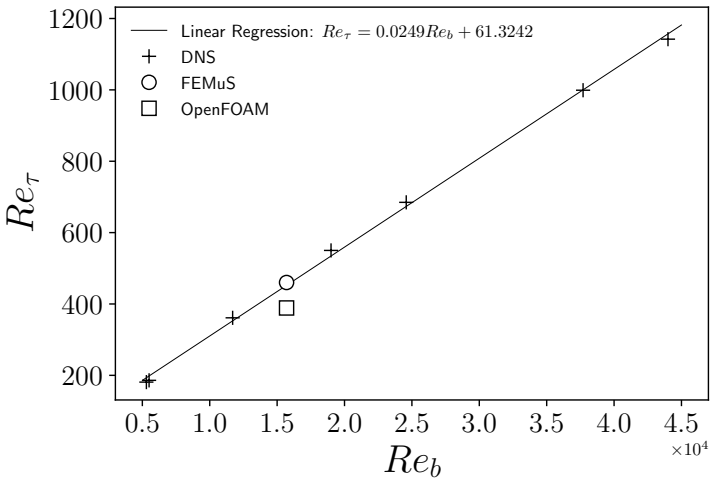

Figure 4.6: Regression line of the relationship between the bulk Reynolds numbers and the corresponding friction Reynolds numbers (cross markers). Values of the bulk Reynolds numbers computed by OpenFOAM (square marker) and FEMuS (circular marker) simulations for the heat exchanger case.

against the bulk Reynolds numbers. The DNS data of Table 4.3 and the computational results obtained using the FEMuS code and OpenFOAM. A linear regression fit, expressed as $Re_\tau = 0.0249\, Re_b + 61.3242$, is superimposed to highlight the trend of the data set. For a bulk Reynolds number of 1.57×10^4, the value of Re_τ calculated using the FEMuS code is 459, corresponding to a friction velocity of 0.00534, while the value computed by OpenFOAM is 389, with a friction velocity of 0.00452. As shown in the Re_τ-Re_b plot the FEMuS result aligns well with the linear trend. The precision in the accuracy of Re_τ gives confidence in the reliability of the model used to simulate turbulent flow. The correctness of the FEMuS turbulence model is further confirmed by the validation performed in [145]. In contrast, the OpenFOAM results show a noticeable deviation from the linear regression line, suggesting that these results cannot be considered an accurate prediction of the turbulent flow.

A qualitative comparison can be performed between the DNS profile of the dynamic fields and the results obtained with the two codes. Variables are normalized using wall units, such as friction velocity, u_τ, and kinematic viscosity, ν. The friction velocity is used to normalize the velocity as

$$v^+ = \frac{u}{u_\tau}\,. \tag{4.16}$$

The components of the Reynolds stress tensor and the turbulent kinetic energy are normalized as follows

$$\langle u'u' \rangle^+ = \frac{\langle u'u' \rangle}{u_\tau^2},\ \langle u'v' \rangle^+ = \frac{\langle u'v' \rangle}{u_\tau^2},\ \langle v'v' \rangle^+ = \frac{\langle v'v' \rangle}{u_\tau^2},\ k^+ = \frac{k}{u_\tau^2}. \tag{4.17}$$

All the variables are reported against the non-dimensional radial coordinates

$$x^+ = \frac{r u_\tau}{\nu}\,. \tag{4.18}$$

At the top of Figure 4.7 the non-dimensional velocity is reported, on the right a zoom-in view is provided. The turbulent kinetic energy is displayed in the center-left of the figure. For both variables the FEMuS and OpenFOAM solutions are provided. In the middle right and bottom of Figure 4.7, the Reynolds stress tensor components are reported exclusively for FEMuS code. The solutions provided by FEMuS and OpenFOAM are indicated as PbLi (F) and PbLi (OF). Comparisons have been made by plotting the DNS profiles for the nearest lower and higher values of Re_τ. The simulated case of the Lead-Lithium flow represents an intermediate case between the DNS cases with

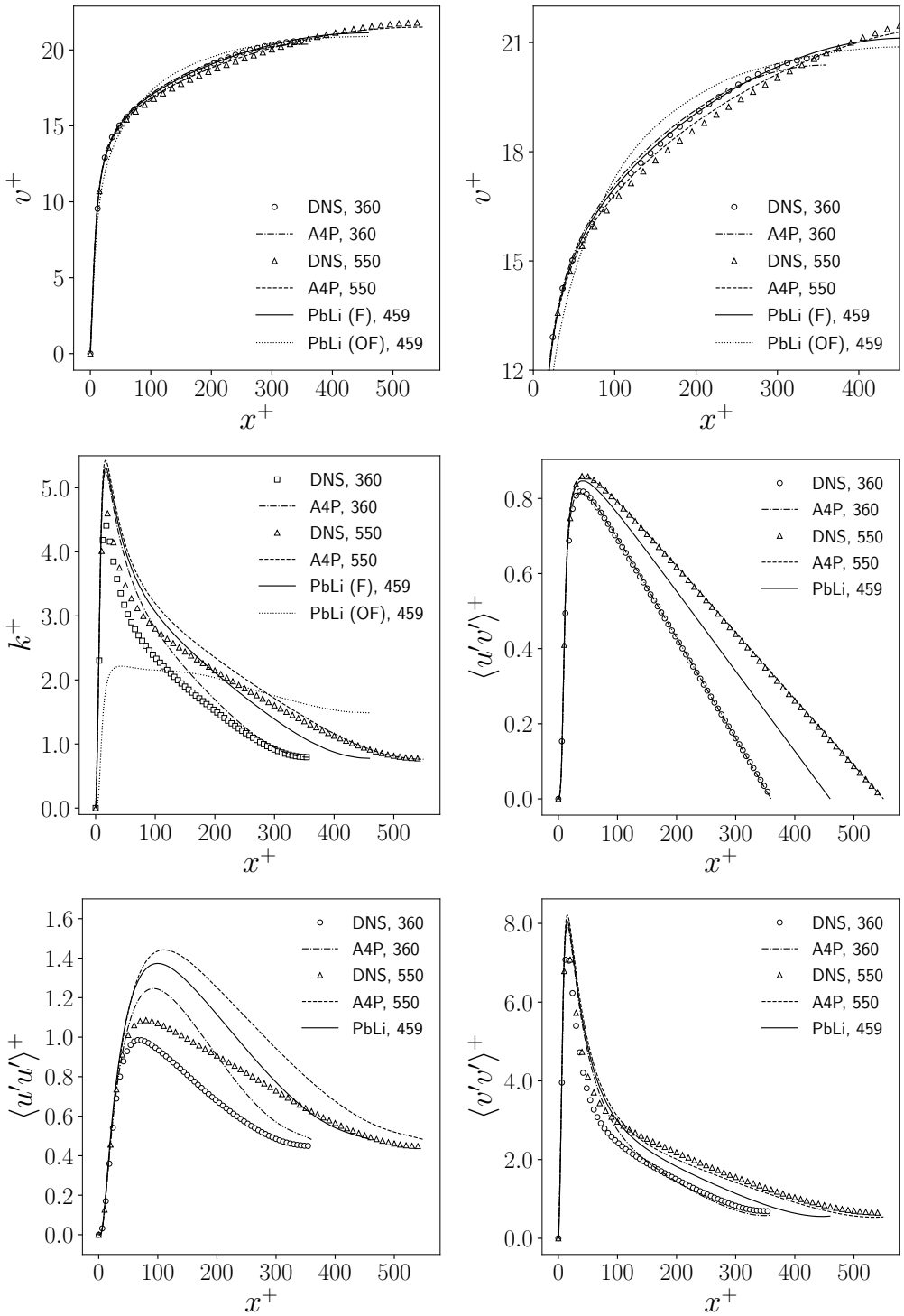

Figure 4.7: Comparison of the simulated cases, PbLi (F) for FEMuS and PbLi (OF) for OpenFOAM, with two DNS dataset ($Re_\tau = 360$ and $Re_\tau = 550$) from [154] and the results from [145] (A4P) for non-dimensional velocity, turbulent kinetic energy and non-dimensional components of the Reynolds stress tensor.

$Re_\tau = 360$ and $Re_\tau = 550$. Moreover, the corresponding solutions provided by the validation in [145] are shown for further comparisons. These latter simulations are labeled with the acronym A4P, which stands for "anisotropic four parameter", referring to the turbulence model used.

The non-dimensional velocity and turbulent kinetic energy profiles demonstrate good agreement with FEMuS solutions. For all the friction Reynolds numbers, the numerical results of the A4P solutions for the dimensionless velocity align perfectly with the reference data, and the new simulated case fits well as an intermediate profile between the two DNS cases. A similar observation can be made for k^+, where the FEMuS solutions consistently exhibit an overestimation of the near-wall peak in all cases. This suggests that the new case solution is also likely to overestimate its corresponding peak. However, the velocity profile computed using OpenFOAM shows a slightly different trend, which may explain the observed discrepancy in friction velocity. Furthermore, the turbulent kinetic energy profile provided by OpenFOAM significantly underestimates the near-wall peak and exhibits notable discrepancies compared to the DNS profile. The three components of the Reynolds stress tensor for the PbLi case provide similar considerations for the FEMuS simulation. The diagonal components, $\langle u'u' \rangle^+$ and $\langle v'v' \rangle^+$, show some discrepancies compared to the reference data. In particular, $\langle u'u' \rangle^+$ exhibits an overall overestimation, while $\langle v'v' \rangle^+$ tends to overestimate only the peak of the profile, providing satisfactory agreement in the bulk region. The new simulated case follows similar profiles for both variables. The off-diagonal component, $\langle u'v' \rangle^+$, on the other hand, aligns perfectly with the simulation results of the A4P validation for the two DNS cases provided. The PbLi case once again exhibits an intermediate profile, consistent with its expected behavior.

These results suggest that the FEMuS model, that includes formulations for the components of the Reynolds stress tensor and provides a more advanced near-wall treatment of the variables, can be used to accurately predict liquid metal turbulent flow. Thus, the conjugate heat transfer problem presented later in this chapter employs FEMuS to solve the fluid domain and OpenFOAM to solve the solid domain.

4.3 Pipe and Fins Simulations

In this section, the solid domain is analyzed and the results obtained using OpenFOAM-v11 are presented. The computational domain of the problem

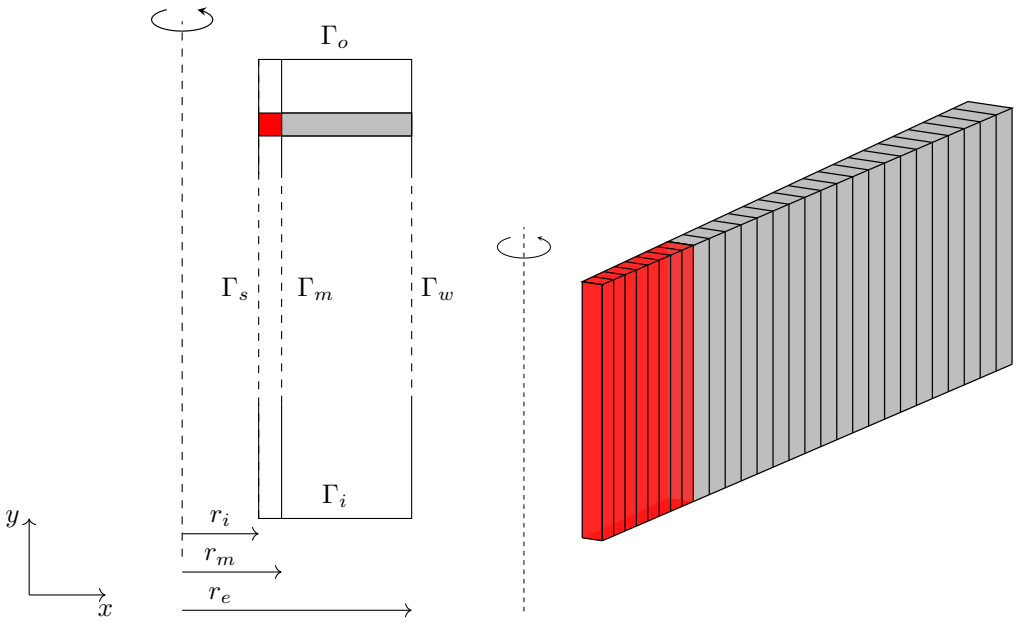

Figure 4.8: Schematic representation of the computational domains used for the EUROFER pipe and copper fins simulation by OpenFOAM (left). Mesh of a single cell along the length of the pipe (right), pipe and fins mesh are reported in red and in gray, respectively.

consists of two distinct regions: the EUROFER pipe and the copper fins. The scheme of the computational domain is illustrated in Figure 4.8 on the left, where the EUROFER pipe is highlighted in red, and the fins in gray. Figure 4.8 also shows a y-section of the mesh used for the OpenFOAM simulation. The mesh consists of 2576 cells, with 92 cells in the y-direction, and blocks of 8 cells in the radial direction for the pipe domain and 20 cells in the radial direction for the fins region. Both regions are treated in OpenFOAM using a wedge configuration, leveraging the symmetry of the problem. Given the internal radius of 0.015 m and a tube thickness of 0.004 m, the interface radius of the pipe is $r_m = 0.019$ m. The average length of the fins is 0.014 m, resulting in an external radius equal to $r_e = 0.033$ m.

The `solid` solver of OpenFOAM is used to calculate the temperature distribution for the multi-region domain. It solves the energy equation for

both solid regions, which is expressed as

$$\rho c \frac{\partial T}{\partial t} = \frac{\partial}{\partial x_i}\left(\lambda \frac{\partial T}{\partial x_i}\right), \tag{4.19}$$

where ρ is the density, c is the specific heat capacity and λ is the thermal conductivity. All these material properties listed in Table 4.1 are assumed to be constant; therefore, the built-in thermophysical model `constSolidThermo` is used. Regarding the boundary conditions for the pipe, a Dirichlet boundary condition is imposed at the internal wall in contact with the liquid metal flow, Γ_s. At the top and bottom of the pipe, corresponding to the inlet and outlet sections, adiabatic conditions are imposed. The front and back faces are treated as wedge boundaries in OpenFOAM to enforce the symmetry condition. At the interface between the two domains, Γ_m, OpenFOAM provides a coupled temperature boundary condition, ensuring the coupling of both temperature and wall heat flux between the two regions. The boundary conditions applied to the fins region are similar to those described for the pipe. The top and bottom surfaces are treated as adiabatic, while a fixed temperature of 338.85 K is assigned to Γ_w, which is the external wall in contact with air flow. This temperature corresponds to the mean air temperature of the PbLi-air heat exchanger.

In the following, two configurations are presented to simulate the interaction between the fins and the air domains.

Case 1. In the first configuration, Case 1, the fins region is treated as if it is composed entirely of solid material. This approach involves modifying the properties of the copper material to account for all thermal effects that occur in this region. In particular, the fins domain must account for the thermal resistance of the parallel block in Figure 4.3. This block introduces a thermal resistance calculated as

$$R_{eq} = \frac{1}{\frac{1}{R_{fins,air}+R_{fins}} + \frac{1}{R_{EU,air}}} = 0.0121\,\frac{K}{W}. \tag{4.20}$$

This computational domain corresponds to a hollow cylinder, and the thermal resistance introduced by this material must be equal to 0.0121 K/W. Using the definition of thermal resistance for a hollow cylinder as

$$R = \frac{\ln \frac{r_e}{r_m}}{2\pi\lambda L}, \tag{4.21}$$

we can deduce an equivalent thermal conductivity as

$$\lambda_{eq} = \frac{\ln \frac{r_e}{r_m}}{2\pi R_{eq} L} = 12.103 \, \frac{W}{mK}. \tag{4.22}$$

For Case 1, two simulations were performed. In the first, a uniform Dirich-

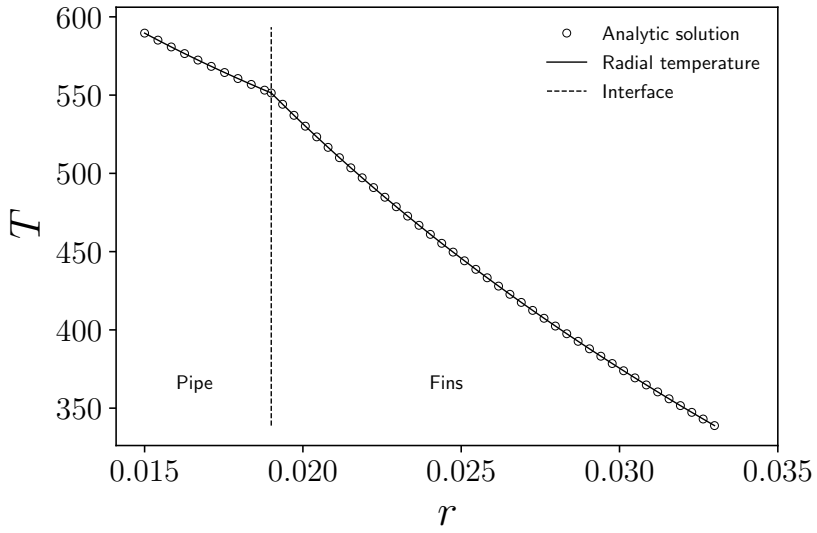

Figure 4.9: Radial temperature distribution for the solid multi-region simulation compared to the analytic solution (circle markers).

let boundary condition is applied to Γ_s, using the mean wall temperature computed by FEMuS in the previous section, which is 589.6 K. The solution is shown in Figure 4.9, where the radial temperature profile is compared to the analytical solution for the conduction problem. The analytical solution for heat conduction in two connected hollow cylinders with different thermal conductivities (λ_1 and λ_2) under steady-state conditions is derived as follows

$$T(r) = \begin{cases} C_1 \ln(r) + (T_{w,in} - C_1 \ln(r_i)) & \text{if } r_i \leq r \leq r_m \\ C_3 \ln(r) + (T_{w,ext} - C_3 \ln(r_e)) & \text{if } r_m \leq r \leq r_e \,, \end{cases} \tag{4.23}$$

where

$$C_1 = \frac{T_{w,ext} - T_{w,in}}{\ln(r_m) - \ln(r_i) - \frac{\lambda_1}{\lambda_2} \ln(r_m) + \frac{\lambda_1}{\lambda_2} \ln(r_e)}, \tag{4.24}$$

and

$$C_3 = \frac{\lambda_1}{\lambda_2} C_1. \tag{4.25}$$

Here, $T_{w,in}$ represents the temperature of the internal wall of the pipe, while $T_{w,ext}$ is the temperature applied to the external wall of the fin cylinder. The thermal conductivity λ_1 corresponds to the conductivity of EUROFER, while λ_2 is the equivalent thermal conductivity used for the fins region. Due to the uniform temperature imposed at the inner wall and the adiabatic boundaries at the inlet and outlet sections, the problem reduces to a one-dimensional radial heat conduction case, resulting in temperature profiles that is the solution of the analytical problem.

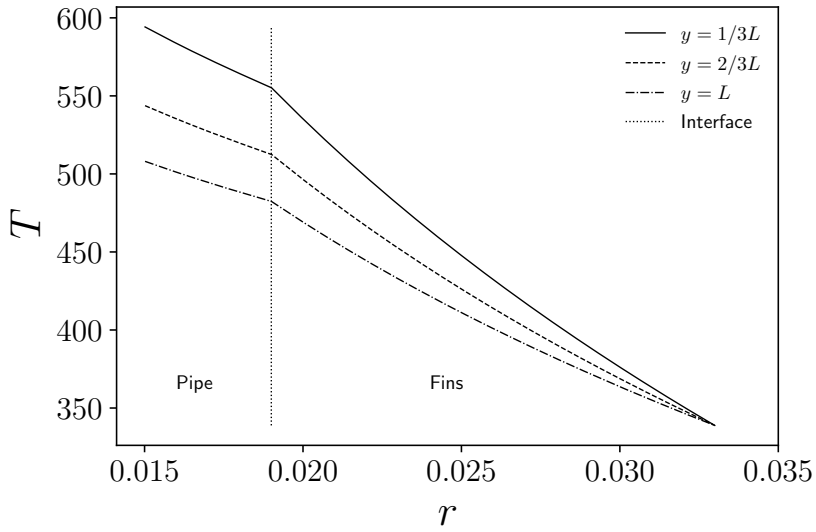

Figure 4.10: Radial temperature distribution for the solid multi-region simulation at three different heights of the pipe.

The second simulation for Case 1 is performed by applying the non-uniform wall temperature distribution, obtained from the lead-lithium flow simulated with FEMuS, as a Dirichlet boundary condition on Γ_s. Figure 4.10 shows the radial temperature profiles at three different heights along the pipe: one-third of the length, two-thirds of the length, and at the outlet surface. As expected, the non-uniform wall temperature results in a varying temperature profile along the solid domain.

Figure 4.11 compares the temperature distributions from the two simulations of Case 1. The distributions across the pipe and fins domains (separated with a black line) are shown for both simulations, with the aspect ratio adjusted for visual clarity. The simulation with a uniform Dirichlet boundary condition (left) results in parallel temperature isolines, while the non-uniform

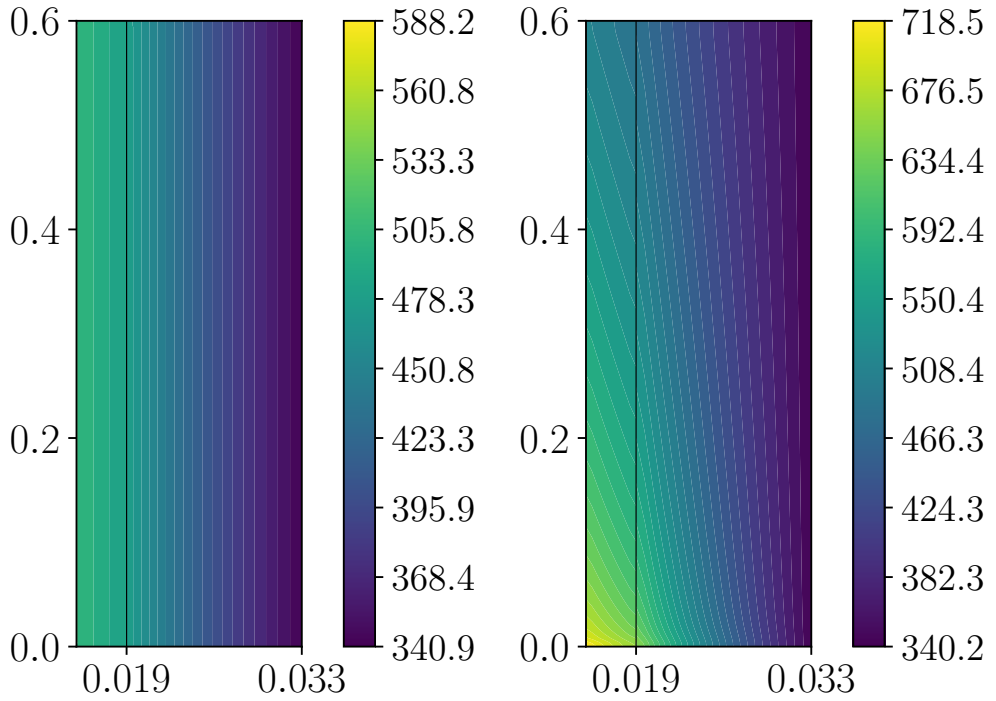

Figure 4.11: Temperature distribution for Case 1 for case with a uniform Dirichlet boundary condition on Γ_s (left) and a non-uniform Dirichlet boundary condition on Γ_s (right).

temperature distribution (right) produces curved isothermal lines.

Case 2. The second approach, Case 2, considers the fins region as a porous material, introducing a third region represented by the air domain. In this porous approach, the air region completely overlaps the fins region and is used only for thermal coupling purposes. To implement this case, the `fvModel` class in OpenFOAM is used, and the `interRegionHeatTransfer` coupling is applied. This functionality provided by OpenFOAM introduces a coupled quantity that is used by the solver of the two domains (fins and air) to compute the heat transferred between them. In particular, it introduces the heat source term into energy equations of both air and fins region as

$$\dot{Q} = hA_v(T_{nbr,mapped} - T), \tag{4.26}$$

where h is the heat transfer coefficient between air and fins, A_v is the exchange area per unit volume, T is the temperature variable of the region being considered (either air or fins) and $T_{nbr,mapped}$ is the coupled temperature field

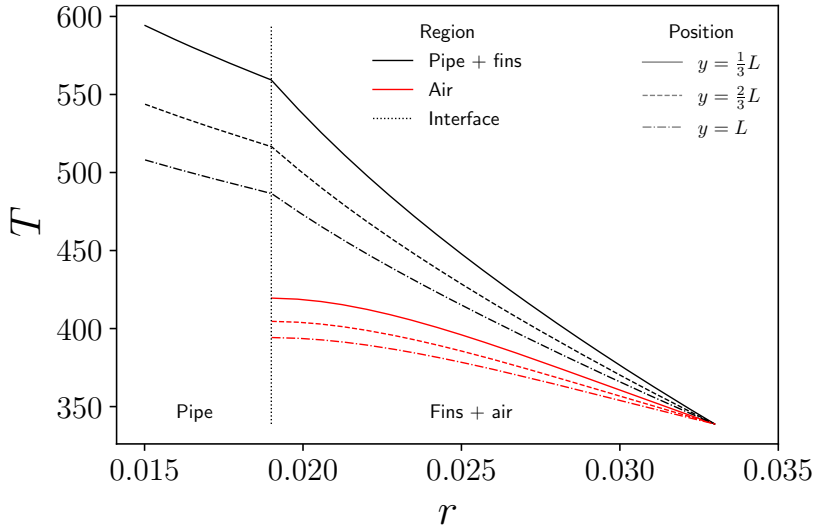

Figure 4.12: Radial temperature profiles for the solid multi-region simulation, comprising pipe and fins temperature profiles (black lines) and air temperature profiles (red lines). The temperature distributions refer to three positions along the heat exchanger: one-third of the length (solid lines), two-thirds of the length (dash-dotted lines), and at the outlet surface (dashed lines).

provided by the neighboring region (fins or air). The coupled temperature field is computed using the `intersection` method as `interpolationMethod`, a built-in interpolation algorithm available in OpenFOAM. This quantity is used in Equation (4.19) as a volumetric heat source in both regions, accounting for the heat exchanged between them. This approach allows for the specification of the heat exchange area of the fins, which is significantly larger than the area provided by the external surface of the hollow cylinder. In this simulation, the resistance due to convection between air and fins, $R_{fins,air}$, is accounted for through this contribution. The values of the heat transfer coefficient and the heat exchange area, presented in the previous section, are $140.4\,\mathrm{W/m^2K}$ and $0.630\,\mathrm{m^2}$, respectively. The remaining thermal resistances, $R_{fins} = 0.002\,\mathrm{K/W}$ and $R_{EU,air} = 0.123\,\mathrm{K/W}$, are modeled using equivalent thermal conduction. In particular, the resistance R_{fins} corresponds to heat conduction within the fin region, while $R_{EU,air}$ represents the resistance of the air domain. Applying Equation (4.21) to the external hollow cylinder, we obtain $\lambda_{eq} = 66.5\,\mathrm{W/(mK)}$ for the fins domain. Then, considering the volume fraction occupied by the fins, we reduce the thermal conductivity of the

hollow cylinder to $\lambda_{eq} = 9.76\,\mathrm{W/(mK)}$. For the air region, based on Equation (4.21) and the volume fraction occupied by air, an equivalent thermal conductivity of $\lambda_{eq} = 1.02\,\mathrm{W/(mK)}$ is imposed within the domain.

Figure 4.12 shows the radial temperature profiles computed in Case 2. The black lines correspond to the radial profile simulated in the solid regions comprising EUROFER tube and copper fins regions. The red lines display the temperature distributions for the air domain. Temperature profiles are shown at three different heights along the pipe to highlight the variations caused by the non-uniform wall temperature imposed as a boundary condition at the inner surface of the pipe. In this simulation, the thermal power transferred from the fins to the air is 1017 W. The total heat transfer rate, $\dot{Q}$, extracted from the external surface of the computational domain, Γ_w, is 17878 W. This includes 15122 W through the external surface of the fins region, accounting for both conduction and convection heat transfer, and 3774 W through the air's external surface, corresponding to the thermal power in the parallel branch of $R_{EU,air}$.

4.4 Conjugate Heat Transfer Application

In this setup, the entire heat exchanger is simulated using the boundary coupling application, where FEMuS and OpenFOAM codes perform the liquid metal flow and solid regions simulations, respectively. Figure 4.13 illustrates the scheme of the computational domain. The orange zone represents the liquid metal domain, the red zone corresponds to the EUROFER pipe region, and the gray zone indicates the fins region. On the right, a single cell along the y direction of the heat exchanger mesh is reported. This geometry includes two distinct interfaces: the first, Γ_{m_1}, connects FEMuS and OpenFOAM and represents the interface between the fluid domain (PbLi flow) and the solid domain; the second, Γ_{m_2}, is the interface between the two solid regions, specifically between the EUROFER pipe and the fins region.

The integration between FEMuS and OpenFOAM across the Γ_{m_1} interface is handled using the coupling application for boundary data transfer described in Chapter 2. In the context of the conjugate heat transfer problem, the wall heat flux in Γ_{m_1}, provided by FEMuS, is transferred to OpenFOAM. These values are used as non-homogeneous Neumann boundary conditions on the internal wall of the pipe geometry. In turn, OpenFOAM provides the wall temperature to FEMuS, which uses it as a non-homogeneous Dirichlet

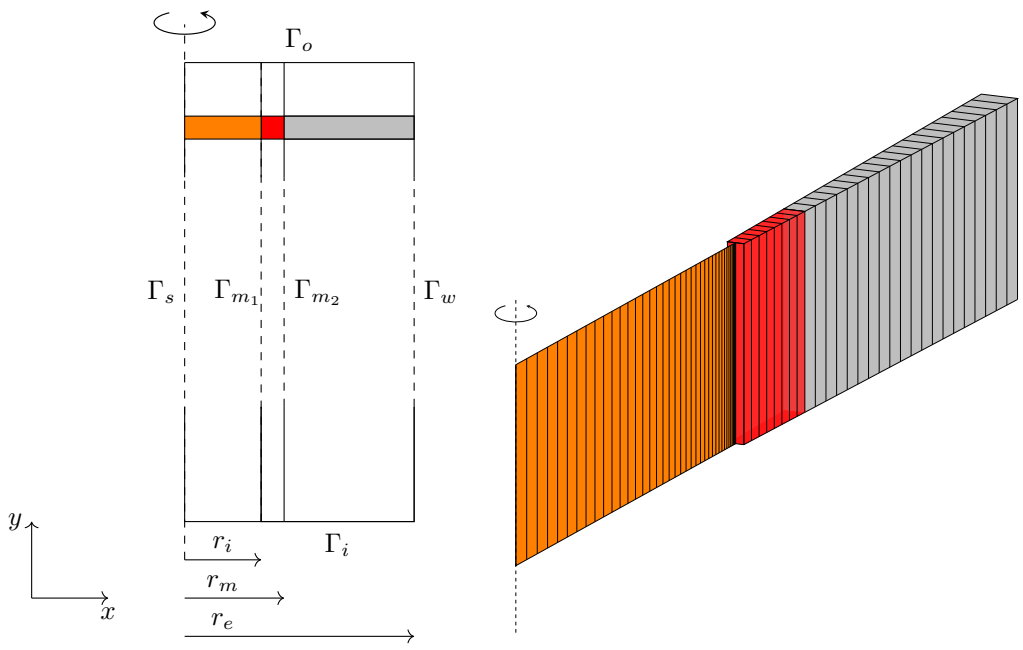

Figure 4.13: Schematic representation of the computational domains used for the CHT simulations (left). Mesh of a single cell along the length of the heat exchanger (right). It comprises the liquid metal domain (orange), EUROFER pipe domain (red) and copper fins domain (gray).

boundary condition for the temperature equation.

The simulation of the solid domain and the coupling through the Γ_{m_2} interface are carried out using two different configurations.

Case A. The first approach uses the coupling application not only to connect FEMuS and OpenFOAM across Γ_{m_1}, but also to couple the two solid regions managed by OpenFOAM across Γ_{m_2}. At this interface, the wall heat flux from the EUROFER pipe is transferred as a non-homogeneous Neumann boundary condition to the copper fins region. In return, the copper fins region provides the wall temperature as a Dirichlet boundary condition back to the pipe. The coupling configuration adopted for this approach, referred to as Case A, is schematically illustrated in Figure 4.14.

In this scenario, the fins domain is treated as described in Case 1 of Section 4.3, where the thermal conductivity used for the fins region is taken as calculated in (4.22). Figure 4.15 shows the temperature profiles at three different heights along the heat exchanger. The solid line corresponds to 1/3 of

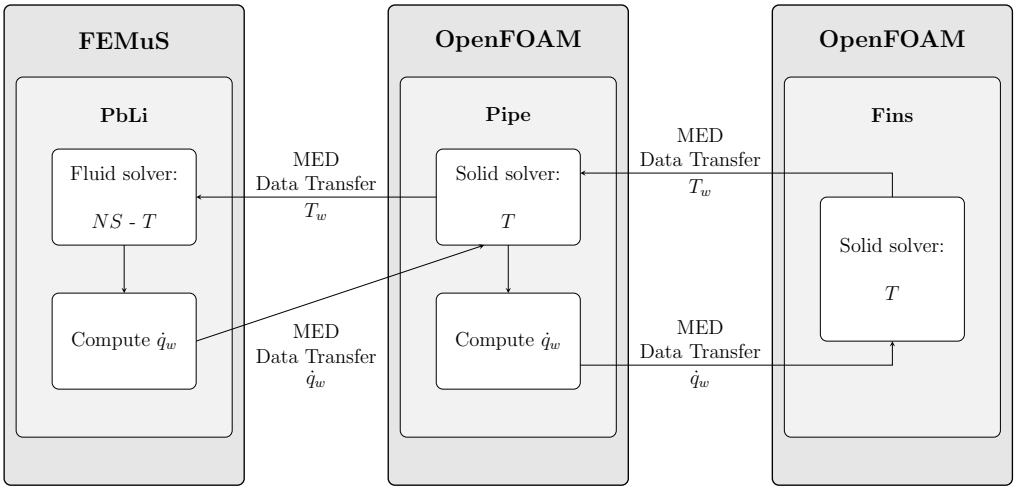

Figure 4.14: Schematic representation of the coupling algorithm for Case A.

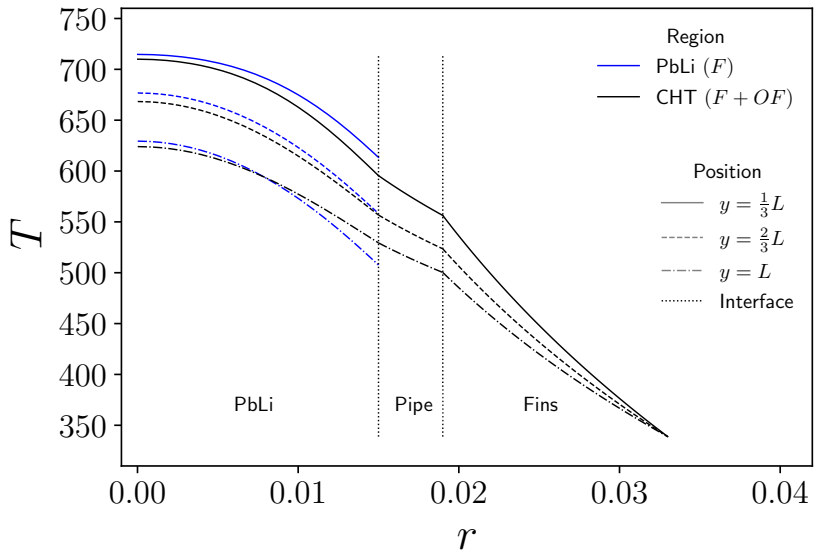

Figure 4.15: Multi-region radial temperature profiles at three sections of the pipe, $y/L = 1/3$ (solid), $y/L = 2/3$ (dashed) and $y/L = 1$ (dash-dotted), for both the CHT simulation (black) and PbLi flow simulation as in Section 4.2 (blue).

the total pipe length, the dashed profile is taken at $2/3L$, and the remaining profile represents the outlet section of the pipe. The black lines correspond to the radial temperature profile across liquid metal, pipe, and fins computed in

the CHT simulation. For comparison purposes, in blue, we have reported the temperature profiles for the fluid flow simulation performed using FEMuS, as described in 4.2.1. This simulation is referred to as PbLi. This PbLi simulation is performed with a fixed wall heat flux applied to the external wall of the fluid domain, here Γ_{m_1}, whereas in the CHT simulation, the wall heat flux on the same surface is non-homogeneous. This difference results in non-uniform cooling along the length of the pipe. At the inlet, the temperature difference between the PbLi and the pipe is at its maximum, leading to a maximum wall heat flux at the interfaces. As the flow progresses along the pipe, the temperature difference between the solid and the fluid domains decreases, leading to a progressive reduction in the wall heat flux between the regions. For completeness, Figure 4.16 shows the two-dimensional temperature field

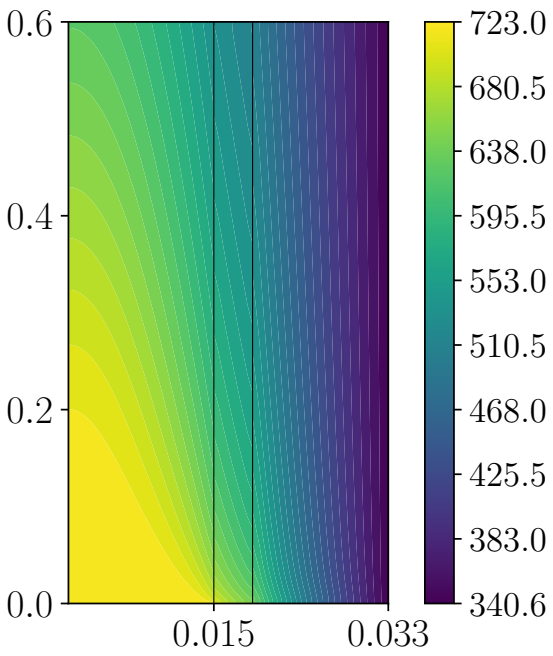

Figure 4.16: Temperature distribution in the liquid metal flow, EUROFER pipe, and fins domains, as obtained from the Case A simulation.

in the three regions involved in the simulation of Case A. A solid black line separates each of the three regions. From left to right, the field distribution is shown within the liquid metal, the pipe, and the fins. Table 4.4 summarizes the calculated results and compares them with the predicted values. It can be observed that the bulk temperature at the pipe outlet satisfies the

	CHT	Prediction
$T_{b,out}$ $[K]$	572.1	573
$\dot{Q}$ $[W]$	17964	17870
T_w $[K]$	585.8	-
T_b $[K]$	633.6	648
R_{PbLi} $[K/W]$	0.0028	0.0031
h $[W/(m^2K)]$	6413	5704
Nu	8.79	7.81

Table 4.4: Comparison between CHT results with the predicted quantities for Case A.

design constraint, reaching 572.1 K, which is slightly lower than the target value of 573 K, which corresponds to a total cooling power of 17964 W. The CFD simulation provides a heat transfer coefficient of 6413 $\mathrm{W/m^2K}$, which is higher than the predicted value. This suggests that the convective resistance of the liquid metal flow is lower than initially estimated and, consequently, the simulated Nusselt number exceeds the prediction.

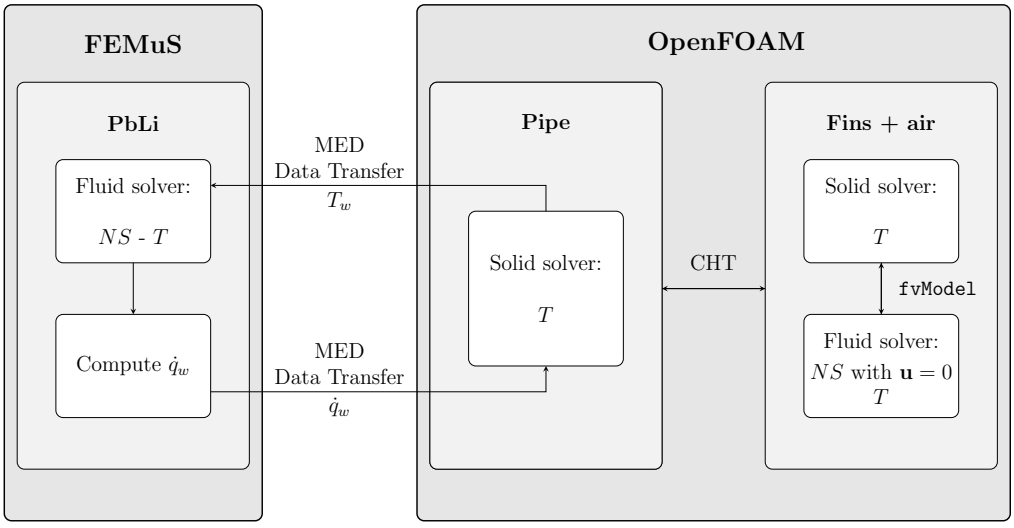

Figure 4.17: Schematic representation of the coupling algorithm for Case B.

Case B. The second CHT simulation adopts the full configuration setup, involving the air domain similarly to Case 2 of the previous section. Unlike

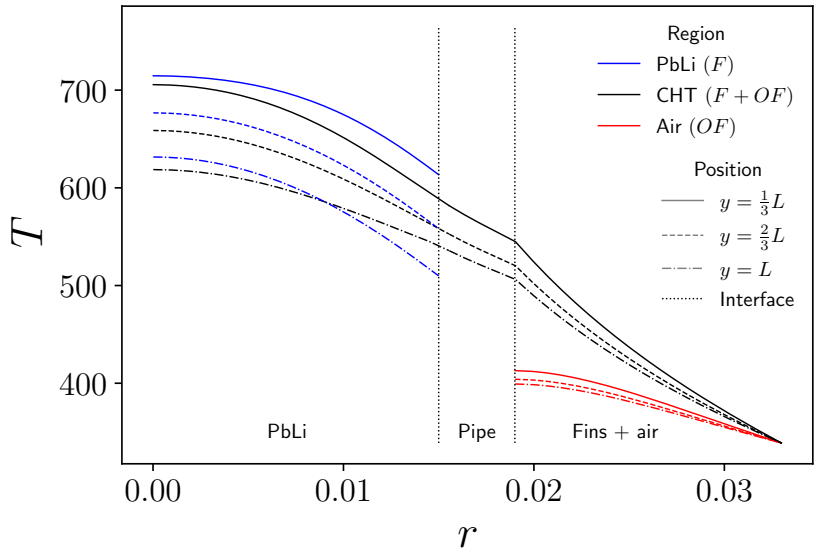

Figure 4.18: Multi-region radial temperature profiles at three sections of the pipe, $y/L = 1/3$ (solid), $y/L = 2/3$ (dashed) and $y/L = 1$ (dash-dotted), for the CHT simulation (black), air (red) and PbLi flow as in Section 4.2 (blue).

the boundary data transfer of Case A, this configuration requires the use of the dedicated wrapper developed for the OpenFOAM multi-region solver. In this setup, the Γ_{m_1} interface is managed through the MED library. The MED communication on Γ_{m_1} has been implemented between FEMuS and the multi-region application of OpenFOAM. Consequently, the Γ_{m_2} interface, located between the EUROFER pipe and the copper fins region, is handled internally by the OpenFOAM CHT solver, without any external coupling. Additionally, the interaction between the fins region and the air domain is modeled using OpenFOAM's `fvModel` functionality, which applies volumetric coupling within the two domains. Figure 4.17 shows the coupling application for Case B.

Similar considerations to those discussed for Case A can be drawn by examining the radial temperature profiles shown in Figure 4.18, evaluated at the same three heights along the pipe. The temperature profiles for the liquid metal, EUROFER pipe, and fins are plotted in black, while the air domain profile is shown in red. For reference, the temperature distribution obtained from the FEMuS solver in Section 4.2.1 is indicated in blue. Table 4.5 summarizes the results of the main quantities compared to the predicted values. The CHT simulation results in a Nusselt number of 7.97, which is

	CHT	Prediction
$T_{b,out}$ $[K]$	571.4	573
$\dot{Q}$ $[W]$	18065	17870
T_w $[K]$	579.1	-
T_b $[K]$	634.1	648
R_{PbLi} $[K/W]$	0.0030	0.0031
h $[W/(m^2K)]$	5810	5704
Nu	7.97	7.81

Table 4.5: Comparison between CHT results with the predicted quantities for Case A.

slightly higher than the predicted value of 7.81. This values indicates that the convective heat transfer of liquid metal is marginally higher than predicted.

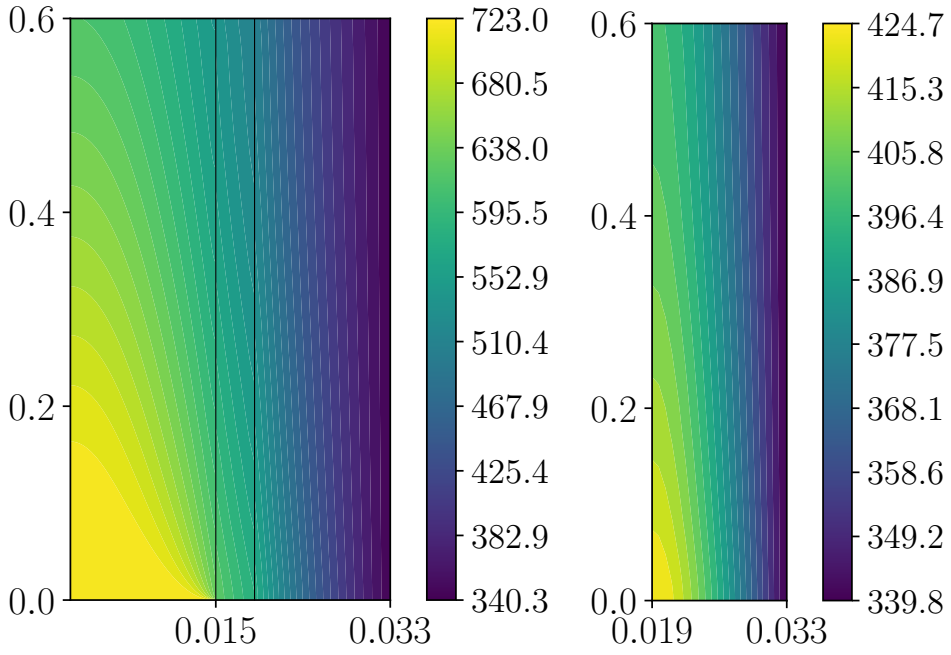

Figure 4.19: Temperature distribution in the liquid metal flow, EUROFER pipe, and fins domains (left), and in the air domain (right), as obtained from the Case B simulation.

The 2D temperature field distribution is shown in Figure 4.19. On the left, the temperature field for the liquid metal, the pipe, and the fins domains are

displayed, while on the right, the temperature distribution for the air domain is presented.

The results from Case A and Case B may suggest that the theoretical prediction of the Nusselt number is underestimated and that the actual heat flux is higher than initially expected. The liquid metal's heat transfer performance allows the heat exchanger to operate effectively, achieving an outlet temperature that satisfies the required constraint. Moreover, the resulting outlet temperature remains sufficiently above the critical value. This result provides a margin of safety that protects against potential discrepancies between the computational model and the real system configuration. Furthermore, the presence of a reheater downstream of the PbLi heat exchanger ensures that the critical temperature threshold is never reached.

Conclusions

The primary objective of this dissertation is to develop a numerical platform for studying complex engineering systems involving liquid metals. These materials are a vibrant research topic because of their promising applicability in emerging energy technologies, such as Concentrated Solar Power plants and Generation IV nuclear reactors. In these power plants, the research community is actively investigating the use of liquid metals as heat transfer fluids, leveraging their exceptional thermal properties to address the demands of high-performance thermal management.

Accurate modeling of turbulent natural convection and heat transfer is essential for liquid metals. It is important to overcome the limitations of eddy viscosity and eddy diffusivity models commonly implemented in commercial software. This challenge can be tackled by using more advanced turbulence closure models. In the FEMuS code, indeed, the Reynolds stress tensor and the turbulent heat flux are modeled using the Explicit Algebraic class of models, and the four-parameter model is used to close the system of equations. In this Thesis, buoyancy terms have been incorporated into the EASM and EAHFM equations to improve the prediction of turbulent natural convection flows.

Within the numerical platform framework, the coupling between the FEM code FEMuS and the FVM code OpenFOAM has been realized using the open-source MED library. The volume data transfer algorithm has been validated by studying a differentially Heated Cavity domain under a laminar natural convection regime for standard fluids, such as air and water. In

these coupling applications, the exchanged quantities are the temperature and velocity fields across the entire domain. On the other hand, the boundary data transfer algorithm has been validated using a Conjugate Heat Transfer problem, where a natural convection flow inside a square cavity is thermally coupled with solid domains on the side walls. The interface coupling has involved the mutual exchange of wall temperature and heat flux between the domains. The results obtained from the DHC and CHT problems have validated the implemented coupling algorithm.

The turbulent natural convection in liquid metals has been investigated using monolithic and code coupling approaches. The buoyant algebraic model has been validated in a DHC configuration for three cases involving liquid metals. The FEMuS results have been compared with both DNS benchmark and experimental data. Including buoyancy terms considerably improved the prediction of the variables in all three cases. However, some discrepancies have been found in dynamic field simulation, particularly in predicting $k-\omega$ variables.

The same configuration has been investigated using OpenFOAM's built-in turbulent models. All of them rely on the eddy viscosity and eddy diffusivity models. The comparison with DNS and experimental data shows a good alignment, particularly for the velocity profile for all the turbulent models.

A third approach has been used to investigate this phenomenon. The volume data transfer algorithm has been used to leverage the thermal turbulent model used in FEMuS with the dynamic, turbulent model implemented in OpenFOAM. We could observe a general improvement in the prediction of velocity and temperature fields meaning that both solvers could benefit from a coupled approach.

Finally, a realistic case of a liquid metal heat exchanger involving a Lead-Lithium alloy flowing through a cylindrical pipe with fins has been analyzed. Separate simulations were conducted for the fluid and solid domains using various configurations. The turbulent flow of the liquid metal was simulated using FEMuS with the anisotropic four-parameter model and OpenFOAM with the $k-\omega$ SST model. A comparison with DNS data demonstrated the greater reliability of the FEMuS solutions with respect to OpenFOAM ones in turbulent forced regimes. The pipe and fins regions were simulated in two different configurations using the solid solver in OpenFOAM, both employing OpenFOAM's built-in coupling paradigm. Within the numerical coupling framework, two conjugate heat transfer configurations have been employed to

thermally couple the fluid region, solved by FEMuS, with the solid regions, solved by OpenFOAM. The first simulation used the MED-based coupling to thermally couple all three domains: the EUROFER pipe was coupled with both the liquid metal flow and the copper fins region through the developed coupling application. In the second simulation, the coupling between the fluid flow and the pipe was handled using the MED interface, while the interaction between the pipe and the fins was managed internally by OpenFOAM's native multi-region solver. The interface coupling enabled the analysis of the heat transfer performance of the liquid metal under a non-uniform wall heat flux. Specifically, it allowed for a more precise calculation of the convection resistance of liquid metals and comparison with the predicted value.

It can be concluded that the coupling application proves to be a valuable tool for simulating complex phenomena that stand-alone commercial codes cannot fully or accurately capture. Its modular design allows for future extension to other numerical solvers and the inclusion of additional physical models, enabling accurate simulations of more advanced multiphysics systems.

List of Figures

List of Tables

Bibliography

[1] L. Marocco, G. Cammi, J. Flesch, and T. Wetzel, "Numerical analysis of a solar tower receiver tube operated with liquid metals," *International Journal of Thermal Sciences*, vol. 105, pp. 22–35, 2016. 3

[2] D. Frazer, E. Stergar, C. Cionea, and P. Hosemann, "Liquid metal as a heat transport fluid for thermal solar power applications," *Energy Procedia*, vol. 49, pp. 627–636, 2014. 3

[3] X. Cheng and N.-i. Tak, "Investigation on turbulent heat transfer to lead–bismuth eutectic flows in circular tubes for nuclear applications," *Nuclear Engineering and Design*, vol. 236, no. 4, pp. 385–393, 2006. 3

[4] J. Pacio, K. Litfin, A. Batta, M. Viellieber, A. Class, H. Doolaard, F. Roelofs, S. Manservisi, F. Menghini, and M. Böttcher, "Heat transfer to liquid metals in a hexagonal rod bundle with grid spacers: Experimental and simulation results," *Nuclear Engineering and Design*, vol. 290, pp. 27–39, 2015. 3

[5] G. Grasso, C. Petrovich, D. Mattioli, C. Artioli, P. Sciora, D. Gugiu, G. Bandini, E. Bubelis, and K. Mikityuk, "The core design of alfred, a demonstrator for the european lead-cooled reactors," *Nuclear Engineering and Design*, vol. 278, pp. 287–301, 2014. 3

[6] A. Alemberti, M. Caramello, M. Frignani, G. Grasso, F. Merli, G. Morresi, and M. Tarantino, "Alfred reactor coolant system design," *Nuclear engineering and design*, vol. 370, p. 110884, 2020. 3

[7] M. Tarantino, M. Angiolini, S. Bassini, S. Cataldo, C. Ciantelli, C. Cristalli, A. Del Nevo, I. Di Piazza, D. Diamanti, M. Eboli, *et al.*, "Overview on lead-cooled fast reactor design and related technologies development in enea," *Energies*, vol. 14, no. 16, p. 5157, 2021. 3

[8] M. Caramello, M. Frignani, G. Grasso, M. Tarantino, D. Martelli, P. Lorusso, and I. Di Piazza, "Improvement of alfred thermal hydraulics through experiments and numerical studies," *Nuclear Engineering and Design*, vol. 409, p. 112365, 2023. 3

[9] NextTower, "Advanced materials solutions for next generation high efficiency concentrated solar power (csp) tower systems." 3, 157

[10] F. M. Aprà, S. Smit, R. Sterling, and T. Loureiro, "Overview of the enablers and barriers for a wider deployment of csp tower technology in europe," *Clean Technologies*, vol. 3, no. 2, pp. 377–394, 2021. 3, 157

[11] V. Casalegno, L. Ferrari, M. Jimenez Fuentes, A. De Zanet, S. Gianella, M. Ferraris, and V. M. Candelario, "High-performance sic–based solar receivers for csp: component manufacturing and joining," *Materials*, vol. 14, no. 16, p. 4687, 2021. 3, 157

[12] S. B. Pope, "Turbulent flows," *Measurement Science and Technology*, vol. 12, no. 11, pp. 2020–2021, 2001. 9

[13] L. F. Richardson, *Weather prediction by numerical process*. University Press, 1922. 17

[14] A. N. Kolmogorov, "Dissipation of energy in the locally isotropic turbulence," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 434, no. 1890, pp. 15–17, 1991. 17

[15] A. N. Kolmogorov, "The local structure of turbulence in incompressible viscous fluid for very large reynolds numbers," *Proceedings of the Royal Society of London. Series A: Mathematical and Physical Sciences*, vol. 434, no. 1890, pp. 9–13, 1991. 17

[16] O. Reynolds, "Iv. on the dynamical theory of incompressible viscous fluids and the determination of the criterion," *Philosophical transactions of the royal society of london.(a.)*, no. 186, pp. 123–164, 1895. 19

[17] J. Boussinesq, *Essai sur la théorie des eaux courantes*. Imprimerie nationale, 1877. 23

[18] L. Prandtl, "7. bericht über untersuchungen zur ausgebildeten turbulenz," *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, vol. 5, no. 2, pp. 136–139, 1925. 23, 26

[19] E. R. Van Driest, "On turbulent flow near a wall," *Journal of the aeronautical sciences*, vol. 23, no. 11, pp. 1007–1011, 1956. 24

[20] A. N. Kolmogorov, "Equations of turbulent motion in an incompressible fluid," in *Dokl. Akad. Nauk SSSR*, vol. 30, pp. 299–303, 1941. 27, 30

[21] P.-Y. Chou, "On velocity correlations and the solutions of the equations of turbulent fluctuation," *Quarterly of applied mathematics*, vol. 3, no. 1, pp. 38–54, 1945. 27

[22] B. E. Launder and D. B. Spalding, "The numerical computation of turbulent flows," *Computer Methods in Applied Mechanics and Engineering*, vol. 3, pp. 269–289, 1974. 27, 147

[23] F. H. Harlow and P. I. Nakayama, "Turbulence transport equations," *The Physics of Fluids*, vol. 10, no. 11, pp. 2323–2332, 1967. 28

[24] K. Hanjalić and B. E. Launder, "A reynolds stress model of turbulence and its application to thin shear flows," *Journal of fluid Mechanics*, vol. 52, no. 4, pp. 609–638, 1972. 28

[25] S. Manservisi and F. Menghini, "A cfd four parameter heat transfer turbulence model for engineering applications in heavy liquid metals," *International Journal of Heat and Mass Transfer*, vol. 69, pp. 312–326, 2014. 29, 30, 35, 40, 55

[26] V. Yakhot and S. A. Orszag, "Renormalization group analysis of turbulence. i. basic theory," *Journal of Scientific Computing*, vol. 1, no. 1, pp. 3–51, 1986. 29, 147

[27] V. Yakhot, S. A. Orszag, S. Thangam, T. Gatski, and C. Speziale, "Development of turbulence models for shear flows by a double expansion technique," *Physics of Fluids A: Fluid Dynamics*, vol. 4, no. 7, pp. 1510–1520, 1992. 29, 30

[28] D. C. Wilcox, "Re-assessment of the scale-determining equation for advanced turbulence models," *AIAA Journal*, vol. 26, no. 11, pp. 1299–1310, 1988. 31, 147

[29] D. Wilcox, "Turbulence modeling for cfd," *DCW industries, La Canada*, 1998. 31

[30] D. C. Wilcox, "Formulation of the kw turbulence model revisited," *AIAA journal*, vol. 46, no. 11, pp. 2823–2838, 2008. 32

[31] D. Cerroni, R. Da Viá, S. Manservisi, F. Menghini, G. Pozzetti, and R. Scardovelli, "Numerical validation of a κ-ω-$\kappa\theta$-$\omega\theta$ heat transfer turbulence model for heavy liquid metals," in *Journal of Physics: Conference Series*, vol. 655, p. 012046, IOP Publishing, 2015. 32, 39

[32] F. Menter, "Zonal two equation kw turbulence models for aerodynamic flows," in *23rd fluid dynamics, plasmadynamics, and lasers conference*, p. 2906, 1993. 33

[33] F. R. Menter, "Two-equation eddy-viscosity turbulence models for engineering applications," *AIAA Journal*, vol. 32, no. 8, pp. 1598–1605, 1994. 33, 147

[34] D. R. Chapman and G. D. Kuhn, "The limiting behaviour of turbulence near a wall," *Journal of Fluid Mechanics*, vol. 170, pp. 265–292, 1986. 34, 35

[35] Y. Nagano and M. Tagawa, "An improved k-e model for boundary layer flows," *Journal of Fluids Engineering-transactions of The Asme*, vol. 112, pp. 33–39, 1990. 34

[36] K. Abe, T. Kondoh, and Y. Nagano, "A new turbulence model for predicting fluid flow and heat transfer in separating and reattaching flows—i. flow field calculations," *International journal of heat and mass transfer*, vol. 37, no. 1, pp. 139–151, 1994. 35

[37] K. Abe, T. Kondoh, and Y. Nagano, "On reynolds-stress expressions and near-wall scaling parameters for predicting wall and homogeneous turbulent shear flows," *International journal of heat and fluid flow*, vol. 18, no. 3, pp. 266–282, 1997. 35, 42, 47

[38] H. Kawamura, H. Abe, and Y. Matsuo, "Dns of turbulent heat transfer in channel flow with respect to reynolds and prandtl number effects," *International Journal of Heat and Fluid Flow*, vol. 20, no. 3, pp. 196–207, 1999. 36

[39] B. E. Launder, "Heat and mass transport," *Turbulence*, pp. 231–287, 2005. 37

[40] R. Da Vià and S. Manservisi, "Numerical simulation of forced and mixed convection turbulent liquid sodium flow over a vertical backward facing step with a four parameter turbulence model," *International Journal of Heat and Mass Transfer*, vol. 135, pp. 591–603, 2019. 39

[41] W. M. Kays, "Turbulent prandtl number. where are we?," *ASME Journal of Heat Transfer*, vol. 116, no. 2, pp. 284–295, 1994. 41

[42] S. B. Pope, "A more general effective-viscosity hypothesis," *Journal of Fluid Mechanics*, vol. 72, no. 2, pp. 331–340, 1975. 42, 44, 45, 46

[43] W. Rodi, "A new algebraic relation for calculating the reynolds stresses," in *Gesellschaft Angewandte Mathematik und Mechanik Workshop Paris France*, vol. 56, p. 219, 1976. 42

[44] B. E. Launder, G. J. Reece, and W. Rodi, "Progress in the development of a reynolds-stress turbulence closure," *Journal of fluid mechanics*, vol. 68, no. 3, pp. 537–566, 1975. 42

[45] H. Hattori and Y. Nagano, "Nonlinear two-equation model taking into account the wall-limiting behavior and redistribution of stress components," *Theoretical and Computational Fluid Dynamics*, vol. 17, no. 5, pp. 313–330, 2004. 42

[46] H. Hattori, A. Morita, and Y. Nagano, "Nonlinear eddy diffusivity models reflecting buoyancy effect for wall-shear flows and heat transfer," *International journal of heat and fluid flow*, vol. 27, no. 4, pp. 671–683, 2006. 42, 48, 49, 52, 56, 199

[47] T. B. Gatski and C. G. Speziale, "On explicit algebraic stress models for complex turbulent flows," *Journal of fluid Mechanics*, vol. 254, pp. 59–78, 1993. 44, 47

[48] C. G. Speziale, S. Sarkar, and T. B. Gatski, "Modelling the pressure–strain correlation of turbulence: an invariant dynamical systems approach," *Journal of fluid mechanics*, vol. 227, pp. 245–272, 1991. 44

[49] R. So, P. Vimala, L. Jin, C. Zhao, and T. Gatski, "Accounting for buoyancy effects in the explicit algebraic stress model: homogeneous turbulent shear flows," *Theoretical and Computational Fluid Dynamics*, vol. 15, pp. 283–302, 2002. 46

[50] Y. Nagano, H. Hattori, and K. Abe, "Modeling the turbulent heat and momentum transfer in flows under different thermal conditions," *Fluid dynamics research*, vol. 20, no. 1-6, pp. 127–142, 1997. 47

[51] Y. Nagano and H. Hattori, "A new low-reynolds-number turbulence model with hybrid time-scales of mean flow and turbulence for complex wall flows," 01 2003. 48

[52] K.-i. Abe, T. Kondoh, and Y. Nagano, "A two-equation heat transfer model reflecting second-moment closures for wall and free turbulent flows," *International journal of heat and fluid flow*, vol. 17, no. 3, pp. 228–237, 1996. 48, 50

[53] B. E. Launder, *Heat and Mass Transport*, p. 231–287. Berlin, Heidelberg: Springer Berlin Heidelberg, 1976. 50

[54] T. Craft, N. Ince, and B. Launder, "Recent developments in second-moment closure for buoyancy-affected flows," *Dynamics of atmospheres and oceans*, vol. 23, no. 1-4, pp. 99–114, 1996. 50

[55] R. Da Via, S. Manservisi, and F. Menghini, "A k–ω–kθ–$\omega\theta$ four parameter logarithmic turbulence model for liquid metals," *International Journal of Heat and Mass Transfer*, vol. 101, pp. 1030–1041, 2016. 54

[56] R. Da Và, V. Giovacchini, and S. Manservisi, "A logarithmic turbulent heat transfer model in applications with liquid metals for pr= 0.01–0.025," *Applied Sciences*, vol. 10, no. 12, p. 4337, 2020. 54

[57] G. Barbi, V. Giovacchini, and S. Manservisi, "A new anisotropic four-parameter turbulence model for low prandtl number fluids," *Fluids*, vol. 7, no. 1, p. 6, 2021. 54, 55

[58] F. Ilinca, D. Pelletier, *et al.*, "A unified finite element algorithm for two-equation models of turbulence," *Computers & fluids*, vol. 27, no. 3, pp. 291–310, 1998. 54

[59] D. Drikakis, M. Frank, and G. Tabor, "Multiscale computational fluid dynamics," *Energies*, vol. 12, no. 17, p. 3272, 2019. 57

[60] D. Groen, S. J. Zasada, and P. V. Coveney, "Survey of multiscale and multiphysics applications and communities," *Computing in Science & Engineering*, vol. 16, no. 2, pp. 34–43, 2013. 57

[61] M. E. Cordero, S. Uribe, L. G. Zárate, R. N. Rangel, A. Regalado-Méndez, and E. P. Reyes, "Cfd modelling of coupled multiphysics-multiscale engineering cases," *Comput. Fluid Dyn.-Basic Instruments Appl. Sci*, 2018. 57

[62] J. G. Michopoulos, C. Farhat, and J. Fish, "Modeling and simulation of multiphysics systems," *Journal of Computing and Information Science in Engineering*, vol. 5, no. 3, pp. 198–213, 2005. 57

[63] B. Engquist, P. Lötstedt, and O. Runborg, *Multiscale modeling and simulation in science*, vol. 66. Springer Science & Business Media, 2009. 57

[64] M. Peksen, *Multiphysics Modeling: Materials, Components, and Systems*. Academic Press, 2018. 57

[65] H. Jasak, A. Jemcov, Z. Tukovic, *et al.*, "Openfoam: A c++ library for complex physics simulations," in *International workshop on coupled methods in numerical dynamics*, vol. 1000, pp. 1–20, 2007. 58, 59, 75

[66] P.-E. Angeli, U. Bieder, and G. Fauchet, "Overview of the triocfd code: Main features, vetv procedures and typical applications to nuclear engineering," in *NURETH 16-16th International Topical Meeting on Nuclear Reactor Thermalhydraulics*, 2015. 58

[67] F. Archambeau, N. Méchitoua, and M. Sakiz, "Code saturne: A finite volume code for the computation of turbulent incompressible flows-industrial applications," *International Journal on Finite Volumes*, vol. 1, no. 1, 2004. 58

[68] J. Levesque, "The code aster: a product for mechanical engineers; le code aster: un produit pour les mecaniciens des structures," *Epure*, 1998. 58

[69] T. Helfer, B. Michel, J.-M. Proix, M. Salvo, J. Sercombe, and M. Casella, "Introducing the open-source mfront code generator: Application to mechanical behaviours and material knowledge management within the PLEIADES fuel element modelling platform," vol. 70, no. 5, pp. 994–1023. 58

[70] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey, "`libMesh`: A C++ Library for Parallel Adaptive Mesh Refinement/Coarsening Simulations," *Engineering with Computers*, vol. 22, no. 3–4, pp. 237–254, 2006. https://doi.org/10.1007/s00366-006-0049-3. 58

[71] W. Bangerth, R. Hartmann, and G. Kanschat, "deal. ii—a general-purpose object-oriented finite element library," *ACM Transactions on Mathematical Software (TOMS)*, vol. 33, no. 4, pp. 24–es, 2007. 58

[72] M. Alnæs, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M. E. Rognes, and G. Wells, "The fenics project version 1.5," *Archive of numerical software*, vol. 3, no. 100, 2015. 58

[73] R. Da Vià, *Development of a computational platform for the simulation of low Prandtl number turbulent flows*. PhD thesis, University of Bologna, 2019. 58

[74] "Numeric platform." https://github.com/FemusPlatform/NumericPlatform. 58, 59

[75] G. Barbi, A. Cervone, F. Giangolini, S. Manservisi, and L. Sirotti, "Numerical coupling between a fem code and the fvm code openfoam using the med library," *Applied Sciences*, vol. 14, no. 9, p. 3744, 2024. 59

[76] G. Barbi, G. Bornia, D. Cerroni, A. Cervone, A. Chierici, L. Chirco, R. Da Vià, V. Giovacchini, S. Manservisi, and R. Scardovelli, "Femus-platform: A numerical platform for multiscale and multiphysics code coupling," in *9th International Conference on Computational Methods*

for Coupled Problems in Science and Engineering, COUPLED PROBLEMS 2021, pp. 1–12, International Center for Numerical Methods in Engineering, 2021. 59, 72

[77] "Salome." https://www.salome-platform.org/?page_id=23, 2023. 59, 61

[78] J. Ahrens, B. Geveci, C. Law, C. Hansen, and C. Johnson, "36-paraview: An end-user tool for large-data visualization," *The visualization handbook*, vol. 717, pp. 50038–1, 2005. 59

[79] W. Gropp, E. Lusk, N. Doss, and A. Skjellum, "A high-performance, portable implementation of the mpi message passing interface standard," *Parallel computing*, vol. 22, no. 6, pp. 789–828, 1996. 59

[80] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: portable parallel programming with the message-passing interface*, vol. 1. MIT press, 1999. 59

[81] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, "Open MPI: Goals, concept, and design of a next generation MPI implementation," in *Proceedings, 11th European PVM/MPI Users' Group Meeting*, (Budapest, Hungary), pp. 97–104, September 2004. 59

[82] S. Balay, S. Abhyankar, M. F. Adams, S. Benson, J. Brown, P. Brune, K. Buschelman, E. M. Constantinescu, L. Dalcin, A. Dener, V. Eijkhout, J. Faibussowitsch, W. D. Gropp, V. Hapla, T. Isaac, P. Jolivet, D. Karpeev, D. Kaushik, M. G. Knepley, F. Kong, S. Kruger, D. A. May, L. C. McInnes, R. T. Mills, L. Mitchell, T. Munson, J. E. Roman, K. Rupp, P. Sanan, J. Sarich, B. F. Smith, S. Zampini, H. Zhang, H. Zhang, and J. Zhang, "PETSc Web page." https://petsc.org/, 2024. 60

[83] S. Bna, I. Spisso, M. Olesen, and G. Rossi, "Petsc4foam: A library to plug-in petsc into the openfoam framework," *PRACE White paper*, 2020. 60

[84] A. Ribes and C. Caremoli, "Salome platform component model for numerical simulation," in *31st annual international computer software*

and applications conference (COMPSAC 2007), vol. 2, pp. 553–564, IEEE, 2007. 61

[85] A. Chierici, V. Giovacchini, and S. Manservisi, "Analysis and numerical results for boundary optimal control problems applied to turbulent buoyant flows.," *International Journal of Numerical Analysis & Modeling*, vol. 19, 2022. 72

[86] R. Da Vià, V. Giovacchini, and S. Manservisi, "A logarithmic turbulent heat transfer model in applications with liquid metals for pr = 0.01–0.025," *Applied Sciences*, vol. 10, no. 12, 2020. 72

[87] L. Chirco, *On the optimal control of steady fluid structure interaction systems*. PhD thesis, University of Bologna, 2020. 72

[88] D. Cerroni, *Multiscale multiphysics coupling on a finite element platform*. PhD thesis, University of Bologna, 2016. 72

[89] R. D. Vià, *Development of a computational platform for the simulation of low Prandtl number turbulent flows*. PhD thesis, alma, Aprile 2019. 74, 75

[90] I. Farajpour and S. Atamturktur, "Optimization-based strong coupling procedure for partitioned analysis," *Journal of computing in civil engineering*, vol. 26, no. 5, pp. 648–660, 2012. 79

[91] H. G. Matthies, R. Niekamp, and J. Steindorf, "Algorithms for strong coupling procedures," *Computer methods in applied mechanics and engineering*, vol. 195, no. 17-18, pp. 2028–2049, 2006. 81

[92] L. De Giorgi, V. Bertola, and E. Cafaro, "Thermal convection in double glazed windows with structured gap," *Energy and Buildings*, vol. 43, no. 8, pp. 2034–2038, 2011. 87

[93] M. Bitaab, R. Hosseini Abardeh, and S. Movahhed, "Experimental and numerical study of energy loss through double-glazed windows," *Heat and Mass Transfer*, vol. 56, pp. 727–747, 2020. 87

[94] H. Singh and P. C. Eames, "A review of natural convective heat transfer correlations in rectangular cross-section cavities and their potential applications to compound parabolic concentrating (cpc) solar collector

cavities," *Applied Thermal Engineering*, vol. 31, no. 14-15, pp. 2186–2196, 2011. 87

[95] M. Dahmani and F. Z. Ferahta, "Enhancing convective heat loss reduction in flat-plate solar collectors by optimal integration of transparent partitions in the air gap," *Heat Transfer Research*, vol. 55, no. 15, 2024. 87

[96] N. B. Balam, T. Alam, A. Gupta, and P. Blecich, "Higher order accurate transient numerical model to evaluate the natural convection heat transfer in flat plate solar collector," *Processes*, vol. 9, no. 9, p. 1508, 2021. 87

[97] M. Ciofalo and T. Karayiannis, "Natural convection heat transfer in a partially—or completely—partitioned vertical rectangular enclosure," *International journal of Heat and Mass transfer*, vol. 34, no. 1, pp. 167–179, 1991. 88

[98] G. de Vahl Davis, "Natural convection of air in a square cavity: a benchmark numerical solution," *International Journal for numerical methods in fluids*, vol. 3, no. 3, pp. 249–264, 1983. 88, 94, 96, 97, 99, 100, 105, 199

[99] N. Massarotti, P. Nithiarasu, and O. Zienkiewicz, "Characteristic-based-split (cbs) algorithm for incompressible flow problems with heat transfer," *International Journal of Numerical Methods for Heat & Fluid Flow*, vol. 8, no. 8, pp. 969–990, 1998. 88, 94, 99, 100, 103, 104, 105, 193, 199

[100] M. Manzari, O. Hassan, K. Morgan, and N. Weatherill, "Turbulent flow computations on 3d unstructured grids," *Finite elements in analysis and design*, vol. 30, no. 4, pp. 353–363, 1998. 88

[101] M. Manzari, "An explicit finite element algorithm for convection heat transfer problems," *International Journal of Numerical Methods for Heat & Fluid Flow*, vol. 9, no. 8, pp. 860–877, 1999. 88, 94, 99, 100, 103, 104, 105, 193

[102] D. A. Mayne, A. S. Usmani, and M. Crapper, "h-adaptive finite element solution of high rayleigh number thermally driven cavity prob-

lem," *International Journal of Numerical Methods for Heat & Fluid Flow*, vol. 10, no. 6, pp. 598–615, 2000. 88, 94, 99, 100, 105

[103] W. D.C., P. B.S.V., and W. G.W., "A new benchmark quality solution for the buoyancy-driven cavity by discrete singular convolution," *Numerical Heat Transfer: Part B: Fundamentals*, vol. 40, no. 3, pp. 199–228, 2001. 88, 94, 96, 99, 100, 101, 102, 103, 104, 105, 192, 193

[104] S. Grossmann and D. Lohse, "Scaling in thermal convection: a unifying theory," *Journal of Fluid Mechanics*, vol. 407, pp. 27–56, 2000. 104

[105] T. R. Kennelly and S. Dabiri, "Natural convection in a differentially heated cavity-effect of prandtl number," *Available at SSRN 4542982*. 105, 107, 108, 128, 193

[106] T. Perelman, "On conjugated problems of heat transfer," *International Journal of Heat and Mass Transfer*, vol. 3, no. 4, pp. 293–303, 1961. 112

[107] A. S. Dorfman, *Conjugate problems in convective heat transfer*. CRC Press, 2009. 112

[108] A. Gil, M. Medrano, I. Martorell, A. Lázaro, P. Dolado, B. Zalba, and L. F. Cabeza, "State of the art on high temperature thermal energy storage for power generation. part 1—concepts, materials and modellization," *Renewable and sustainable energy reviews*, vol. 14, no. 1, pp. 31–55, 2010. 113

[109] Y. Tian and C.-Y. Zhao, "A review of solar collectors and thermal energy storage in solar thermal applications," *Applied energy*, vol. 104, pp. 538–553, 2013. 113

[110] K. R. Kumar and K. Reddy, "Thermal analysis of solar parabolic trough with porous disc receiver," *Applied energy*, vol. 86, no. 9, pp. 1804–1812, 2009. 113

[111] J. A. Ackermann, L.-E. Ong, and S. C. Lau, "Conjugate heat transfer in solar collector panels with internal longitudinal corrugated fins—part i: overall results," *Forschung Im Ingenieurwesen*, vol. 61, no. 4, pp. 84–92, 1995. 113

[112] J. A. Ackermann, L.-E. Ong, and S. C. Lau, "Conjugate heat transfer in solar collector panels with internal longitudinal corrugated fins-part ii: Local results," *Forschung im Ingenieurwesen*, vol. 61, no. 6, pp. 172–179, 1995. 113

[113] H. M. Regue, B. Bouali, T. Benchatti, and A. Benchatti, "Numerical simulation of conjugate heat transfer in a ptc with secondary reflector," *International Journal of Heat and Technology*, vol. 38, no. 1, pp. 9–16, 2020. 113

[114] F. Nees and Y. Pai, "Conjugate heat transfer analysis of the transient thermal discharge of a metallic latent heat storage system," in *Journal of Physics: Conference Series*, vol. 2766, p. 012212, IOP Publishing, 2024. 113

[115] J. Solano, F. Roig, F. Illán, R. Herrero-Martín, J. Pérez-García, and A. García, "Conjugate heat transfer in a solar-driven enhanced thermal energy storage system using pcm," in *de Proceedings of the 4th World Congress on Mechanical, Chemical, and Material Engineering (MCM'18), Madrid*, 2018. 113

[116] Y. Ito, N. Inokura, and T. Nagasaki, "Conjugate heat transfer in air-to-refrigerant airfoil heat exchangers," *Journal of heat transfer*, vol. 136, no. 8, p. 081703, 2014. 113

[117] P. Renze and K. Akermann, "Simulation of conjugate heat transfer in thermal processes with open source cfd," *ChemEngineering*, vol. 3, no. 2, p. 59, 2019. 113

[118] E. Greiciunas, D. Borman, J. Summers, and S. J. Smith, "A multi-scale conjugate heat transfer modelling approach for corrugated heat exchangers," *International Journal of Heat and Mass Transfer*, vol. 139, pp. 928–937, 2019. 113

[119] N. Lebaal, A. SettaR, S. Roth, and S. Gomes, "Conjugate heat transfer analysis within in lattice-filled heat exchanger for additive manufacturing," *Mechanics of Advanced Materials and Structures*, vol. 29, no. 10, pp. 1361–1369, 2022. 113

[120] H. Ahmed, H. Sadat, and S. Nasrazadani, "High-fidelity conjugate heat transfer simulation of micro-channel heat exchanger," *Journal of Advanced Research in Fluid Mechanics and Thermal Sciences*, vol. 106, no. 1, pp. 165–181, 2023. 113

[121] F. Espinosa, R. Avila, J. Cervantes, and F. Solorio, "Numerical simulation of simultaneous freezing–melting problems with natural convection," *Nuclear engineering and design*, vol. 232, no. 2, pp. 145–155, 2004. 113

[122] J.-W. Park, J.-h. Bae, and H.-J. Song, "Conjugate heat transfer analysis for in-vessel retention with external reactor vessel cooling," *Annals of Nuclear Energy*, vol. 88, pp. 57–67, 2016. 113

[123] A. Timperi, "Conjugate heat transfer les of thermal mixing in a t-junction," *Nuclear Engineering and Design*, vol. 273, pp. 483–496, 2014. 113

[124] S. Chen, Y. Yan, and W. Gong, "A simple lattice boltzmann model for conjugate heat transfer research," *International Journal of Heat and Mass Transfer*, vol. 107, pp. 862–870, 2017. 113

[125] X. Chen and P. Han, "A note on the solution of conjugate heat transfer problems using simple-like algorithms," *International Journal of Heat and Fluid Flow*, vol. 21, no. 4, pp. 463–467, 2000. 113

[126] N. Sato, S. Takeuchi, T. Kajishima, M. Inagaki, and N. Horinouchi, "A consistent direct discretization scheme on cartesian grids for convective and conjugate heat transfer," *Journal of Computational Physics*, vol. 321, pp. 76–104, 2016. 113

[127] S. V. Patankar, *Numerical heat transfer and fluid flow*. Hemisphere Publishing Corporation, 1980. 113

[128] T. Basak, R. Anandalakshmi, and A. K. Singh, "Heatline analysis on thermal management with conjugate natural convection in a square cavity," *Chemical engineering science*, vol. 93, pp. 67–90, 2013. 114, 118, 125, 126, 194, 199

[129] B. John, P. Senthilkumar, and S. Sadasivan, "Applied and theoretical aspects of conjugate heat transfer analysis: A review," *Archives of Computational Methods in Engineering*, vol. 26, pp. 475–489, 2019. 115

[130] R. Henkes, F. Van Der Vlugt, and C. Hoogendoorn, "Natural-convection flow in a square cavity calculated with low-reynolds-number turbulence models," *International Journal of Heat and Mass Transfer*, vol. 34, no. 2, pp. 377–388, 1991. 128

[131] R. Henkes and C. Hoogendoorn, "Scaling of the turbulent natural convection flow in a heated square cavity," 1994. 128

[132] F. P. Incropera, D. P. DeWitt, T. L. Bergman, A. S. Lavine, *et al.*, *Fundamentals of heat and mass transfer*, vol. 6. Wiley New York, 1996. 129

[133] S. Paolucci and D. R. Chenoweth, "Transition to chaos in a differentially heated vertical cavity," *Journal of Fluid Mechanics*, vol. 201, pp. 379–410, 1989. 129

[134] J. Lage and A. Bejan, "The ra-pr domain of laminar natural convection in an enclosure heated from the side," *Numerical heat transfer*, vol. 19, no. 1, pp. 21–41, 1991. 129, 194

[135] L. Zwirner, A. Tilgner, and O. Shishkina, "Elliptical instability and multiple-roll flow modes of the large-scale circulation in confined turbulent rayleigh-bénard convection," *Physical Review Letters*, vol. 125, no. 5, p. 054502, 2020. 130

[136] T. Zürner, F. Schindler, T. Vogt, S. Eckert, and J. Schumacher, "Combined measurement of velocity and temperature in liquid metal convection," *Journal of Fluid Mechanics*, vol. 876, pp. 1108–1128, 2019. 130

[137] T. Vogt, S. Horn, A. M. Grannan, and J. M. Aurnou, "Jump rope vortex in liquid metal convection," *Proceedings of the National Academy of Sciences*, vol. 115, no. 50, pp. 12674–12679, 2018. 130

[138] S. Bawazeer, A. Mohamad, and P. Oclon, "Natural convection in a differentially heated enclosure filled with low prandtl number fluids with modified lattice boltzmann method," *International Journal of Heat and Mass Transfer*, vol. 143, p. 118562, 2019. 130, 131, 134, 194

[139] F. Wolff, C. Beckermann, and R. Viskanta, "Natural convection of liquid metals in vertical cavities," *Experimental Thermal and Fluid Science*, vol. 1, no. 1, pp. 83–91, 1988. 130, 131, 132, 134, 139, 145, 149, 194, 195

[140] R. Viskanta, D. Kim, and C. Gau, "Three-dimensional natural convection heat transfer of a liquid metal in a cavity," *International journal of heat and mass transfer*, vol. 29, no. 3, pp. 475–485, 1986. 130, 132, 134, 145, 149, 195

[141] A. Mohamad and R. Viskanta, "Transient natural convection of low-prandtl-number fluids in a differentially heated cavity," *International Journal for numerical methods in fluids*, vol. 13, no. 1, pp. 61–81, 1991. 130, 131, 132, 134, 145, 149, 194, 195

[142] A. Mohamad and R. Viskanta, "Modeling of turbulent buoyant flow and heat transfer in liquid metals," *International journal of heat and mass transfer*, vol. 36, no. 11, pp. 2815–2826, 1993. 130, 131, 132, 194

[143] J. Oder, M. Alsailani, L. Koloszar, W. Munters, D. Laboureur, and J. Pacio, "Direct numerical simulation of flow in a confined differentially heated cavity at low prandtl numbers," pp. 1206–1218, 01 2023. 130, 131, 132, 134, 142, 150, 155, 194

[144] A. Mohamad and R. Viskanta, "Application of low reynolds number k-ε turbulence model to buoyant and mixed flows in a shallow cavity," *Fundamentals of Mixed Convection*, vol. 223, pp. 43–54, 1992. 132, 134, 139, 145, 146, 149, 195

[145] V. Giovacchini, *Development of a numerical platform for the modeling and optimal control of liquid metal flows.* PhD thesis, alma, Luglio 2022. 142, 170, 171, 172, 196

[146] A. Ying, M. Abdou, C. Wong, S. Malang, N. Morley, M. Sawan, B. Merrill, D. K. Sze, R. Kurtz, S. Willms, *et al.*, "An overview of us iter test blanket module program," *Fusion engineering and design*, vol. 81, no. 1-7, pp. 433–441, 2006. 158

[147] L. Boccaccini, J.-F. Salavy, R. Lässer, A. L. Puma, R. Meyder, H. Neuberger, Y. Poitevin, and G. Rampal, "The european test blanket mod-

ule systems: Design and integration in iter," *Fusion engineering and design*, vol. 81, no. 1-7, pp. 407–414, 2006. 158

[148] P. Calderoni, A. Aiello, B. Ghidersa, Y. Poitevin, J. Pacheco, *et al.*, "Current design of the european tbm systems and implications on demo breeding blanket," *Fusion Engineering and Design*, vol. 109, pp. 1326–1330, 2016. 158

[149] C. Ciurluini, V. Narcisi, A. Tincani, C. O. Ferrer, and F. Giannetti, "Conceptual design overview of the iter wcll water cooling system and supporting thermal-hydraulic analysis," *Fusion Engineering and Design*, vol. 171, p. 112598, 2021. 158

[150] D. Martelli, A. Venturini, and M. Utili, "Literature review of lead-lithium thermophysical properties," *Fusion Engineering and Design*, vol. 138, pp. 183–195, 2019. 160

[151] K. Mergia and N. Boukos, "Structural, thermal, electrical and magnetic properties of eurofer 97 steel," *Journal of Nuclear Materials*, vol. 373, no. 1-3, pp. 1–8, 2008. 160

[152] C. A. Sleicher Jr and M. Tribus, "Heat transfer in a pipe with turbulent flow and arbitrary wall-temperature distribution," *Transactions of the American Society of Mechanical Engineers*, vol. 79, no. 4, pp. 789–797, 1957. 162

[153] V. Gnielinski, "New equations for heat and mass transfer in turbulent pipe and channel flow," *International chemical engineering*, vol. 16, no. 2, pp. 359–367, 1976. 163

[154] G. K. El Khoury, P. Schlatter, A. Noorani, P. F. Fischer, G. Brethouwer, and A. V. Johansson, "Direct numerical simulation of turbulent pipe flow at moderately high reynolds numbers," *Flow, turbulence and combustion*, vol. 91, pp. 475–495, 2013. 169, 171, 196

[155] S. Pirozzoli, J. Romero, M. Fatica, R. Verzicco, and P. Orlandi, "One-point statistics for turbulent pipe flow up to," *Journal of fluid mechanics*, vol. 926, p. A28, 2021. 169

[156] X. Wu and P. Moin, "A direct numerical simulation study on the mean velocity characteristics in turbulent pipe flow," *Journal of Fluid Mechanics*, vol. 608, pp. 81–112, 2008. 169

[157] L. Redjem-Saad, M. Ould-Rouiss, and G. Lauriat, "Direct numerical simulation of turbulent heat transfer in pipe flows: Effect of prandtl number," *International Journal of Heat and Fluid Flow*, vol. 28, no. 5, pp. 847–861, 2007. 169

Appendix A

Developed Routines

In this appendix, we describe the main routines implemented during the project. The aim is to offer a clearer and more detailed view of the algorithms' internal logic, complementing the developed C++ classes of the coupling application described in Chapter 2. Each routine is presented in a structured format to highlight key computational steps and decision-making processes.

A.1 MED Class Routines

This section describes the main methods implemented in the C++ class, named `MEDclass`, of the developed coupling application. The first routine generates the MED mesh starting from the connectivity and coordinate vectors provided by the coupled codes (e.g., FEMuS and OpenFOAM). The algorithm of the routine `create_mesh()` is shown in Algorithm 5. The function creates an instance of an unstructured mesh, named `_mesh`, and allocates sufficient memory to store a number of cells equal to *nel*, using the MED routine `mesh.allocateCells(nel)`. Each cell is then added to the mesh by specifying the cell type (e.g., `QUAD4`, `HEXA8`, `TRI3` etc.), the number of DOFs per cell, and its connectivity through the `_mesh.insertNextCell(type, dof, conn)` routine. The MED method `_mesh.finishInsertingCells()` marks the completion of the cell insertion process, once the mesh is completed. A

Algorithm 5 Create mesh

1: **function** CREATE_MESH(conn, coord, type, dimension)
2: Set dimension and elements type of the interface mesh
3: Create mesh object of type `MEDCoupling::MEDCouplingUMesh`
4: Allocate memory for the *nel* number of cells
5: **while** $i \neq nel$ **do**
6: Set cell connectivity
7: Insert i-th cell
8: **end while**
9: Close memory
10: Create `tmp` array of type `MEDCoupling::DataArrayDouble`
11: Allocate memory for $ndof \times dimension$ coordinates
12: Copy coordinates into `tmp`
13: Use `tmp` to set coordinates into mesh object
14: **end function**

temporary MED array, `tmp`, is then created. The function `tmp.alloc(ndof, dimension)` allocates the memory needed to store the coordinate values, with a total size corresponding to the number of nodes times the mesh dimensionality. The standard copy method is then used to transfer the values from the `coord` vector provided by the numerical code into the `tmp` array. To add the coordinates to the mesh, the MED routines `_mesh.setCoords(tmp)` and `_mesh.zipCoords()` are employed.

Algorithm 6 Initialization of node fields

1: **function** INIT_MED_FIELD_ON_NODES(mesh, name, dim)
2: Set field type to `MEDCoupling::ON_NODES`
3: Create field object of type `MEDCoupling::MEDCouplingFieldDouble`
4: Assign mesh to the field object
5: Set name of the field
6: SET_FIELD(field,dim)
7: **end function**

The second functionality, outlined in Algorithms 6 and 7, is the initialization of the fields used in the coupling application. The routine begins by selecting the discretization type of the fields, determining whether they are defined at the mesh nodes or are cell-wise fields. The field object (`_field`) is then instantiated and assigned to the mesh using the MED routine

Algorithm 7 Initialization of cell-wise fields

```
1: function INIT_MED_FIELD_ON_NODES(mesh, name, dim)
2:     Set field type to MEDCoupling::ON_CELLS
3:     Create field object of type MEDCoupling::MEDCouplingFieldDouble
4:     Assign mesh to the field object
5:     Set name of the field
6:     SET_FIELD(field,dim)
7: end function
```

`_field.setMesh(_mesh)`. The routine `_field.setName(name)` is used to set up the characteristic name of the field. Then, the algorithm calls the method `set_field(_field, dim)`, shown in Algorithm 8, to initialize the field object. The initialization, performed at the beginning of the solution process, when *time* is zero, involves allocating the memory needed to temporarily store the field values. The `_array_tmp` array must be sized as `ndof` times the field's dimension, depending on whether the field is scalar, vector, or tensor. The `_array_tmp` array is filled with the initial values using the copy function and then assigned to the field object through the `_field.setArray(_array_tmp)` routine. Then, `_field.checkConsistencyLight()` and `_field.setTime(0)` are invoked.

Algorithm 8 Set field

```
1: function SET_FIELD(field, dim)
2:     if time == 0 then
3:         Allocate memory of MEDCoupling::DataArrayDouble _array_tmp
4:         Copy initial values to _array_tmp
5:         Set _array_tmp to field
6:         Set time = 0
7:     else
8:         Fill _array_tmp with updated values
9:         Set _array_tmp to field
10:        Set time to time
11:    end if
12: end function
```

The same routine shown in Algorithm 8 for setting the field is called whenever the solution needs to be updated during execution. The inverse routine, `get_field()`, described in Algorithm 9, is used to retrieve the field

Algorithm 9 Get field

```
1: function GET_FIELD(field, _array_tmp)
2:     Get field updated values
3:     Set values to _array_tmp
4: end function
```

values from the `_field` object and store them in the `_array_tmp` array.

Algorithm 10 Get remapper

```
1: function GET_REMAPPER(method, src_mesh, trgt_mesh)
2:     Create P object of type MEDCoupling::MEDCouplingRemapper
3:     Set interpolation method
4:     Compute P for src_mesh and trgt_mesh
5:     return P
6: end function
```

Algorithm 11 Interpolate field

```
1: function INTERPOLATE_FIELD(remapper, src_field, trgt_field, nature)
2:     Set src_field nature
3:     Compute trgt_field from src_field
4: end function
```

Beyond mesh and field creation, a main functionality of the MED class is the interpolation process, which is managed by the routines `get_remapper()` and `interpolate_field()`. The `get_remapper()` routine, detailed in Algorithm 10, creates the `_remapper` object, which represents the projection matrix $\mathbf{P}$ used to map fields from one mesh to another. The interpolation method is specified using `_remapper.setIntersectionType(method)`, a REMAPPER class routine. The remapper object is then computed by calling the `_remapper.prepare(src_mesh, trgt_mesh, type)` method, which calculates the projection matrix $\mathbf{P}$ based on the source mesh, the target mesh, and the selected interpolation type (e.g., $P_0 - P_0$, $P_0 - P_1$, etc.).

Finally, the `interpolate_field()` routine performs the following matrix operation

$$\Phi_t = \mathbf{P}\Phi_s, , \tag{A.1}$$

to compute the target field starting from the source field. The nature of the source field (e.g., `IntensiveMaximum`, `ExtensiveConservation`) is specified

using the `_field.setNature(nature)` function. This definition assigns a physical meaning to the field and influences the selection of the appropriate interpolation strategy. Then, the `_remapper.transferField(src_field)` routine is called to generate the `trgt_field`.

A.2 FEMuS Interface Class Routines

This section describes the main functionality of the wrapper class between FEMuS code and the MED framework.

Algorithm 12 Interface initialization

```
1: function INIT_INTERFACE(name)
2:     Create interface object
3:     SET_MESH_CONNECTIVITY(interface)
4:     SET_MESH_COORDINATES(interface)
5:     Set interface parameters (i.e. _n_nodes, mesh_name)
6:     Set useful structures (i.e. _map_med2mg, _map_mg2med, _indices)
7: end function
```

The design of the class is centered around the creation of an `interface` object. Specifically, the `init_interface()` function groups all the functionalities required to gather the necessary information for constructing the interface mesh from the FEMuS code. This method is outlined in Algorithm

Algorithm 13 Set mesh connectivity

```
1: function SET_MESH_CONNECTIVITY(interface)
2:     Get total number of element nel
3:     for i < nel do
4:         if quad then
5:             Get i-th quadratic cell connectivity
6:         else if lin then
7:             Get i-th linear cell connectivity
8:         end if
9:         Insert connectivity to interface._conn
10:    end for
11: end function
```

12. It instantiates an object of the `interface_femus` structure and begins

initializing its data members. The routine invokes the retrieval of mesh connectivity.

The `set_mesh_connectivity()` function, detailed in Algorithm 13, is used to populate the `interface._conn` data member with the connectivity information for each of the *nel* interface elements. The connectivity is computed based on the requirements of the MED mesh format and supports both linear and quadratic elements. It is worth noting that the logic for extracting the i-th cell connectivity from the FEMuS solver mesh has been specialized to differentiate between volume and boundary interfaces. Algorithm 14 shows the implementation of the function `set_mesh_coordinates()` invoked after the connectivity retrieval. For each of the *ndof* nodes of the mesh (volume or boundary), the `interface._coords` vector is initialized with the corresponding j-th coordinate in each spatial dimension of the mesh. The `init_interface()` function ends by initializing key data structures required for the interface's operation. It includes the creation of mapping structures that associate the nodes of the FEMuS mesh with those of the MED interface. `femusMED` and `MEDclass` use these maps to convert indices from the internal solver mesh to the MED interface and vice-versa. Additionally, the function sets up an indices vector that holds the global node indices to efficiently retrieve the solver's solution during later stages of computation.

Algorithm 14 Set mesh coordinates

```
1: function SET_MESH_COORDINATES(interface)
2:     Get total number of element ndof
3:     for i < ndof do
4:         for j < dimension do
5:             Get i, j-th node coordinate
6:             Insert coordinate to interface._coords
7:         end for
8:     end for
9: end function
```

Another core functionality of the FEMuS interface class is retrieving solution values from and assigning external fields to the internal solution structures of FEMuS. Algorithm 15 describes the procedure implemented in the `get_field_from_femus()` function, which is responsible for extracting the solver solution from the FEMuS code and assigning it to the interface structure. The routine begins by retrieving the PETSc solution object, referred

Algorithm 15 Get field from FEMuS

1: **function** GET_FIELD_FROM_FEMUS(interface)
2: Get PETSc solution object `numvec`
3: `numvec` → get(`interface._indices, interface._field_val`)
4: **end function**

to as `numvec`, which contains the numerical solution data. It then employs the `get()` method of the PETSc library to extract the values corresponding to the global indices specified in `interface._indices`. The resulting field values are stored in the `interface._field_val` container, making them accessible for transfer into MED-compatible data structures.

Algorithm 16 Set field to FEMuS

1: **function** SET_FIELD_TO_FEMUS(interface,dim)
2: Get PETSc object `numvec`
3: **for** $i < ndof \times dim$ **do**
4: `numvec` ← set(`interface._indices, interface._field_val`)
5: **end for**
6: **end function**

The inverse operation, setting an external field into FEMuS, is described in Algorithm 16. It uses the `set()` method provided by PETSc to write values directly into the FEMuS solution field.

A.3 OpenFOAM Interface Class Routines

This section outlines the main functionalities of the wrapper class that connects the OpenFOAM code with the MED framework. The base class, `foamMED`, implements all the methods required to create the interface between OpenFOAM and MED. Derived classes enable the supervisor of the coupled application to execute specific OpenFOAM solvers. As regards the base class, the `init_interface()` function is described. It follows the same structure presented in Algorithm 12 and is used to populate the `interface_of` structures.

The `set_mesh_connectivity()` function is presented in Algorithm 17. In OpenFOAM, this function retrieves the list of cells (`cell`) from the mesh and assigns their connectivity to the `interface._conn` data member. A

specialized version of this routine is also implemented for boundary mesh interfaces, as shown in Algorithm 18. The boundary information is extracted by accessing the mesh `patch` using the identification number associated with the boundary. The logic of the `set_mesh_coordinates()` function follows

Algorithm 17 Set mesh connectivity

```
1: function SET_MESH_CONNECTIVITY(interface)
2:     Get mesh cells cell
3:     for i in cell do
4:         Get i−th cell connectivity
5:         Insert connectivity to interface._conn
6:     end for
7: end function
```

Algorithm 18 Set boundary mesh connectivity

```
1: function SET_MESH_CONNECTIVITY(interface,patch)
2:     Get patch identification number
3:     Get patch mesh
4:     Get patch cell list cell
5:     for i in cell do
6:         Get i−th cell connectivity
7:         Insert connectivity to interface._conn
8:     end for
9: end function
```

the structure outlined in Algorithm 14.

In Algorithm 19, we have detailed the procedure used by the function named `get_field_from_of()`, which extracts solver results from OpenFOAM and assigns them to the interface structure. The solution is retrieved by accessing the field through the name of the corresponding OpenFOAM field (e.g., `U`, `T`, `epsilon`, etc.). In OpenFOAM, fields are associated with the mesh object and are accessed through the mesh entities. The OpenFOAM field structures can vary depending on whether the field belongs to the volume mesh or the boundary mesh (e.g., `volScalarField`, `volVectorField`, `surfaceScalarField` etc.). The retrieved solution is then copied in the `interface._field_val` container. The `set_field_from_of()` function follows a logic similar to that of Algorithm 19; however, instead of retrieving

Algorithm 19 Get field from OpenFOAM

```
function GET_FIELD_FROM_OF(interface, name, patch)
    if patch ≠ null then
        Get boundary mesh object
    else
        Get volume mesh object
    end if
    for i in fields do
        if i == name then
            Get field values from mesh object
            Fill interface._field_val
        end if
    end for
end function
```

the solution and storing it in `interface._field_val`, it performs the reverse operation by assigning the values from `interface._field_val` to the corresponding OpenFOAM field.

A.3.1 Derived Classes

For clarity, the full C++ implementation of the derived classes is provided below. The following wrapper can be used by interested users to access internal structures of OpenFOAM v11. As regards the `foamSingleProblem` class, the implementation of the methods are detailed in the following program listing:

```
void foamSingleProblem::init(int argc, char *argv[]) {

  Foam::argList::addOption("solver", "name", "Solver name");
  Foam::argList args_base(argc, argv);

  //Set path to OpenFOAM folder
  auto opts = Foam::HashTable<Foam::string>();
  opts.set("case", path_problem);
  Foam::argList args(args_base, opts);
  if (!args.checkRootCase()) {
    Foam::FatalError.exit();
```

```
12    }
13
14    // Create time
15    _runTime.reset(new Foam::Time(Foam::Time::controlDictName, args));
16
17    // Read the solverName in controlDict
18    solverName = _runTime->controlDict().lookupOrDefault("solver",
      ↪  Foam::word::null);
19    // Optionally reset the solver name from the command-line
20    args.optionReadIfPresent("solver", solverName);
21
22    // Check that the solverName has been set
23    if (solverName == Foam::word::null) {
24      args.printUsage();
25      FatalErrorIn(args.executable())
26          << "solver not specified in the controlDict or on the
            ↪  command-line"
27          << exit(Foam::FatalError);
28    } else {
29      // Load the solver library
30      Foam::solver::load(solverName);
31    }
32  }
33
34  void foamSingleProblem::init_mesh() {
35    // Create the default single region mesh
36    mesh_ = new
      ↪  Foam::fvMesh(Foam::IOobject(Foam::fvMesh::defaultRegion,
      ↪  _runTime->name(), *_runTime, Foam::IOobject::MUST_READ));
37  }
38
39  void foamSingleProblem::init_solver() {
40    // Instantiate the selected solver
41    _solver.reset(Foam::solver::New(solverName, *mesh_).ptr());
42  }
43
44  void foamSingleProblem::init_pimple_control() {
45    // Create the outer PIMPLE loop and control structure
46    _pimple.reset(new
      ↪  Foam::pimpleSingleRegionControl(_solver->pimple));
```

```
47
48    // Set the initial time-step
49    setDeltaT(*_runTime, *_solver);
50  }
51
52  bool foamSingleProblem::run() { return _pimple->run(*_runTime); }
53
54  void foamSingleProblem::pre_solve() { _solver->preSolve(); }
55
56  void foamSingleProblem::post_solve() { _solver->postSolve(); }
57
58  void foamSingleProblem::write() {  _runTime->write(); }
59
60  void foamSingleProblem::setup_dt() {
61    // Adjust the time-step according to the solver maxDeltaT
62    adjustDeltaT(*_runTime, *_solver);
63    (*_runTime)++;
64  }
65
66  void foamSingleProblem::solve() {
67    // PIMPLE corrector loop
68    while (_pimple->loop()) {
69      _solver->moveMesh();
70      _solver->fvModels().correct();
71      _solver->prePredictor();
72      _solver->momentumPredictor();
73      _solver->thermophysicalPredictor();
74      _solver->pressureCorrector();
75      _solver->postCorrector();
76    }
77  }
```

The `foamMultiProblem` class methods are detailed in the following listing:

```
1   void foamMultiProblem::init(int argc, char * argv[])
2   {
3     Foam::argList::addOption("solver", "name", "Solver name");
4     Foam::argList args_base(argc, argv);
5
6     //Set path to OpenFOAM folder
```

```
7     auto opts = Foam::HashTable<Foam::string>();
8     opts.set("case", std::string{path_problem});
9     Foam::argList args(args_base, opts);
10    if (!args.checkRootCase()) {
11      Foam::FatalError.exit();
12    }
13
14    //Create time
15    _runTime.reset(new Foam::Time{Foam::Time::controlDictName, args});
16
17    // Create the region meshes and solvers
18    _solver.reset(new Foam::regionSolvers{*_runTime});
19
20    // Create the outer PIMPLE loop and control structure
21    _pimple.reset(new Foam::pimpleMultiRegionControl{*_runTime,
      ↪  *_solver});
22  }
23
24  void foamMultiProblem::pre_solve()
25  {
26    forAll(*_solver, i){  (*_solver)[i].preSolve(); }
27
28    _solver->setGlobalPrefix();
29  }
30
31  void foamMultiProblem::solve()
32  {
33    // Multi-region PIMPLE corrector loop
34    while (_pimple->loop())
35    {
36        forAll(*_solver, i){  (*_solver)[i].moveMesh(); }
37        forAll(*_solver, i){  (*_solver)[i].fvModels().correct(); }
38        forAll(*_solver, i){  (*_solver)[i]..prePredictor(); }
39        forAll(*_solver, i){  (*_solver)[i].momentumPredictor(); }
40        while (_pimple->correctEnergy())
41        {
42            forAll(*_solver, i)
43            {
44              (*_solver)[i].thermophysicalPredictor();
45            }
```

```
46        }
47        forAll(*_solver, i){  (*_solver)[i].pressureCorrector(); }
48        forAll(*_solver, i){  (*_solver)[i].postCorrector(); }
49    }
50  }
51
52  void foamMultiProblem::post_solve()
53  {
54    forAll(*_solver, i){  (*_solver)[i].postSolve();}
55    _solver->setGlobalPrefix();
56  }
57
58  bool foamMultiProblem::run(){  return _pimple->run(*_runTime); }
59
60  void foamMultiProblem::write(){  _runTime->write(); }
61
62  void foamMultiProblem::setup_dt(){ setDeltaT(*_runTime, *_solver); }
63
64  void foamMultiProblem::adjust_dt()
65  {
66    // Adjust the time-step according to the solver maxDeltaT
67    adjustDeltaT(*_runTime, *_solver);
68    (*_runTime)++;
69  }
70
71  void foamMultiProblem::set_mesh(Foam::word region_name)
72  {
73    mesh = _solver->get_mesh(region_name);
74  }
```

Appendix B

Configuration Parameters for OpenFOAM Simulations

This appendix provides an overview of the OpenFOAM configuration used for the simulations presented in this thesis. It includes the key numerical settings such as discretization schemes, solver tolerances, and relaxation factors, as defined in the respective configuration files (e.g., `fvSchemes`, `fvSolution`, and `controlDict`). The numerical parameters used in the OpenFOAM simulations are reported in Table B.2 for the Differentially Heated Cavity configuration, and in Table B.3 for the heat exchanger case discussed in Chapter 4. Table B.1 lists the abbreviations used for the discretization schemes.

Discretization Scheme	Abbreviation
`Gauss linear`	GL
`Gauss linearUpwind`	GLU
`Gauss linearUpwind limited`	GLU limited
`Gauss limitedLinear`	GLL
`Gauss linear corrected/uncorrected`	GL corrected/uncorrected

Table B.1: Discretization scheme abbreviations.

Differentially Heated Cavity				
Simulation	Laminar		Turbulent	
`controlDict`				
Solver name	`fluid`		`fluid`	
`deltaT`	0.01		0.005	
`fvSchemes`				
Time derivative scheme	`Euler`		`backward`	
Gradient scheme	GL		GL	
Div scheme `U`	GLU		GLU limited	
Div scheme `e` and `K`	GL		GLL	
Div scheme turbulence	-		GLL	
Laplacian scheme	GL corrected		GL corrected	
`fvSolution`				
Solver	Type	Tolerance	type	tol
`p`	`GAMG`	10^{-9}	`GAMG`	10^{-8}
`rho`	`diagonal`	-	`diagonal`	-
Other variables	`PBiCGStab`	10^{-9}	`PBiCGStab`	10^{-8}
`nOuterCorrectors`	0		1	
`nCorrectors`	2		1	
Field Relaxation factor	1.0		0.9	
Equation Relaxation factor	1.0		0.7	

Table B.2: Configuration parameters for laminar and turbulent OpenFOAM simulations of the DHC problem.

Heat Exchanger Simulations				
Region	PbLi		Pipe + Fins	
`controlDict`				
Solver name	`fluid`		`solid`	
`deltaT`	0.01		0.1	
`fvSchemes`				
Time derivative scheme	`backward`		`Euler`	
Gradient scheme	GL		GL	
Div scheme `U`	GLU limited		-	
Div scheme `e` and `K`	GLL		-	
Div scheme turbulence	GLL		-	
Laplacian scheme	GL corrected		GL uncorrected	
`fvSolution`				
Solver	Type	Tolerance	type	tol
`p`	`GAMG`	10^{-6}	-	-
`rho`	`diagonal`	-	-	-
Other variables	`smoothSolver`	10^{-7}	`PCG`	10^{-6}
`nOuterCorrectors`	1		-	
`nCorrectors`	1		-	
Field Relaxation factor	0.7		1.0	
Equation Relaxation factor	0.7		1.0	

Table B.3: Configuration parameters for fluid and solid OpenFOAM simulations of the PbLi-air heat exchanger.