Universitat Politècnica de Catalunya

Doctoral Thesis

A high-performance computing coupling tool for partitioned multi-physics applications

J. Miguel Zavala-Aké

Advisors:
Dr. Mariano Vázquez
Dr. Daniel Mira

Doctoral Program in Aerospace Science and Technology

Barcelona, Spain
July 2018

Abstract

The simulation and modelling of complex applications involving the interaction of processes governed by different physical principles is addressed in this thesis. The interaction of a fluid with a deformable body, or the exchange of thermal energy between fluid and solid are examples of these multi-physics applications. In these two cases, the modelling strategy proposed here combines the solution of separated physical systems to account for the interactions taking place through the entire domain. As a consequence, the simulation process resulting from the use of separated systems considers independent codes to find the solution of each system, while the entire system is reconstructed through an iterative approach combining these solutions.

One of the main advantages of this partitioned approach is that each parallel code can use the most appropriate model and algorithm which allow achieving an accurate solution for the complete physical system. Nevertheless, several challenges must be considered when using this approach. For instance, from a physical point of view, the most of variables involved in the modelling of a multi-physical application must be continuous across the entire domain. From a computational point of view, efficient data transference between parallel codes is required to model the physical interactions taking place through the entire system. In addition, the simulation of multi-physics applications must be robust and maintain scalability not only for each parallel code, but also for the coupling problem.

The present work describes the development, validation and use of a high performance computing coupling tool designed for solving efficiently partitioned multi-physics applications. The emphasis has been placed to the development of strategies to make efficient use of large-scale computing architectures, but always keeping the robustness and accuracy of the solutions. The coupling tool developed controls the data transference between the parallel codes establishing peer-to-peer communication layouts between the processors, the dynamic localization of regions where physical interactions take place, and the possible interpolations required between the different meshes composing large-scale multi-physics application. In this work, these features are applied to solve two multi-physics applications: contact of deformable bodies, and conjugate heat transfer.

The contact problem involves the interaction of two or more solids which could deform. The state of this system is determined by the fact that its interactions are limited to the exchange of momentum through its closed surfaces. In this work, a parallel algorithm to deal with this problem is described. Firstly, the continuity of the variables involved in the coupling problem is ensured using a domain decomposition method, i.e., solving iteratively pairs of independent problems (partitions) where boundary conditions are modified. Broadly, Dirichlet-like boundary conditions are used on a given partition, while, Neumann condition are applied on the other. The regions of the surface for each body where the contact takes place are identified using the localization process implemented in the coupling tool. The results show that the parallel algorithm used here for the solution of contact problems agrees well with those achieved by the elastic contact theory as well as those obtained by commercial codes.

The conjugate heat transfer problem referes to the thermal interaction between a fluid and a solid. The state of this system requires to determine the temperature and heat flux distribution through its fluid-solid interface. In this case, the coupled process is similar to the contact problem. An iterative process based on the Dirichlet-Neumann coupling algorithm is also used to enforce the exchange of thermal energy at the fluid-solid interface. The results show the capability of the framework developed in this thesis to deal with practical engineering applications. Finally, a method to deal with the disparity between the temporal scales involved in fluid-solid thermal coupling encountered in practical applications is also proposed.

In order to demonstrate the capability of the coupling tool to deal with large-scale applications, a parallel performance study of the partitioned approach is developed in this thesis. The study leads to a load balance strategy that allows estimating the optimal performance of a parallel multi-physics application. In general, these expressions can be used to calculate analytically the maximum efficiency, and scalability achievable by a multi-physics application. Like in the case of parallel performance analysis conducted for a single parallel code, the analytical estimations presented here can be compared with the results achieved by the performance analysis of general multi-physics problem. As a result, the behaviour of this application is described and analysed in order to identify and overcome the issues introduced by parallel codes interacting through a coupling tool. The parallel performance analysis of a conjugate heat transfer problem shows that the optimal efficiency of this application is well represented by the expressions derived in this study and therefore, under some assumptions, it can be used to assign the most appropriate distribution of processors to guarantee an optimal performance.

Contents

1

Introduction

1.1 Motivation

A relevant issue in the Exascale age is the modelling of *multi-physics applications* where a physical phenomenon is addressed by combining separate physical systems [1]. Applications as climate modelling [2], fluid-structure interaction [3], or conjugate heat transfer [4] are examples of these type of problems. In general, the solution of all of them depend on the physical interactions taking place in the entire system. For instance, in a convective heat transfer problem as the cooling of a turbine blade, a fluid interacts with a solid through a common interface in which thermal energy is exchanged [5]. In this case, an accurate solution of this problem requires to determine the temperature and heat flux distribution through the common interface between them.

Partitioned approach In general, two main approaches can be used to model multi-physics applications: monolithic and partitioned [6]. A *monolithic approach* makes use of an unique solver code to simulate a multi-physics system. A *partitioned approach* is based on splitting the entire computational domain into independent regions (partitions) each of them governed by a particular physical principle. In this approach, the solution of the entire domain can be reconstructed from each partition through a *coupling algorithm*. This algorithm firstly assumes that an appropriate numerical method is used to solve individually each partition. Then, the physical interactions between pair of partitions are modelled by an appropriate iterative procedure.

Coupling tool The responsible for accomplishing most of the operations related to this algorithm is a *coupling tool*. Lets establish a coupling tool as a software which has been specially designed to solve a multi-physics problem by combining the single-physics present in each partition. Particularly, a coupling tool deals with the *parallel coupling problem.* It has been defined as *the transmission and transformation of the various distributed data between the component models comprising a parallel coupled system* [7]. Transmission usually refers to move the necessary data from one partition to another. Transformation consists principally of interpolations between partitions.

A great deal of effort has been made to develop parallel coupling tools specialized on the solution of partitioned multi-physics application. In this sense, each of them have developed its own application programming interface (API), which usually allows defining different *coupling schemes* along with appropriate methods to deal with the parallel coupling problem. Coupling tools should satisfy other important characteristics such as robustness, efficiency, and scalability [6]. Examples of these developments are MCT (Model Coupling Toolkit) [7], CHIMPS (Coupler for High-Performance Integrated Multi-Physics Simulations) [8] MpCCI (Mesh-based parallel Code Coupling Interface) [9], OpenPALM (Projet d'Assimilation par Logiciel Multi-methodes) [10,11], LIME (Lightweight Integrating Multi-physics Environment) [12], DTK (Data Transfer Kit) [13], and CTL (Component Template Library) [14]. Of particular interest are those recently developed and that furthermore have shown to be appropriate to solve large scale multi-physical problems. Such is the case of libraries as CWIPI (Coupling With Interpolation Parallel Interface) [15, 16], preCICE (Precise Code Interaction Coupling Environment) [17], and MUI (Multiscale Universal Interface) [18].

Challenges In this work, the emphasis is on those challenges related to the coupling problem. For instance, most of the physical variables involved in a multi-physical application should be continuous through the entire domain. Then, an important challenge when a partitioned approach is used to solve a multi-physical system is related to how to achieve the continuity through the coupling interface. Another challenge is related to a suitable use of the computational resources. In this sense, the reduction of the number of coupling iterations is an important area of research, as well as the convergence and stability of different coupling approaches [19]. Regarding the operations related to the coupling procedure, the time spent on localization and data transference (data mapping and data communication) can produce major disadvantages. This can take place when the sizes of the meshes given to each partition are significantly different. In this case, the localization time could increase due to the differences on the number of vertices forming each partition. A last example is the workload related to the coupling. Generally, in order to use a large-scale computer in the most efficient manner, each partition is divided so that the set of processors perform the same amount of calculations. Although it is possible to fulfil this requirement for *each* partition, usually it requires additional considerations when the entire coupled system is considered. For details related to another challenges see [1, 6, 20].

1.2 Scope of the thesis

The aim of this thesis is to describe the development, validation and use of a *high-performance computing coupling tool* that allows the solution of *partitioned multi-physical simulations.* The emphasis is on the efficient use of large-scale computers, but always considering the robustness and accuracy of the solutions. In this sense, this thesis can be divided into three main parts. In the fist part, the computational development of the coupling tool PLE++ (Parallel Location and Exchange++) is presented along with a mathematical background related to the parallel coupling problem. Then, two multi-physics applications are addressed: contact between deformable bodies, and conjugate heat transfer. Finally, in the last part, a performance analysis for multi-physics applications is introduced.

The development of this thesis has been part of three projects: PRACE Second Implementation Phase (PRACE-2IP) [21], Coupled Parallel Simulation of Gas Turbines (COPA-GT) [22], and Structural HEalth Monitoring, Manufacturing and Repair Technologies for Life Management Of Composite Fuselage (SHERLOC) [23]. The PRACE-2IP project was focused on the re-design and refactoring of a number of codes for scientific numerical applications. In order to effectively run on new generation of supercomputing architectures. The COPA-GT project was a Marie Curie Action designed to train young fellows in Europe in Gas Turbines design using High Performance Computing (HPC). The objective of the SHERLOC project is to combine advanced Structural Health Monitoring (SHM) and smart repair technologies with a probabilistic design philosophy. A common interest in these projects has been the capabilities shown by the HPC-code Alya. It is a finite element code developed by the Barcelona Supercomputing Center (BSC) which has already proved to be highly scalable in advanced HPC facilities [24]. In this sense, the possibility of coupling Alya with others parallel codes appears to be very attractive. The results achieved during the PRACE-2IP project allowed to develop an interface for the coupling of Alya with Code_Saturne and Syrthes [25, 26]. This first stage was done by modifying the internal coupling tool of Code_Saturne named PLE (Parallel Location and Exchange). During COPA-GT project, PLE evolves into an independent library named PLE++, which has the capability of coupling parallel codes written in Fortran, C/C++, or Python.

Results achieved during PRACE-2IP and COPA-GT projects allow dealing with multi-physics applications as: systems involving heterogeneous computing [27], conjugate heat transfer problems [28–30], and contact between deformable bodies [31]. The first case considers the use of PLE++ for the solution of large-scale Eulerian-Lagrangian systems. Regarding conjugate heat transfer problems, PLE++ has been used to study the effect of the heat transfer condition at the solid wall on a premixed impinging jet flame, and the heat loss of a confined turbulent jet flame in a labscale combustor. Finally, the PLE++ library is crucial in the SHERLOC project for predicting impact damage in composite panel structures. Details related to *how* these applications make use of the PLE++ library are given along this work.

Another important point related to the modelling of partitioned multi-physical applications is a suitable use of the computational resources. In this sense, two important contributions are given in this thesis. Firstly, one of the main issues in conjugate heat transfer problems is the time-disparity due to the time-scales characterising each partition. In this thesis, a strategy which allows reducing this disparity is presented. The results show that the computational time necessary to perform fluid-solid thermal coupling can be reduced by applying this approach. Another important contribution is related to the parallel performance analysis of partitioned multi-physics applications. This analysis allows the introduction of a load balance strategy that takes into account details related to how a multi-physics application solved by a partitioned coupling approach can be affected by the distribution of processors. These two strategies along with a suitable use of the HPC coupling tool PLE++ make possible the modelling of large-scale multi-physical application.

In the remaining of this work details related to the coupling library developing as part of this thesis are given. Before of that, a general introduction of the procedure followed throughout this work for the modelling of a partitioned multi-physical simulations is given below. Finally, the content of this thesis are summarised for each chapter.

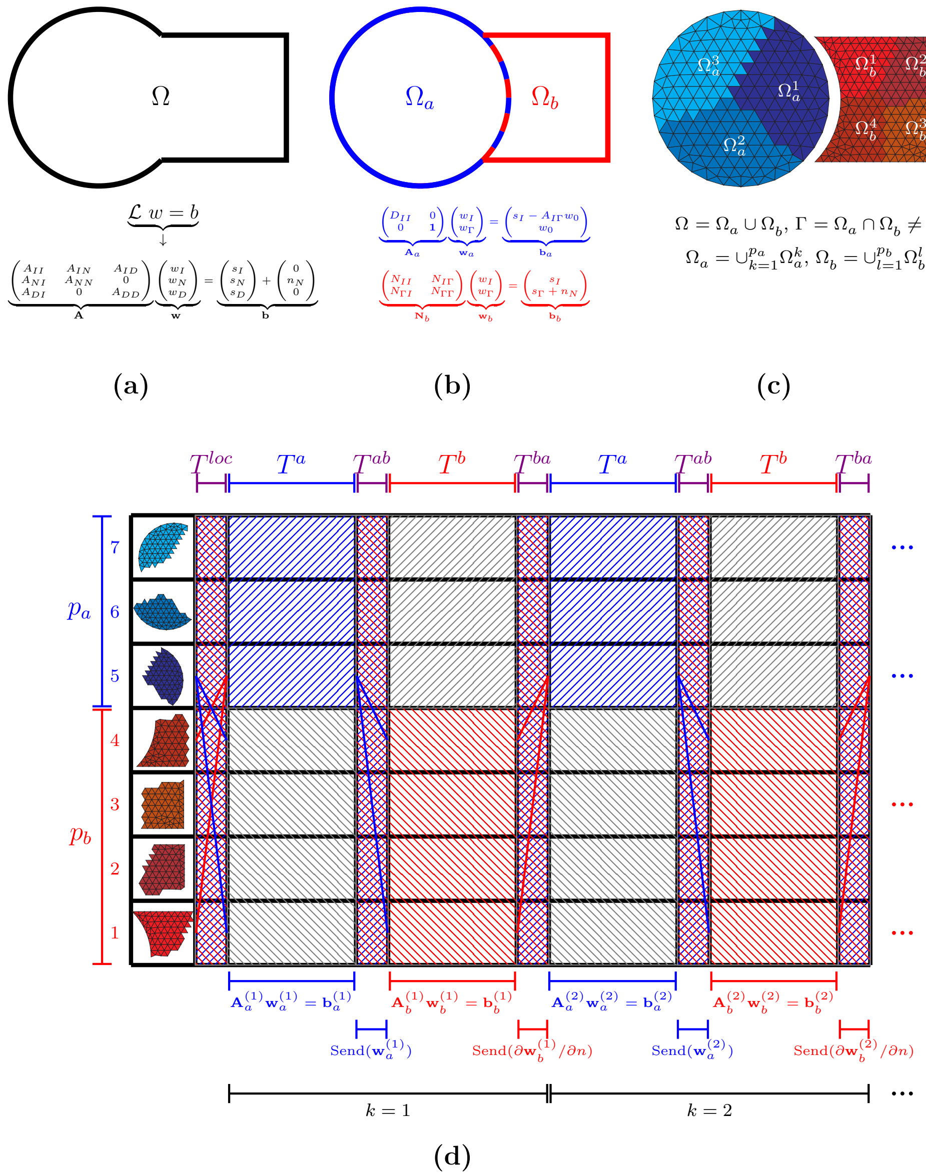
1.3 Partitioned approach

The entire process of a multi-physical simulation solved as a coupled system is summarized in Figure 1.1. Broadly speaking, each partition domain is solved independently, while the interaction between them are taken into account through the operations related to the parallel coupling problem. A brief description of the procedure followed is given below. Details are extensively covered in the following chapters.

Parallel coupling problem

Firstly, lets suppose that inside a given computational domain Ω, two different physical systems interact. Furthermore, lets suppose that each of these physical systems defines a computational region (partition), so that the interactions between this pair of partitions take place exclusively throughout the common interface Γ. The next step consists on defining a suitable numerical method for the solution of each partition. Under the assumptions mentioned above, it results on two system of algebraic equations $\mathbf{A}_a\mathbf{w}_a = \mathbf{b}_a$ and $\mathbf{A}_b\mathbf{w}_b = \mathbf{b}_b$, one for each partition Ω_a and Ω_b, respectively. These two systems replace an unique system of equations $\mathbf{A}\ \mathbf{w} = \mathbf{b}$, which would have been derived from discretizing the partial differential equation $\mathcal{L}\ w = b$ through the whole Ω. In order to use a large-scale computer in the most efficient manner, generally each partition is divided into as many sub-domains as processors has been assigned to a given partition. Thus, each partition Ω_α is constituted of a number p_α of sub-domains $\{\Omega_\alpha^1, \Omega_\alpha^2, ..., \Omega_\alpha^{p_\alpha}\}$. In turn, each of these sub-domains is associated with an unique processor. As a result, each of them allocates one of the corresponding values $\{\mathbf{w}_\alpha^1, \mathbf{w}_\alpha^2, ..., \mathbf{w}_\alpha^{p_\alpha}\}$ which result from resolving each algebraic equation $\mathbf{A}_\alpha \mathbf{w}_\alpha = \mathbf{b}_\alpha$ in parallel.

Once each partition has been prepared to be solved through a large-scale computer, the next step consists of accomplishing the operations related to the iterative coupling procedure. The main features to solve a partitioned multi-physical application with a coupling procedure are: I) coupling approach, II) data mapping, and III) data communication [17]. Additionally to this three features, a preprocessing is necessary before starting the coupling simulation. Figure 1.1(d) depicts the chronology in which the workflow of a *staggered coupling approach* is executed in parallel. In this case, the partitions Ω_a and Ω_b, in which the entire domain Ω has been split, are composed of three and four processors, $p_a = 3$ and $p_b = 4$, respectively. Before executing the iterative coupling approach, the pairs of processors related to the coupling must be identified. In order to do that, a *localization process* must be executed by all the processors $p = p_a + p_b$ during a time T^{loc}. As a result, it is possible to know what coupling takes place between sub-domains Ω_b^1 and Ω_b^4 (processors 1 and 4) of partition Ω_b, and the sub-domain Ω_a^1 (processor 5) of the partition Ω_a. Once this preprocessing has finished, the staggered coupling approach establishes that *algebraic equations must be solved one after the other.* In the case under study, for each k-iteration, the system $\mathbf{A}_a^{(k)} \mathbf{w}_a^{(k)} = \mathbf{b}_a^{(k)}$ is firstly executed during a time $T^{a(k)}$ using p_a processors. Now, data transference takes place before solving the system $\mathbf{A}_b^{(k)} \mathbf{w}_b^{(k)} = \mathbf{b}_b^{(k)}$ during time $T^{b(k)}$. The localization processes determines that this transference is performed from processor five to one, and from processor five to four. This transference is necessary due to the fact that all values $\mathbf{w}_a^{(k)} = \{\mathbf{w}_a^{1(k)}, \mathbf{w}_a^{2(k)}, \mathbf{w}_a^{3(k)}\}$ in Ω_a are updated, so that the values $\mathbf{w}_b^{(k)}$ through the boundary interface Γ in Ω_b must be also updated before solving its system of algebraic equations. The transference time $T^{ab(k)}$ includes the time spent on data transmission (sending and receiving data between processors), as well as data transformation (the data mapping in case of non-conforming meshes). Once this is done, the p_b processors associated to Ω_b can solve its corresponding system of algebraic equations enforcing the received information as boundary conditions on its coupling interface Γ. Each k-iteration finishes when the first partition executed (Ω_a) receives from the other partition (Ω_b) the new boundary conditions to be enforced through its coupling interface. Finally, the whole procedure is repeated as often as necessary.



Ω
Ω_a
Ω_b
$\mathcal{L}\, w = b$
$\mathbf{A}$ $\mathbf{w}$ $\mathbf{b}$
$\mathbf{A}_a$ $\mathbf{w}_a$ $\mathbf{b}_a$
$\mathbf{N}_b$ $\mathbf{w}_b$ $\mathbf{b}_b$
$\Omega = \Omega_a \cup \Omega_b,\ \Gamma = \Omega_a \cap \Omega_b \neq \emptyset$
$\Omega_a = \cup_{k=1}^{p_a} \Omega_a^k,\ \Omega_b = \cup_{l=1}^{p_b} \Omega_b^l$
(a)
(b)
(c)
T^{loc} T^a T^{ab} T^b T^{ba}
p_a
p_b
$\mathbf{A}_a^{(1)}\mathbf{w}_a^{(1)} = \mathbf{b}_a^{(1)}$
$\mathbf{A}_b^{(1)}\mathbf{w}_b^{(1)} = \mathbf{b}_b^{(1)}$
$\mathbf{A}_a^{(2)}\mathbf{w}_a^{(2)} = \mathbf{b}_a^{(2)}$
$\mathbf{A}_b^{(2)}\mathbf{w}_b^{(2)} = \mathbf{b}_b^{(2)}$
Send($\mathbf{w}_a^{(1)}$)
Send($\partial \mathbf{w}_b^{(1)}/\partial n$)
Send($\mathbf{w}_a^{(2)}$)
Send($\partial \mathbf{w}_b^{(2)}/\partial n$)
$k = 1$
$k = 2$
(d)

Figure 1.1: (a) Whole domain. (b) Partitioned domain. (c) Discretization. (d) Execution. The total number of processors $p = p_a + p_b$ assigned to the whole coupled system $\Omega = \Omega_a \cup \Omega_b$ are divided between two disjoint partitions Ω_a and Ω_b involved in the coupling. Each partition Ω_α is constituted of a number p_α of sub-domains $\{\Omega_\alpha^1, \Omega_\alpha^2, ..., \Omega_\alpha^{p_\alpha}\}$. The interface $\Gamma_{ab} = \Omega_a \cap\ \Omega_b$ defines the communication between the partitions. In this case, communication takes place between sub-domains Ω_b^1 and Ω_b^4 (processors 1 and 4) of partition b, and the sub-domain Ω_a^1 (processor 5) of the partition a. The data transfer is only performed between overlapping sub-domains, in two steps: interpolation and exchange. The interpolation maps property values $\mathbf{v}_\alpha^l$ on Ω_α^l to values $\mathbf{v}_\alpha^k$ on Ω_β^k through the coordinates $\mathbf{r}_\beta^k$.

1.4 Thesis outline

As stated above, this thesis is organized in three main parts. Chapters 2 and 3 address on the mathematical background and the computational development used in this thesis. Two multi-physics applications addressed through the PLE++ library are analysed in Chapters 4 and 5. Finally, in Chapter 6, a performance analysis for multi-physics applications is presented.

Chapter 2 The aim of this chapter is to introduce the mathematical nature of the *coupling approaches* used in this thesis for the solution of partitioned multi-physics applications. These approaches are based on *domain decomposition methods*: to divide a large problem into smaller problems forming partitions of the original domain.

Chapter 3 In this chapter, details related to the implementation of a *parallel coupling tool* that allows the solution of large-scale partitioned multi-physical simulations are given. The coupling tool developed here (the PLE++ library) allows dealing with the *parallel coupling problem* using a coupling approach to combine the solution achieved by the single-parallel solver present in each partition.

Chapter 4 This chapter addresses the development of a novel parallel algorithm to deal with the problem of *contact between deformable bodies*. This algorithm make uses of an iterative coupling approach to enforce the constrains arising from contact, while the PLE++ library is the responsible for identifies regions on the surface of each body where contact takes place.

Chapter 5 This chapter is focused on describing a methodolgy to address *conjugate heat transfer* problems using a partitioned approach. Here, an iterative process is used to enforce the effects related to the exchange of thermal energy through the fluid-solid interface. Additionally to this iterative process, a methodology to deal with the time-disparity arising form the difference in the temporal scales between fluids and solids is also presented.

Chapter 6 In this chapter, a *load balance strategy* to run partitioned multi-physics applications on extreme scale architectures with an optimal parallel performance is presented. This strategy provides a detailed analysis of the influence of the assignment of the processors to the partitions and provides an expression that relates the performance metrics of each partition with the overall parallel efficiency of the coupled simulation.

2

Approaches and algorithms

Divide and conquer

The aim of this chapter is to introduce the mathematical nature of the *coupling approaches* used in this thesis for the solution of partitioned multi-physics applications. These approaches are based on *domain decomposition methods*: to divide a large problem into smaller problems forming partitions of the original domain. In general, an important challenge is to ensure that the solution found individually for each partition converges to the one achieved when the whole system is solved by using an unique solver code.

Details related to the iterative coupling approaches used along this thesis are given as follows: A general introduction of domain decomposition for *non-overlapping partitions* is presented in next section. Iterative methods to deal with partitioned multi-physical systems are then introduced. In Section 2.2 an overview of muti-physical systems to be solved in the next chapters is given. Finally, remarks and conclusions are drawn in last Section.

2.1 Domain decomposition methods

Domain decomposition makes reference to *the splitting of a partial differential equation into coupled problems on smaller sub-domains forming a partition of the original domain* [32]. In general, two classes of domain decomposition methods can be considered: *Schwarz* and *Schur complement* [33]. Schwarz methods are the earliest domain decomposition method known [34]. They are generally applied when partitions have *overlapping* regions. In multi-physics, the Schwarz method can be used to solve the interactions of a fluid containing particles [35], as well as the solution of aero-acoustic problems where the domain is divided into two regions; an acoustic near field where a fluid flow simulation is considered, and a far field where the acoustic wave propagates [36]. It is worth noting that, in general, as the overlapping region is reduced, the rate of convergence of the Schwarz method decreases [33]. If the interaction between the partitions is reduced to the common surface defined between them (in a 3D case), a Schur complement method can be used instead. This is the case of fluid-solid interaction, conjugate heat transfer, or contact of deformable bodies. In these three cases, the interaction between partitions are given exclusively through a common boundary. Due to the fact that this thesis is focused on the contact of deformable bodies, and the conjugate heat transfer, the rest of this chapter emphasises some features of domain decomposition methods that can be applied for non-overlapping (disjoint) partitions.

2.1.1 Iterative methods for non-overlapping partitions

As stated above, domain decomposition methods considering non-overlapping partitions are known as Schur complement methods (also referred as Steklov-Poincaré [37] or Substructuring methods [33]). The Schur methods are based on the physical principle of continuity: the *transmission conditions* [37]. These conditions establish that a given variable and its corresponding flux must match across the coupling interface, i.e., they must be continuous throughout the surface defined between each pair of partitions. In order to achieve this matching, an iterative process must be performed. Without limiting the generality, details are given through an example. Firstly, the case studied is defined. Then, transmission conditions for this case are described. An iterative method which allows achieving the continuity of the transmission conditions is finally described.

Case study

An *advection-diffusion system* is firstly considered. Based on the above, its interaction with a convection dominated system is analysed in the next section. The interaction between Navier-Stokes and Euler equations is an example of this situation. Details can be found in [37, 38].

Definition Lets consider the *non-conservative* form of a *diffusion-transport-reaction* equation

$$\begin{cases} \mathcal{L}u \equiv -\nabla\cdot(a(x)\nabla u) + \mathbf{b}(x)\cdot\nabla u + c(x)u &= f(x) \qquad \text{in } \Omega\subset\mathcal{R}^2 \\ u &= \varphi \qquad\quad \text{on } \Gamma_D\subset\partial\Omega \\ \mathbf{n}\cdot a\nabla u &= \psi \qquad\quad \text{on } \Gamma_N\subset\partial\Omega \end{cases} \tag{2.1.1}$$

in the domain Ω with boundary $\partial\Omega$, enforced to the Dirichlet boundary condition $u=\varphi$ on $\Gamma_D\subset\partial\Omega$, and Neumann boundary condition $\mathbf{n}\cdot a\nabla u=\psi$ on $\Gamma_N\subset\partial\Omega$, see Figure 2.1. Where $\Gamma_D\cup\Gamma_N=\partial\Omega$, $\Gamma_D\cap\Gamma_N=\emptyset$, and unit outgoing normal $\mathbf{n}$. a, $\mathbf{b}$, and c are three *smooth* assigned functions with $0\leq a_0\leq a$, $c\geq 0$, and

$-\frac{1}{2}\nabla \cdot \mathbf{b} + c \geq \beta > 0 \; \forall x \in \Omega$. Note that Dirichlet boundary conditions can be used to prescribe either a velocity profile on the *inflow* boundary

$$\Gamma_{in} \;\; \equiv \;\; \{\Gamma_D \subset \partial\Omega : \mathbf{n} \cdot \mathbf{b} < 0\} \tag{2.1.2}$$

or forcing the fluid to stick to a wall (*no-slip* boundary conditions). On the other hand, *free outflow* boundary conditions Γ_{out} are prescribed where the *force* per unit *area* at the Neumann boundary is zero, i.e., where $\psi = 0$. Equation 2.1.1 can be seen as the scalar version of a stationary linearized problem in fluid dynamics where a represents the kinematic viscosity, $\mathbf{b}$ the transport field, and f the source term [39].

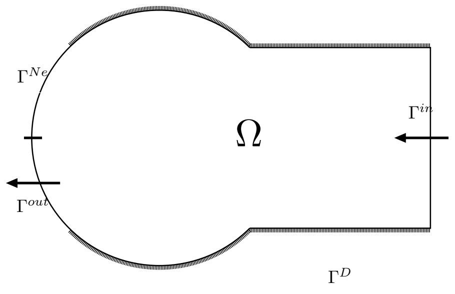


Figure 2.1: Geometry of the domain Ω. The arrows denote the *local* directions of the transport field $\mathbf{b}$.

Transmission conditions In order to understand the role of the transmission conditions some details should be considered. The decomposition of a domain $\Omega = \cup_{i=1}^{p} \Omega_i$, in p non-overlapping partitions Ω_i, involves the existence of an interface Γ_{ij} separating two regions Ω_i and Ω_j. As consequence, *it is necessary to identify suitable boundary conditions throughout the interface*, the so-called transmission conditions. These boundary conditions should ensure (whenever possible) that the coupled problem is well posed, i.e., the solution w_i in each partition Ω_i exists, be unique, and it depends continuously on the data [37]. Furthermore, such boundary conditions must ensure that the solutions w_i converges to the real solution u of the problem 2.1.1, i.e., $w_i - u|_{\Omega_i} \to 0$. Thus, the application of the domain decomposition of Equation 2.1.1 leads to the solutions w_a and w_b of the partitions Ω_a and Ω_b along with the suitable transmission conditions across the interface Γ_{ab} between Ω_a and Ω_b.

Weak form As shown below, the *weak form* of Equation 2.1.1 can be used to clarify the origin of the transmission conditions. Under the given mixed boundary conditions, the weak form of the diffusion-transport-reaction equation reads:

$$\text{find } u \in \mathcal{U}\text{:} \quad \mathcal{A}(u, \nu) \;\; = \;\; F(\nu) \quad \forall \nu \in \mathcal{V} \tag{2.1.3}$$

where

$$\begin{cases} \mathcal{A}(u,\nu) & \equiv & \int_\Omega \left(a\nabla u\cdot\nabla\nu + \nu\mathbf{b}\cdot\nabla u + c\;u\;\nu\right)d\Omega \\ F(\nu) & \equiv & \int_\Omega f\nu\;d\Omega + \int_{\Gamma_N}\psi\nu\;d\Gamma \\ \mathcal{U} & \equiv & \{u\in H^1(\Omega):\;\; u=\varphi \text{ on } \Gamma_D\;\} \\ \mathcal{V} & \equiv & \{\nu\in H^1(\Omega):\;\; \nu=0 \text{ on } \Gamma_D\;\} \end{cases}$$

Here, Equation 2.1.1, has been multiplied by a *test function* ν and integrated on the domain Ω. In order to have *well defined* integrals, each function u and ν along with its respective derivative must be *square integrable*, i.e., $\mathcal{U}$ and $\mathcal{V}$ represent *Sobolev spaces* of order one[1] . The difference between both spaces is related to the condition to be fulfilled when Dirichlet boundary conditions are imposed. In the first case, $\mathcal{U}$, the values of u are enforced to be equal to φ. On the other hand, $\mathcal{V}$, the test functions ν must vanish [39, 40]. Equation 2.1.4 shows the deduction of the weak form 2.1.3 in detail. The Neumman boundary Γ_N comes naturally as a consequence of the integration by parts of the diffusive therm $\nabla\cdot(a\nabla u)$. However, the test functions ν vanish where Dirichlet boundary conditions are applied, and then its contribution on the boundary Γ_D is small and can be neglected. This fact is important in order to understand the origin of the transmission conditions.

$$\begin{aligned} &\int_\Omega \Big(\nabla\cdot(-a\nabla u) + \mathbf{b}\cdot\nabla u + cu\Big)\nu\;d\Omega \\ &= \int_\Omega \Big(a\nabla u\cdot\nabla\nu - \nabla\cdot(\nu a\nabla u) + \nu\mathbf{b}\cdot\nabla u + cu\nu\Big)d\Omega \\ &= \int_\Omega \Big(a\nabla u\cdot\nabla\nu + \nu\mathbf{b}\cdot\nabla u + cu\nu\Big)d\Omega + \int_{\partial\Omega} -\nu a\nabla u\cdot\mathbf{n}\;d\Gamma \\ &= \int_\Omega \Big(a\nabla u\cdot\nabla\nu + \nu\mathbf{b}\cdot\nabla u + cu\nu\Big)d\Omega + \int_{\Gamma_N}\nu\psi\;d\Gamma + \cancel{\int_{\Gamma_D} -\nu a\nabla u\cdot\mathbf{n}\;d\Gamma}. \end{aligned} \tag{2.1.4}$$

In this equation, the integral of the *flux* $-\nu a\nabla u\cdot\mathbf{n}$ throughout the boundary Γ is defined over Neumann Γ_N and Dirichlet Γ_D boundaries. It is worth noting that the integral throughout Γ_N is different from zero only where $\psi\neq 0$. On the other hand, the integral corresponding to Dirichlet boundary is removed due to the fact that, by construction, $\nu\equiv 0$ on Γ_D. Despite this, the integral of $-\nu a\nabla u\cdot\mathbf{n}$ throughout the interface Γ_{ij} between two partitions Ω_i and Ω_j of Ω can be *different* from zero. This is because ν is nullified in order to enforce the *physical* Dirichlet boundary conditions throughout Γ.

[1] A *Sobolev* space of order k on Ω is the space formed by the totality of functions of $L^2(\Omega)$ whose derivatives up to order k *belong* to $L^2(\Omega)$:

$$H^k(\Omega) \;=\; \{f\in L^2(\Omega): D^\alpha f\in L^2(\Omega)\quad \forall\alpha: |\alpha|\leq k\}$$

The space of square-integrable functions on Ω is defined as:

$$L^2(\Omega) \;=\; \{f:\Omega\rightarrow\Re \text{ such that } \int_\Omega (f)^2 d\Omega < +\infty\}$$

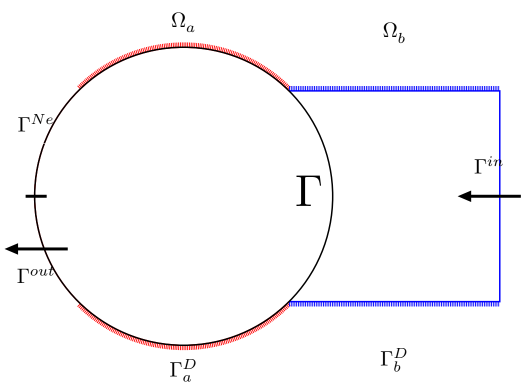


Figure 2.2: Geometry of the partitions Ω_a and Ω_b. The arrows denote the *local* directions of the transport field $\mathbf{b}$.

In summary, given a partition Ω_i, the integral of $-\nu a \nabla u \cdot \mathbf{n}$ can be: (1) zero on its physical Dirichlet boundary, (2) different from zero when either physical non-homogeneous Neumann boundary conditions are used, or throughout the interface Γ_{ij} with other partition Ω_j. Finally, these results are used to define the transmission conditions needed to solve a partitioned system.

Partitioned approach From the previous discussion, the weak form of Equation 2.1.1 shows that an additional term is introduced when the problem is decomposed into partitions. As shown below, this term allows coupling the partitions, so that the given problem can be solved.

The additivity of the volumetric and superficial integrals in 2.1.4 allows splitting them in as many partitions as necessary, so that

$$\begin{aligned}
&\int_\Omega \Big(a\nabla u \cdot \nabla \nu + \nu \mathbf{b} \cdot \nabla u + c u \nu \Big) d\Omega + \int_{\partial\Omega} -\nu a \nabla u \cdot \mathbf{n} \; d\Gamma \\
&= \sum_{i=1}^{p} \Big[\int_{\Omega_i} \Big(a\nabla u \cdot \nabla \nu + \nu \mathbf{b} \cdot \nabla u + c u \nu \Big) d\Omega + \int_{\partial\Omega_i} -\nu a \nabla u \cdot \mathbf{n} \; d\Gamma \\
&+ \int_{\Gamma_{ij}} -\nu a \nabla u \cdot \mathbf{n} \; d\Gamma \Big].
\end{aligned} \tag{2.1.5}$$

The transmission conditions that couple these p local problems are then defined by how the integral on Γ_{ij} are tested. Without limiting the generality, suppose that the domain Ω is decomposed into two non-overlapping partitions Ω_a and Ω_b separated by an interface Γ, thus

$$\begin{aligned}&\int_{\Omega}\Big(a\nabla u\cdot\nabla\nu+\nu\mathbf{b}\cdot\nabla u+cu\nu\Big)d\Omega+\int_{\partial\Omega}-\nu a\nabla u\cdot\mathbf{n}\,d\Gamma\\&=\int_{\Omega_a}\Big(a\nabla u\cdot\nabla\nu+\nu\mathbf{b}\cdot\nabla u+cu\nu\Big)d\Omega+\int_{\partial\Omega_a}-\nu a\nabla u\cdot\mathbf{n}\,d\Gamma\\&+\int_{\Omega_b}\Big(a\nabla u\cdot\nabla\nu+\nu\mathbf{b}\cdot\nabla u+cu\nu\Big)d\Omega+\int_{\partial\Omega_b}-\nu a\nabla u\cdot\mathbf{n}\,d\Gamma\\&+\int_{\Gamma_{ab}}-\nu a\nabla u\cdot\mathbf{n}\,d\Gamma+\int_{\Gamma_{ba}}-\nu a\nabla u\cdot\mathbf{n}\,d\Gamma\end{aligned}\tag{2.1.6}$$

where Γ_{ab} and Γ_{ba} stand for the contribution of *each* partition Ω_i to the weak form of Equation 2.1.1. Under the assumption that the partitions are solved *independently*, and considering that the boundary conditions must ensure that the whole coupled problem is well posed, then one alternative is the use of Dirichelet boundary conditions by an *unique* partition.

Suppose that φ_a is enforced as Dirichlet boundary condition on the interface Γ_{ab}. Then, due to the fact that $\nu\equiv 0$, the integral $\int_{\Gamma_{ab}}-\nu a\nabla u\cdot\mathbf{n}\,d\Gamma$ vanishes. On the other hand, in the partition Ω_b, Neumann boundary conditions $-a\nabla u\cdot\mathbf{n}|_{\Omega_b}\equiv\boldsymbol{\psi}_{ba}$ are necessary on Γ_{ba}. The value ψ_{ba} to be enforced on this boundary can be provided by the flux $\boldsymbol{\psi}_{ab}=-a\nabla u\cdot\mathbf{n}|_{\Omega_a}$ leaving (reaching) the partition Ω_a throughout the interface Γ_{ab}. Thus, the weak form in each partition reads: find $u\in\mathcal{U}$:

$$\int_{\Omega_a}\Big(a\nabla u\cdot\nabla\nu+\nu\mathbf{b}\cdot\nabla u+cu\nu-f\Big)d\Omega+\int_{\Gamma_{Na}}\boldsymbol{\psi}_a\cdot\mathbf{n}\,d\Gamma=0\tag{2.1.7}$$

for the Dirichlet partition, and

$$\int_{\Omega_b}\Big(a\nabla u\cdot\nabla\nu+\nu\mathbf{b}\cdot\nabla u+cu\nu-f\Big)d\Omega+\int_{\Gamma_{Nb}}\boldsymbol{\psi}_b\cdot\mathbf{n}\,d\Gamma+\int_{\Gamma}-\boldsymbol{\psi}_{ab}\cdot\mathbf{n}\,d\Gamma=0\tag{2.1.8}$$

for the Neumann partition. It is worth noting that, the negative value of the flux $\boldsymbol{\psi}_{ab}$ on Γ_{ab} enforced as Neumann boundary condition on Γ_{ba}, ensures that the global effect of the integral on Γ is zero. As a result, the weak form of the Equation 2.1.1 is recovered.

Iterative algorithm Given a domain Ω composed by two partitions Ω_a and Ω_b, where $\Omega=\Omega_a\cup\Omega_b$, the boundary value problem 2.1.1 can be solved by an iterative algorithm where the transmission conditions of Equations 2.1.7 and 2.1.8 can be used. An iterative algorithm is necessary because, clearly, the values of φ_a and $\boldsymbol{\psi}_{ab}$ to be imposed on Γ_{ab} and Γ_{ba}, respectively, are unknown. Thus, it is necessary to start imposing certain values $\varphi_a^{(0)}$ and $\boldsymbol{\psi}_{ab}^{(0)}$ and to use iterative methods to solve the interface problem. One of the most widely used iterative methods to solve these problems uses of a sequential execution order of the partition solvers [40]. Thus, in order to solve $\mathcal{L}u = f$, Equations 2.1.7 and 2.1.8 suggest that, firstly the partition Ω_a must be solved by imposing Dirichlet conditions on Γ_{ab}, and then enforcing Neumann conditions on the boundary Γ_{ba}, of the partition Ω_b.

Figure 2.3 depicts the details related to this algorithm. It starts by enforcing an initial guess $\varphi_a^{(0)}$ on the Dirichlet boundary for the solution of $\mathcal{L}w_1^{(k)} = f_1$, where $w_i^{(k)}$ stands for the solution of the k's iteration on the domain Ω_i. Now, the flux of $w_1^{(k)}$ is enforced through the interface Γ_{ba} of Ω_b. As result of $\mathcal{L}w_2^{(k)} = f_2$, the local solution $w_2^{(k)}$ matches the flux $\mathbf{n}_1 \cdot (a\nabla w_1^{(k)})$, but not necessarily the value of $w_1^{(k)}$, i.e., $w_2^{(k)} - w_1^{(k)} \neq 0$. Thus, in order to achieve the desired convergence, an under relaxation factor $0 < \theta \leq 1$ can be used such that $v_2^{(k+1)} = w_2^{(k)} + \theta(w_2^{(k)} - w_1^{(k)})$. This relation allows updating the value of $v_2^{(k)}$ in such a way that the difference $w_2^{(k)} - w_1^{(k)}$ decreases gradually with each iteration. Finally, when $v_2^{(k+1)} \approx w_2^{(k)}$, the k-iteration is dropped, and the algorithm finishes.

Lets $v_2^{(0)}$ denote a starting guess

1. For $k = 0, 1, ...,$ until convergence do:
2. Solve for $w_1^{(k)}$ as follows:

$$\begin{cases} \mathcal{L}w_1^{(k)} = f_1 & \text{in } \Omega_a \\ w_1^{(k)} = \varphi_a & \text{on } \Gamma_{D_a} \\ w_1^{(k)} = v_2^{(k)} & \text{on } \Gamma_{ab} \end{cases} \tag{2.1.9}$$

3. Solve for $w_2^{(k)}$ as follows:

$$\begin{cases} \mathcal{L}w_2^{(k)} = f_2 & \text{in } \Omega_b \\ w_2^{(k)} = \varphi_b & \text{on } \Gamma_{D_b} \\ \mathbf{n}_2 \cdot (a\nabla w_2^{(k)}) = \mathbf{n}_1 \cdot (a\nabla w_1^{(k)}) & \text{on } \Gamma_{ba} \end{cases} \tag{2.1.10}$$

4. Update:

$$v_2^{(k+1)} = w_1^{(k)} + \theta(w_2^{(k)} - w_1^{(k)}) \qquad \text{on } \Gamma \tag{2.1.11}$$

5. Endfor

Output: $(w_1^{(k)}, w_2^{(k)})$

Figure 2.3: Iterative Dirichlet-Neumann algorithm (DNA).

2.1.2 Partitioned multi-physics approach

As stated above, domain decomposition methods are based on decomposing a given problem into coupled smaller problems [32,39]. *Heterogeneous domain decomposition methods* generalizes this idea by assuming that *different* kind of problems can take place in each partition [40]. The interaction between fluids and solids is an example of multi-physics showing such heterogeneity.

Case Study

Definition The advection-diffusion system analysed in the previous section can be addressed by an heterogeneous domain decomposition method under the assumption that there exists a sub-region of the system completely governed by the convective term.

Figure 2.4 shows a situation in which a domain Ω is composed by a partition Ω_a governed by the the diffusion-transport-reaction equation (Equation 2.1.1), and another partition $\Omega_b = \Omega \backslash \Omega_a$ completely governed by the convective term $\mathbf{b} \cdot \nabla u$. Thus, if in Equation 2.1.1 the function a is re-defined as

$$a \equiv \begin{cases} \eta & \text{for } \Omega_a \subset \Omega \\ \epsilon & \text{for } \Omega_b \subset \Omega \end{cases} \tag{2.1.12}$$

where $|\nabla \cdot (\epsilon \nabla u)| << |\mathbf{b} \cdot \nabla u|$, then $\mathcal{L}u = f$ can be approximated as $\mathcal{L}_0 u = f$ in the region Ω_b, with $\mathcal{L}_0 u = \mathbf{b} \cdot \nabla u + cu$. For a system where convection is predominant over diffusion, the projection $\mathbf{n} \cdot \mathbf{b}$ of convective vector $\mathbf{b}$ on the outgoing normal vector $\mathbf{n}$ defines the portion of $\partial\Omega$ to be considered as inflow or outflow boundary [40]. Thus, $\partial\Omega$ can be split into four parts: Dirichlet Γ_D, Neumann Γ_N, inflow Γ^{in} and outflow Γ^{out}, where

$$\begin{aligned} \Gamma^{in} &\equiv \{\Gamma \subset \partial\Omega : \mathbf{n} \cdot \mathbf{b} < 0\} \\ \Gamma^{out} &\equiv \{\Gamma \subset \partial\Omega : \mathbf{n} \cdot \mathbf{b} \geq 0\}\,. \end{aligned} \tag{2.1.13}$$

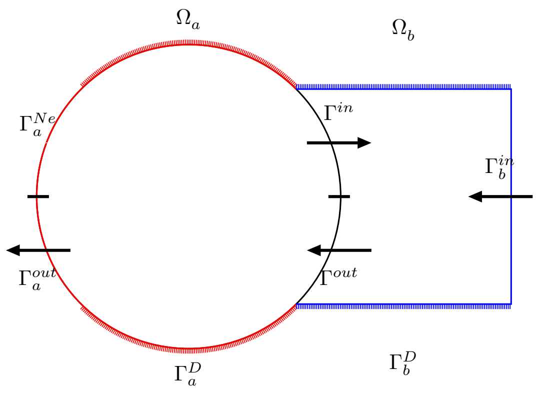


Figure 2.4: Geometry of the partitions Ω_a and Ω_b. The arrows denote the *local* directions of the transport field $\mathbf{b}$.

Lets $v_2^{(0)}$ denote a starting guess

1. For $k = 0, 1, ...,$ until convergence do:
2. Solve for $w_1^{(k)}$ as follows:

$$\begin{cases} \mathcal{L}_0 w_1^{(k)} &= f_1 & \text{in } \Omega_a \\ w_1^{(k)} &= \varphi_a & \text{on } \Gamma^{in} \\ w_1^{(k)} &= v_2^{(k)} & \text{on } \Gamma_{ab}^{in} \end{cases} \tag{2.1.14}$$

3. Solve for $w_2^{(k)}$ as follows:

$$\begin{cases} \mathcal{L} w_2^{(k)} &= f_2 & \text{in } \Omega_b \\ w_2^{(k)} &= \varphi_b & \text{on } \Gamma_{D_b} \\ \mathbf{n}_2 \cdot (a \nabla w_2^{(k)} - \mathbf{b} w_2^{(k)}) &= \mathbf{n}_1 \cdot \mathbf{b} w_1^{(k)} & \text{on } \Gamma_{ba}^{out} \\ \mathbf{n}_2 \cdot (a \nabla w_2^{(k)}) &= 0 & \text{on } \Gamma_{ba}^{in} \end{cases} \tag{2.1.15}$$

4. Update:

$$v_2^{(k)} = w_1^{(k)} + \theta(w_2^{(k)} - w_1^{(k)}) \qquad \text{on } \Gamma \tag{2.1.16}$$

5. Endfor

Output: $(w_1^{(k)}, w_2^{(k)})$

Figure 2.5: Iterative Robin-Neumann algorithm (RNA).

In the same way that $\partial\Omega$, it can be expected that the coupling interface Γ can be sub-divided into inflow and outflow parts, so that a change of the transmission conditions should be considered in order to ensure that the new coupled system is still well posed.

Iterative algorithm Due the presence of two different physical systems in Ω, a similar analysis to the iterative Dirichlet-Neumann algorithm must be performed. By following the procedure used to infer Equations 2.1.6 and 2.1.8, an iterative algorithm with new transmission conditions can be defined.

Figure 2.5 shows the resulting iterative algorithm. Firstly, the Dirichlet condition $w_1^{(k)} = v_2^{(k)}$ throughout Γ_{ab} is changed so it takes place exclusively through the inflow part of the boundary Γ_{ab}^{in}, i.e., where $\mathbf{n} \cdot \mathbf{b} < 0$. Meanwhile, the Neumann boundary is divided into two complementary sections, inflow and outflow. In both sections, the original Neumann condition $\mathbf{n}_2 \cdot (a\nabla w_2) = \mathbf{n}_1 \cdot (a\nabla w_1)$ is changed by a Robin type condition $\mathbf{n}_2 \cdot (a\nabla w_2 - \mathbf{b}w_2) = \mathbf{n}_1 \cdot \mathbf{b}w_1^{(k)}$. These transmission conditions result from: (1) integrating by parts the convective term in Equation 2.1.3, and (2) the weak form of the approximation $\mathcal{L}_0 u = f$ in the partition Ω_b. Due to the fact that, $w_1^{(k)}$ is only equal to $w_2^{(k)}$ on Γ_{ab}^{in}, then the Robin condition is simplified to $\mathbf{n}_2 \cdot (a\nabla w_2) = 0$, while in the complementary section $\Gamma_{ab}^{out} = \Gamma_{ab} \backslash \Gamma_{ab}^{in}$, the entire Robin type condition is applied.

The iterative Robin-Neumann algorithm shown in Figure 2.4 takes into account the physical and numerical nature of the equations involved in the coupled problem. Similarly, alternative algorithms can be proposed in order to improve a particular aspect of the coupling. For instance, both Dirichlet-Neumann and Robin-Neumann algorithms consider a sequential execution order of the partition solvers. However, from a computational point of view, a *concurrent execution* order encourages the use of the computational resources. Nevertheless, concurrent algorithms can lead to other issues.

2.2 Multi-physics systems

The previous section introduced essential concepts behind domain decomposition methods. Principally, the emphasis is given to describe how the transmission conditions work. They establish the foundations for the solution of a multi-physics system whose partitions are defined by the kind of physics predominant in each of them. Summarizing, the continuity of certain transmission conditions throughout a coupling interface between a pair of non-overlapping partitions can be achieved by using an appropriate iterative procedure.

The focus of this section is on providing an overview of muti-physical systems where the heterogeneous domain decomposition methods above described can be applied. Contact of deformable bodies (CDB), conjugate heat transfer (CHT), and fluid structure interaction (FSI) problems are briefly introduced. Details are extensively discussed in Sections 4 and 5.

2.2.1 Contact of deformable bodies

Definition In general, when two bodies interact, the Newton's third law states that *the forces on the bodies from each other are always equal and opposite in direction* [41]. These forces are transmitted from one body to another by a compressible force along with tangential tractions due to friction. How these tractions are distributed throughout a deformable body as consequence of the contact can be described by the equation of balance of momentum

$$\rho \frac{\partial^2 \mathbf{d}}{\partial t^2} - \nabla \cdot \boldsymbol{\sigma} \quad = \quad \mathbf{f} \quad \text{in} \quad \Omega \tag{2.2.1}$$

where ρ, $\mathbf{d}$, $\boldsymbol{\sigma}$ and $\mathbf{f}$, stand for material density, displacement field, stress field, and volume force density.

When a deformable body Ω_a comes into contact with other deformable body Ω_b each of them changes its shape due to the *non-penetration condition*. Thus, at any time, the shape adopted by the surface of each body depends of the surface of other body. When these bodies are in mechanical equilibrium (i.e. the sum of all forces is zero), the final state is achieved. The state for each body can be determined as the sum of its undeformed state and the displacement field $\mathbf{d}$ given by the equation of the balance of momentum. As a coupling process, this can be done by firstly computing the overlapping area (or volume) of the undeformed bodies when they are placed in their desired final positions. After that, this overlapping $\mathbf{g}$ is *subtracted* from the deformable body. The final state is the one where the equilibrium is achieved.

Transmission conditions For a frictionless contact case (tangential tractions are neglected) with two deformables bodies Ω_a and Ω_b, the normal component of the overlapping $\mathbf{g}$ is enforced as Dirichlet boundary condition to the normal displacement $\mathbf{d}_b \cdot \mathbf{n}_b$ through the contact interface of the solid Ω_b

$$\mathbf{d}_b \cdot \mathbf{n}_b \quad = \quad \mathbf{n}_a \cdot \mathbf{g} \, \mathbf{n}_a \tag{2.2.2}$$

Simultaneously, the action-reaction principle requires that the normal force acting on the contacting surfaces must be equal and opposite. Thus, the normal component of the reacting stress $\boldsymbol{\sigma}_b \cdot \mathbf{n}_b$ can be enforced as Neumann boundary condition on the contact interface in the solid Ω_a

$$\mathbf{n}_a \cdot \boldsymbol{\sigma}_a \cdot \mathbf{n}_a \quad = \quad \mathbf{n}_b \cdot \boldsymbol{\sigma}_b \cdot \mathbf{n}_b. \tag{2.2.3}$$

2.2.2 Conjugate heat transfer

Definition The term conjugate heat transfer (CHT) is generally used when a fluid interacts with a solid through a common interface in which thermal energy is exchanged. An accurate solution of this problem requires to determine the temperature and heat flux distribution through the fluid-solid interface. For the fluid, the heat transfer is usually dominated by convection, while for the solid is characterized by a diffusive process. The continuity of this fluid-solid thermal coupling problem can be achieved by enforcing suitable transmission conditions between the energy equation of the fluid

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot (\rho H \mathbf{u}) \quad = \quad -\nabla \cdot \boldsymbol{q} \tag{2.2.4}$$

and the heat conduction equation in the solid

$$\frac{\partial T}{\partial t} + \nabla \cdot (-\alpha \nabla T) \quad = \quad Q \tag{2.2.5}$$

where ρ, $\mathbf{u}$, p, E, H, $\mathbf{q}$, α, and Q stand for density, velocity, pressure, total energy, enthalpy, density heat flux, thermal diffusivity, and external heat source, respectively.

Transmission conditions One of the most widely used CHT approaches imposes the density heat flux $\mathbf{q}_f = -\kappa_f \nabla T_f$ from the fluid into the solid domain by Neumann boundary conditions

$$\mathbf{q}_s \cdot \mathbf{n}_s = \mathbf{q}_f \cdot \mathbf{n}_f \tag{2.2.6}$$

while the solid imposes its surface temperature T_s onto the fluid domain through Dirichlet boundary conditions

$$T_f = T_s. \tag{2.2.7}$$

2.2.3 Fluid-Structure Interaction

Definition The problem considered here consists of an elastic body that can be deformed by the interaction with a fluid flow. This problem can be solved by the use of the arbitrary Lagrangian-Eulerian (ALE) formulation of the Navier-Stokes equations

$$\frac{\partial \mathbf{u}}{\partial t}|_{\mathbf{x}_0} + (\mathbf{u} - \mathbf{u}^G) \cdot \nabla \mathbf{u} - \frac{1}{\rho} \nabla \cdot \boldsymbol{\sigma} = 0 \tag{2.2.8}$$

that takes into account the domain's modifications that the displacements $\mathbf{d}_\Gamma$ of the body surface performs into the fluid.

Transmission conditions The effect of *the fluid on the body* are due to the pressure force that the fluid exerts on the interface of the body. Thus, the normal component of the fluid stress $\boldsymbol{\sigma}_f \cdot \boldsymbol{n}_f$ can be used as Neumann condition in the equation of balance of momentum for the body

$$\boldsymbol{\sigma}_s \cdot \boldsymbol{n}_s = \boldsymbol{\sigma}_f \cdot \boldsymbol{n}_f. \tag{2.2.9}$$

In order to consider the effect of *the body on the fluid*, firstly the current configuration of the fluid domain, consequence of the displacements $\mathbf{d}_\Gamma$ on the fluid-body interface, are estimated. The velocity $\mathbf{u}_\Gamma$ on the interface of the fluid domain results from the temporal variations of the interface displacements $\partial \mathbf{d}_\Gamma / \partial t$. The fluid velocity field $\mathbf{u}$ results from considering the velocity $\mathbf{u}_\Gamma$ as a no-slip boundary condition

$$\mathbf{u} = \partial \mathbf{d}_\Gamma / \partial t \tag{2.2.10}$$

to the ALE formulation of the incompressible Navier-Stokes equations.

2.3 Summary and remarks

The aim of this chapter is to introduce the mathematical basis of the coupling approaches used in this thesis

Firstly, the domain decomposition method for non-overlapping partitions was presented. Then, the iterative Dirichlet-Neumann algorithm was introduced. Finally, an overview of the multi-physics systems to be used through this thesis was presented.

The domain decomposition method for non-overlapping partitions is based on the transmission conditions (suitable boundary conditions throughout the coupling interface). These conditions establish that a given variable and its corresponding flux must be continuous throughout the surface defined between each pair of partitions. To maintain the continuity is fundamental in order to ensure that the solution achieved in each partition converges to the real solution of the entire domain.

The diffusion-transport-reaction equation was used to show how the transmission conditions operate. By using the weak form of this equation, it was found that an additional term is introduced when the problem is decomposed into partitions. This term defines the more appropriate transmission conditions for this problem. Through a suitable use of these conditions, the iterative Dirichlet-Neumann algorithm can be used to obtain the solution. A partitioned multi-physics approach can essentially be addressed by using domain decomposition methods with different kind of physical problems in each partition [40]. This has been exemplified by assuming an advection-diffusion system with a partition completely governed by the convective term. Under this assumption, the iterative Dirichlet-Neumann used to solve the original advection-diffusion system was modified. The modification is mainly due to the fact that new transmission conditions were necessary in order to ensure that the new coupled system is still well posed.

An iterative coupling approach results from each set of physical equations composing a given multi-physics system. In this sense, the procedure used in both examples involving the advection-diffusion system can be used to derive suitable transmission conditions for each of the multi-physics systems studied in this thesis. As shown in the following chapters, the transmission conditions for contact and conjugate heat transfer problems are essentially similar to those used in the advection-diffusion system, i.e., Dirichlet-like conditions imposed in a partition, and Neumann conditions enforced on the other. The difference between them is the physical meaning of the transmission conditions for each problem. In the case of contact, the transmission conditions are related to the conservation of the momentum, while in conjugate heat transfer, the transmission conditions ensure the continuity of the thermal energy.

3

Multi-physics Software

We often think that when we have completed our study of one we know all about two, because 'two' is 'one and one.' We forget that we still have to make a study of 'and'

Eddington, A. S. (2012)

In this chapter, details related to the implementation of a *parallel coupling tool* that allows the solution of large-scale partitioned multi-physical simulations are given. The coupling tool developed here (the PLE++ library) allows dealing with the *parallel coupling problem* using a coupling approach to combine the solution achieved by the single-parallel solver present in each partition.

One of the main goals of this chapter is to present an overview of the challenges related to the implementation of partitioned coupling approaches by using the PLE++ library. In order to do that, the rest of the section is divided in three main parts. The first part, Section 3.1, describes the *parallel execution mode* that is required for the solution of two *coupling schemes*: parallel and staggered. Section 3.2 describes the implementation of a coupling tool that allows to solve a multi-physics problem using parallel or staggered schemes. In the last part, Section 3.3, two cases are briefly discussed. The fist addresses the parallel implementation of a staggered coupling scheme. In the second case, an example involving heterogeneous computing is shown. A brief overview of coupling tools recently developed is given in Section 3.4. Finally, in the last section a summary of the chapter is given.

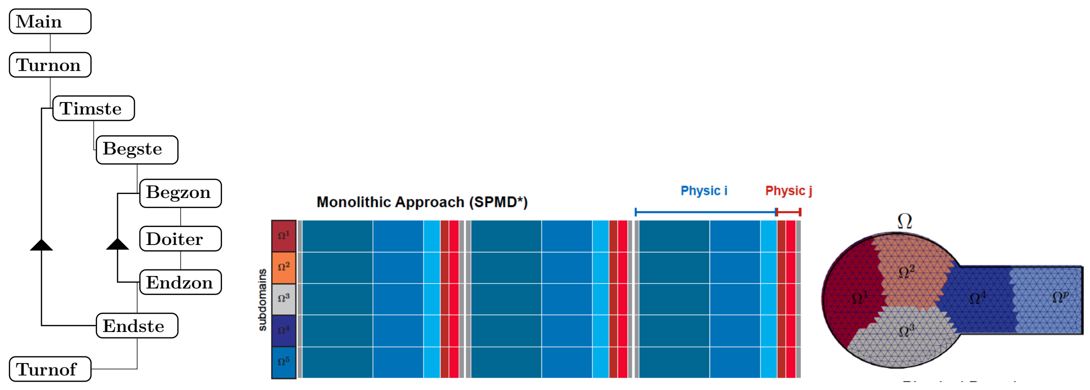
3.1 Parallel execution modes

This section gives an overview of the execution of two parallel coupled codes using two partitioned approaches: parallel and staggered. Firstly, a brief overview of the structure of a parallel code is given. After that, two execution modes of MPI known as *simple-program-multiple-data* and *multi-program-multiple-data* are briefly presented. Finally, the relation between the execution modes and the coupling schemes is discussed.

3.1.1 Parallel code structure

The type of problems addressed in this work are those where the equations governing the system evolves in time. Additionally, these equations can be highly non-linear, so that sophisticated numerical algorithms must be used to solve them. The solution of these systems involves a set of *stages* that can be divided into three: pre-processing, solution, and post-processing. Pre-processing can include the discretization of the physical domain, as well as the domain decomposition of the discretization. The domain decomposition allows solving the set of equation, resulting from the discretization, using small sub-equations. Each of them must be solved by the processors in which such sub-equation is allocated. Additionally to the spatial discretization, a temporal discretization is also neccesary when transient problems are considered. Finally, post-processing generally refers to the analysis of the numerical results, e.g., visualization.

The structure of a parallel code taking into account the stages described above is depicted in Figure 3.1(a). *Turnon* and *Turnof* tasks represent pre- and post-processing stages, respectively. In the case of non-linear transient problems, the solution stage is represented by a set of tasks. *Timste* represents a loop dealing with the time step strategy. Each of these time steps is composed by five tasks. *Begste* performs operations related to the initialization of each time step. *Endste* checks if the overall system has achieved a given temporal convergence, additionally, if necessary, it can perform post-processing tasks related to the current time step. The last three tasks are associated to the solution of the set of equations arising from the discretization. These tasks consider the possibility of solving multi-physics problems through the sequential execution of their physical components, see the next section for details. *Begzon* executes a loop through each equation $\mathbf{A}_{ii}\mathbf{u}_{ii} = \mathbf{b}_{ii}$ coming from the discretization of each physical component. In *Doiter* an iterative method for the numerical solution of a non-symmetric system of linear equations is used to solve individually the equation associated to each physical component ii. The interaction that a given physical component i enforces over another component j is considered introducing an extra term $\mathbf{A}_{ij}$ into the original system. Finally, *Endzon* determines if the set of equations has achieved the target solution, otherwise the procedure is repeated from *Begzon*. Figure 3.1(b) shows how these stages are executed sequentially by a set of processors p_i assigned to the parallel code i. Details are discussed in the next section.



(a) (b)

Figure 3.1: (a) Parallel code structure (b) Single-program-multiple data execution mode.

3.1.2 Single-program-multiple-data

Once a given partition Ω^l has been divided into as many sub-domains $\{\Omega^l_1, \Omega^l_2, ..., \Omega^l_{p^k}\}$ as p^k processors, each processor $\alpha \in p^k$ can execute any task β during a time $\Delta T_{\alpha\beta}$. Each of these times can vary with the type of processor, the size of each sub-domain, or with the complexity of the task to be executed. An ideal case where *identical* processors execute *simultaneously* the same task through *different* sub-domains is depicted in Figure 3.1. This case shows three time steps of a multi-physics system formed by two physical components i and j sequentially executed one after the other. The partition has been divided through five processors. Each of these processors can execute as many times as necessary each of the seven tasks described above, however only *Begzon*, *Doiter*, and *Endzon* are shown.

The procedure described above can be classified as a mode of MPI known as *single-program-multiple-data* (SPMD) [42]. In general, the tasks are divided and executed simultaneously on multiple processors in order to achieve the results faster. The SPMD mode is widely used in cases where the physical components act across the same physical domain, for instance, a turbulent reacting flow. Another example is a monolithic solver. In this case, the multi-physical system is completly solved using a single matrix equation.

3.1.3 Multiple-program-multiple-data

The partitioned multi-physical approach studied in this thesis is based on another mode of MPI known as *multiple-program-multiple-data* (MPMD) [42]. In this case,

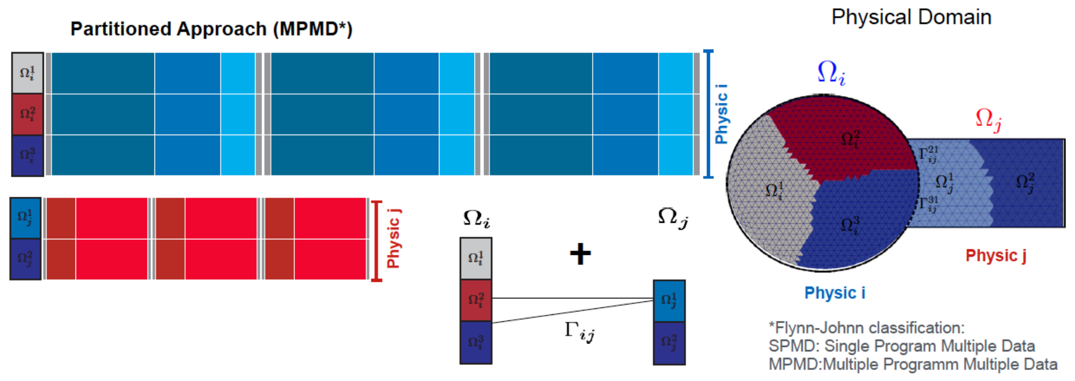

Partitioned Approach (MPMD*)
Physic i
Physic j
Ω_i
Ω_j
Γ_{ij}
Physical Domain
Physic i
Physic j
*Flynn-Johnn classification:
SPMD: Single Program Multiple Data
MPMD:Multiple Programm Multiple Data

Figure 3.2: Multiple-program-multiple-data execution mode.

the multi-physics system has regions governed by particular physics, so that each of these regions define a partition. The essential idea is that the physical solution associated to each partition is found *independently* to each other, so that each partition has its *own* allocation of processors. Additionally, extra communication between partitions is necessary in order to account for the interaction between them.

Figure 3.2 depicts an example of the partition of a multi-physical system. Lets suppose that a total number $p = p^1 + p^2$ of five processors are divided between two partitions Ω^1 and Ω^2 each of them are governed by certain physics interacting exclusively across a common boundary interface Γ, so that $p^1 = 3$ and $p^2 = 2$, respectively. Due to the fact that each partition is *independent*, its execution is equivalent to solve each partition using a SPMD mode. The physical interaction between partitions is considered to take place exclusively in processors where the interface $\Gamma^l \in \partial\Omega^l$ is hosted. Thus, a given sub-domain Ω^i_k hosting a section Γ^i_k on its boundary interface $\partial\Omega^i_k$ can exchange information with another sub-domain section Ω^j_l if and only if $\partial\Omega^i_k \cap \partial\Omega^j_l \neq \emptyset$. In the present case, the physical interactions are regarded to take place between sub-domains Ω^1_2 and Ω^2_1, and sub-domains Ω^1_3 and Ω^2_1.

Summarizing, a partitioned multi-physical approach based on the MPMD execution mode to solve each partition as a SPMD mode requires extra communication between processors. This extra communication takes place in those processors allocated in different partitions with common regions. These regions are defined by the overlapping between the sub-domains hosted in the processors. Details about how to determine if a pair of sub-domains overlap is given in Section 3.2.1.

3.1.4 Coupling schemes

Additionally to the execution mode, another important point is the execution sequence of the partitions. The sequence is defined by the scheme used to couple the multi-physical system. Two cases are considered here: *parallel* and *staggered*

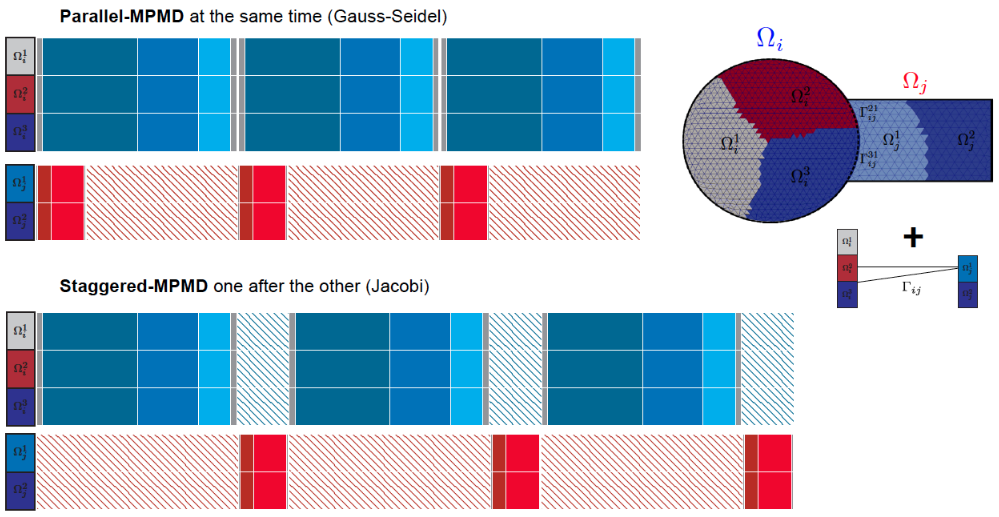


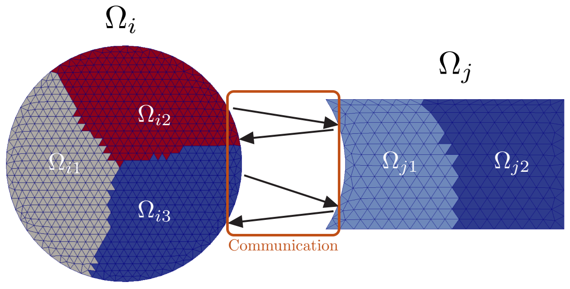
Figure 3.3: Parallel and staggered executions for a partitioned multi-physical system.

schemes. The parallel scheme is characterized by the fact that partitions are executed concurrently, i.e., at the same time. On the other hand, the partitions in the staggered scheme are executed one after the other. Generally, the resulting data obtained from a partition can be used as input for the other, so that the data exchange takes place after a given partition finalizes its execution, and just before the second partition initializes its own execution.

Figure 3.3 shows the execution sequence of the parallel and staggered schemes for the partitioned multi-physical system described in the last section; two partitions Ω^1 and Ω^2 are solved independently through a SPMD mode. In the parallel scheme, the partitions start their execution at the same time. A common situation is that the fastest solver (in this case Ω^2) must wait while the slowest solver (Ω^1) finishes its execution. Once Ω^1 finished its execution, the data exchange is performed between pairs of sub-domains previously identified as coupled. Now, the data received is used by each partition as an input which can be employed to update the properties of the current partition. Finally, the whole procedure can be performed as many times as necessary. Unlike the parallel scheme, a partition must wait while the other is being executed in the staggered scheme. In general, the data exchange takes place after a given partition starts its execution (in this case Ω^1), and finish it. Once the other partition has received the data, it can be executed. The updated data is now sent to the started partition Ω^1, and then the procedure can be repeated.

The choice of the scheme is generally defined by the physical interactions between partitions involved in the coupling. For instance, the implementation of the iterative Robin-Neumann algorithm described in Section 1.1.2 involves the use of the staggered scheme. Thus, for each iteration k of this particular case, partition Ω^a is firstly solved. Then the values of $w_1^{(k)}$ through the coupling interface Γ_{ab}^{in} are sent, and once received them in Ω^b, the values of $w_2^{(k)}$ can be obtained enforcing a Robin boundary condition on Γ_{ba}^{out}. Finally, the values to be enforced on the Dirichlet interface Γ_{ab}^{in} can be updated and sent back to Ω^a. Other coupling cases where the staggered scheme is used are discussed in Chapters 4 and 5.

An important point is the parallel performance of each coupling scheme. In general, the performance of a parallel application is strongly related to two parameters, concurrency and load imbalance. The concurrency refers to the number of tasks that can be executed simultaneously at any given time. The load imbalance is consequence of assigning different amount of workload to the available processors. One of the main drawbacks of coupling schemes is related to the unbalance introduced when an inappropriate selection in the allocation of the available processors is done. In order to overcome this drawback, a strategy based on a suitable resources distribution is introduced in Chapter 6. The goal is to achieve an optimal balance in the data distribution so that, the load is balanced not only for each component, but also for the whole coupled system.



(a)

```
for $\boldsymbol{\Omega}_\beta^l = \{\boldsymbol{\Omega}_\beta^1, ..., \boldsymbol{\Omega}_\beta^{p_\beta}\}$ do
    if $\boldsymbol{\Omega}_\alpha^k \cap \boldsymbol{\Omega}_\beta^l \neq \emptyset$ then
        Interpolation$_\alpha(\mathbf{v}_\alpha^k, \mathbf{r}_\beta^l) \to \mathbf{v}_\beta^l$
        Communication$_{\alpha\beta}(\mathbf{v}_\beta^l)$
    end
end
```

(b)

Figure 3.4: (a) Disjoint partitions with their own sub-domains, $\boldsymbol{\Omega}_\alpha^k$ and $\boldsymbol{\Omega}_\beta^l$. The data transfer is only performed between overlapping sub-domains, in two steps: interpolation and exchange. The interpolation maps property values $\mathbf{v}_\alpha^k$ on $\boldsymbol{\Omega}_\alpha^k$ to values $\mathbf{v}_\beta^l$ on $\boldsymbol{\Omega}_\beta^l$ through the coordinates $\mathbf{r}_\beta^l$. (b) *Brute force* transfer algorithm. The overall system $\boldsymbol{\Omega}$ is divided into a α number of partitions $\boldsymbol{\Omega}_\alpha$, each of them constituted of a number p_α of sub-domains $\{\boldsymbol{\Omega}_\alpha^1, \boldsymbol{\Omega}_\alpha^2, ..., \boldsymbol{\Omega}_\alpha^{p_\alpha}\}$. The number of communication q_{ij}^k that a sub-domain $\boldsymbol{\Omega}_i^k$ should perform with other partition j, is defined by the number of times that this sub-domain overlaps with the sub-domains $\boldsymbol{\Omega}_j^k$. In this particular case, the overlapping surface $\boldsymbol{\Gamma}_{ij} = \boldsymbol{\Omega}_i \cap \boldsymbol{\Omega}_j \neq \emptyset$ defines the communication between the partitions $\boldsymbol{\Omega}_i$ and $\boldsymbol{\Omega}_j$. It takes place between the sub-domains $\boldsymbol{\Omega}_i^2$ and $\boldsymbol{\Omega}_i^3$ of the partition i, and the sub-domain $\boldsymbol{\Omega}_j^1$ of the partition j.

3.2 Coupling tool

Lets define a coupling tool as a software that allows solving a multi-physics problem by combining different single-physics codes. This coupling tool should satisfy some important characteristics such as robustness, efficiency, and scalability for multi-physical algorithms [6]. In light of the above, a high performance computing tool designed to solving multi-physics simulations is presented. The Parallel and Locator Exchange Library++ (PLE++) described here is a C++ environmental library with the capability of allowing the communication between parallel applications written in C/C++, Fortran or Python. Note that the PLE++ library is based on the Parallel and Locator Exchange library, originally developed to couple the CFD code Code_Saturne, and the heat transfer code Syrthes [43]. Some of the most important features of the PLE++ are described in the rest of this section.

3.2.1 Parallel Location and Exchange Library++

The workflow of PLE++ can be divided into three main stages. Firstly, the tool defines the set of MPI communicators to be used by each partition. In the second stage, a localization algorithm is applied and as result a peer-to-peer communication layout is established. The exchange of data is performed in the third stage and repeated as often as necessary.

The data exchange performed by the coupling schemes only takes into account sub-domains of each partition with common overlapping regions. Since each partition is executed independently, it is possible to assume for each of them that the workload across the processors has been performed by a suitable domain decomposition method, for instance using Metis [44]. The set of vertices assigned by Metis to each processor along with their respective connectivities define the sub-domains of each partition. Thus, if a pair of sub-domains, each of them belonging to two different partitions, share a common region (surface or volume), such sub-domains are involved on the coupling. This means that operations related to the coupling (communication and interpolation) take place locally. Interpolations are performed on each processor, while communications are performed in pairs of processors (by a parallel peer-to-peer communication approach), see Algorithm 1 and Figure 3.4.

Parallel localization

As shown in Figure 3.5, each partition solver is executed by different set of processors, and each of these processors can only contain a sub-domain of such partition. Due to the fact that, the partitions are independent between them, the vertices and its connectivities (elements) are locally known into each partition, and not between them. In order to create additional connectivities, each vertice of a local partition must know the element of the other partitions containing it. Additionally, each vertice must know which processors contain such elements. Thus, at the end, it is necessary to know: a) vertices involved in the coupling, b) the cells contained, and c) the processors exchanging information.

The rest of the section describes the algorithm used for localization of the vertices and elements involved in the coupling. The algorithm makes use of a hierarchical localization based on geometrical properties of the partitions. It is divided into two main parts: Global and Local search. See Figure 3.6.

Global search In the global search, the processors are associated in pairs related to the coupling. Moreover, it checks which vertices and elements into such pairs of processors can be potentially related.

Without limiting the generality of the algorithm, lets suppose that each partition Ω_a and Ω_b is divided into sub-domains $\Omega_a^k = \{\Omega_a^1, \Omega_a^2, \Omega_a^3\}$ and $\Omega_b^l = \{\Omega_b^1, \Omega_b^2, \Omega_b^3, \Omega_b^4\}$, and each of them is assigned to only a single processor p, see Figure 3.7. Such processors can be classified as local or remote. *Local* processors are those that

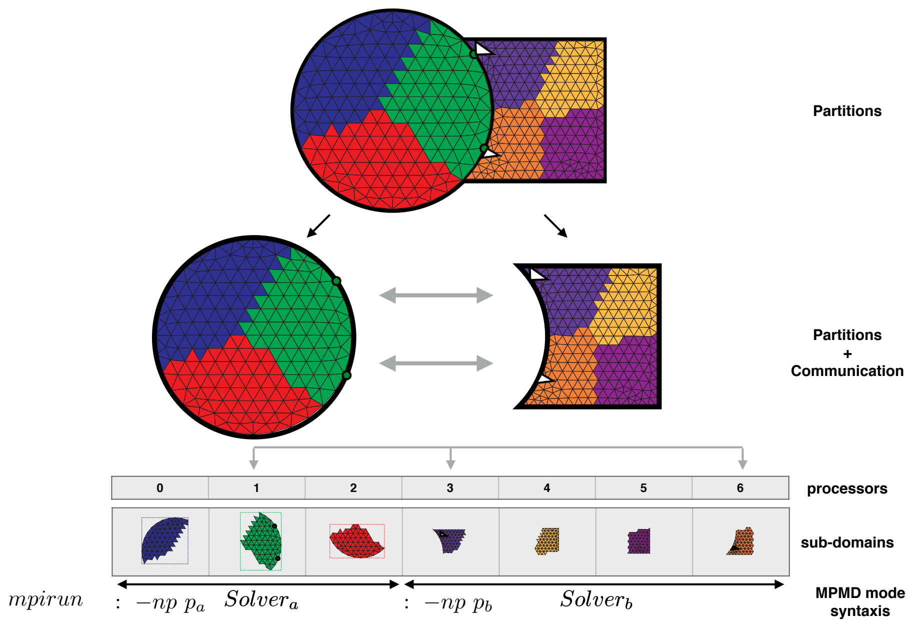

Partitions
Partitions + Communication
0
1
2
3
4
5
6
processors
sub-domains
$mpirun$: $-np\ p_a$ $Solver_a$: $-np\ p_b$ $Solver_b$
MPMD mode syntaxis

Figure 3.5: The overall physical domain is divided into *independent* partitions, each of them characterised by particular physics. Additionally, each of these partitions can be solved using different set of processors, and each of such processors can contain an unique sub-domain's partition. The figure sketches a physical domain formed by two partitions $\boldsymbol{\Omega}_a = \{\boldsymbol{\Omega}_a^1, \boldsymbol{\Omega}_a^2, \boldsymbol{\Omega}_a^3\}$ and $\boldsymbol{\Omega}_b = \{\boldsymbol{\Omega}_b^1, \boldsymbol{\Omega}_b^2, \boldsymbol{\Omega}_b^3, \boldsymbol{\Omega}_b^4\}$. A total of seven processors are used to the whole configuration, three processors are allocated to $\boldsymbol{\Omega}_a$ and four to $\boldsymbol{\Omega}_b$. Two vertices hold in the surface of the sub-domain $\boldsymbol{\Omega}_a^2$ (processor 1) are contained by two elements in different sub-domain $\boldsymbol{\Omega}_b^1$ (processor 3) and $\boldsymbol{\Omega}_b^4$ (processor 6). Thus, processors 1, 3 and 6 are involved in the coupling, and communication between such processors must be performed to exchange physical properties between partitions.

belong to each partition, while *remote* processors are used to execute the rest of partitions. Thus, the global search seeks the remote processors whose sub-domains

```
$Q_a^k = GetBox(\mathbf{\Omega}_a^k),\ k = 2$
for $l = \{1, 2, ..., p_b\}$ do
    $Q_{rmt}^l = SendRecv_l(Q_a^k)$
    if $Q_a^k \cap Q_{rmt}^l \neq \emptyset$ then
        $\mathbf{Id}_a^k = WithinBox(\mathbf{R}_a^k, Q_{rmt}^l)$
        $\mathbf{r}_a = \mathbf{R}_a^k(\mathbf{Id}_a^k)$
        $\mathbf{r}_{rmt} = SendRecv_l(\mathbf{R}_a^k(\mathbf{Id}_a^k))$
        $\mathbf{Id}_a^{\mathbf{r}_b} = LocateCells(\mathbf{r}_{rmt}, \mathbf{\Omega}_a^k)$
        $\mathbf{Id}_a^{\mathbf{r}_a} = SendRecv_l(\mathbf{Id}_a^{\mathbf{r}_b})$
        $\mathbf{ID}_a^k[j] = (\mathbf{Id}_a^k, \mathbf{Id}_a^{\mathbf{r}_a})$
    end
end
$\mathbf{P}_a^k = ArgMin(\mathbf{D}_a^k[:, 1 : p_b])$
$\mathbf{C}_a^k = \mathbf{L}_a^k[:, \mathbf{P}_a^k]$
```

(a)

```
$T_b^j = GetOctree(\mathbf{r}_b^j)$
foreach $\mathbf{e}_i \in \mathbf{\Omega}_a^i$ do
    $q_{ib} = GetQueryBox(\mathbf{e}_{ib})$
    $\mathbf{Id}_b^j = WithinQueryBox(T_b^j, q_{ib})$
    $\mathbf{l}_b^j, \mathbf{d}_b^j \mathrel{+}= WithinCells(\mathbf{Id}_b^j, \mathbf{r}_b^j)$
end
```

(b)

Figure 3.6: (a) Global search. (b) Local search.

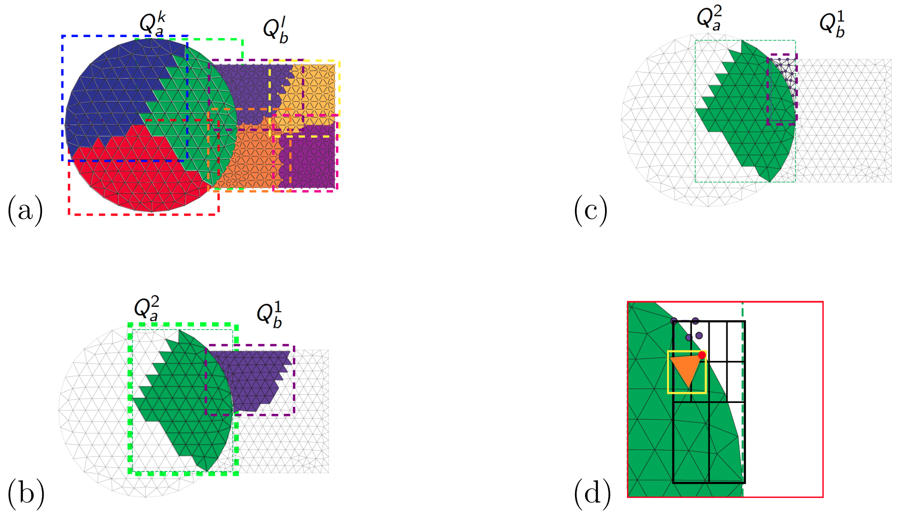


Figure 3.7: (a) Bounding boxes $\mathbf{Q}_a = \{Q_a^1, Q_a^2, Q_a^3\}$ and $\mathbf{Q}_b = \{Q_b^1, Q_b^2, Q_b^3, Q_b^4\}$ around each sub-domain used to find processors related to the coupling. (b) If two bounding boxes hold in different processors overlap, i.e. $Q_a^2 \cap Q_b^1 = \emptyset$, a sub-set of remote vertices in the partition $\mathbf{\Omega}_b^1$ can be contained by local elements in the partition $\mathbf{\Omega}_a^2$. (c) Remote vertices and local elements situated into the overlapping region $Q_a^2 \cap Q_b^1$. An octree search T_a^2 is created only with the remote vertices situated into the overlapping region. Finally, homogeneous barycentric coordinates are used to determine which of such vertices are localized into a particular element.

are overlapped. This is performed by a Multiple Program Multiple Data (MPMD) model described in Section 3.1.3. where all the processors execute the Algorithm 2, concurrently to each other, see Figure 3.6.

First, the sub-domains Ω_a^k and Ω_b^l are characterized by bounding boxes Q_a^k and Q_b^l, see Figure 3.7. The information contained locally by each bounding box Q_a^k is shared with each remote partition Ω_b^l, and vice versa. Thus, the bounding boxes Ω_b^l of each remote sub-domain are available into each local sub-domain. Now, the local and remote bounding boxes are compared. If such bounding boxes overlap $Q_{ab}^{kl} = Q_a^k \cap Q_b^l \neq \emptyset$, the processors associated to these sub-domains can be related to the coupling. The next step involves the searching of the *local* vertices $\mathbf{r}_a^k$ and their respective connectivities $\boldsymbol{\omega}_a^k$ which can be allocated into the overlapping region Q_{ab}^{kl}. Now, the sub-set of local vertices $\mathbf{r}_a^k$ is shared with the remove processor l where the overlapping partition $\boldsymbol{\Omega}_b^l$ is hold, and vice versa. After that, the sub-set of local elements $\boldsymbol{\omega}_a^k$ are used to determine if some of such remote vertices are contained into local elements (see the next section for details). Once the remote vertices localized into local elements are found, their identifiers $\mathbf{Id}_{rmt}$ are shared respect to the remote processor l, and vice versa. As a result, the identifiers $\mathbf{Id}_a$ received are used to assign the remote processor into each local vertex $\mathbf{r}_a^k$ that has been localized. Thus, the results of the global search are: (a) mapping list of each local vertex with its corresponding remote processor, and (b) list of local elements occupied by remote vertices.

The list of remote processors where the elements associated to each local vertex are allocated is used to develop a communication scheduling. This scheduling determines the processors involved in the coupling, and when and how the data is exchanged between them. The list of local elements and the coordinates of the remote vertices are used to perform operations related to the coupling, for instance, interpolation of values from the local sub-domain to the remote sub-domain.

Local search In the local search, the far vertices are associated to local elements. This is done by an octree search algorithm along with some suitable method that allows finding if a vertice is or not inside a given element, see Algorithm 3 and Figure 3.6.

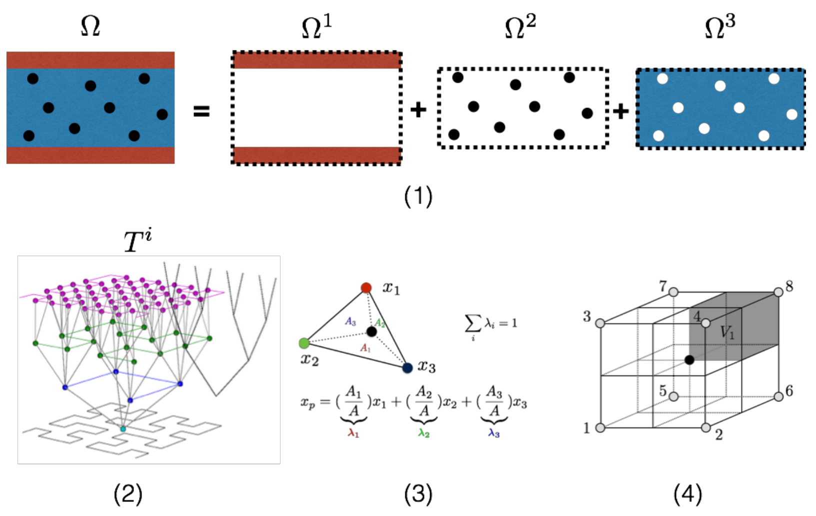
Once the remote vertices $\mathbf{r}_{rmt}$ and the local elements $\boldsymbol{\omega}_a^k$ allocated into the overlapping region Q_{ab}^{kl} has been identified. it is required to search the sub set of remote vertices allocated into a sub set of local elements. In order to do that efficiently, an octree search is performed. Such octree T_b^j is created using the remote vertices $\mathbf{r}_{rmt}$. Now, each element $\mathbf{e}_i$ into the set of local elements $\boldsymbol{\omega}_a^k$ searches for the vertices allocated within it. Thus, for each local element, a query box is created and used as input for the octree. Such query box is compared with the range occupied by each node of the octree. If the dimensions of a leaf are smaller than the dimensions of the query box, the sub set of vertices can be considered as the closest to such element. The last step uses a suitable method to decide if a vertex is located in an element.

For instance, in the case of triangles or tetrahedrons homogeneous barycentric coordinates can be used, or even spherical barycentric coordinates [45] for general cases. The algorithm is detailed in Figure 3.7.

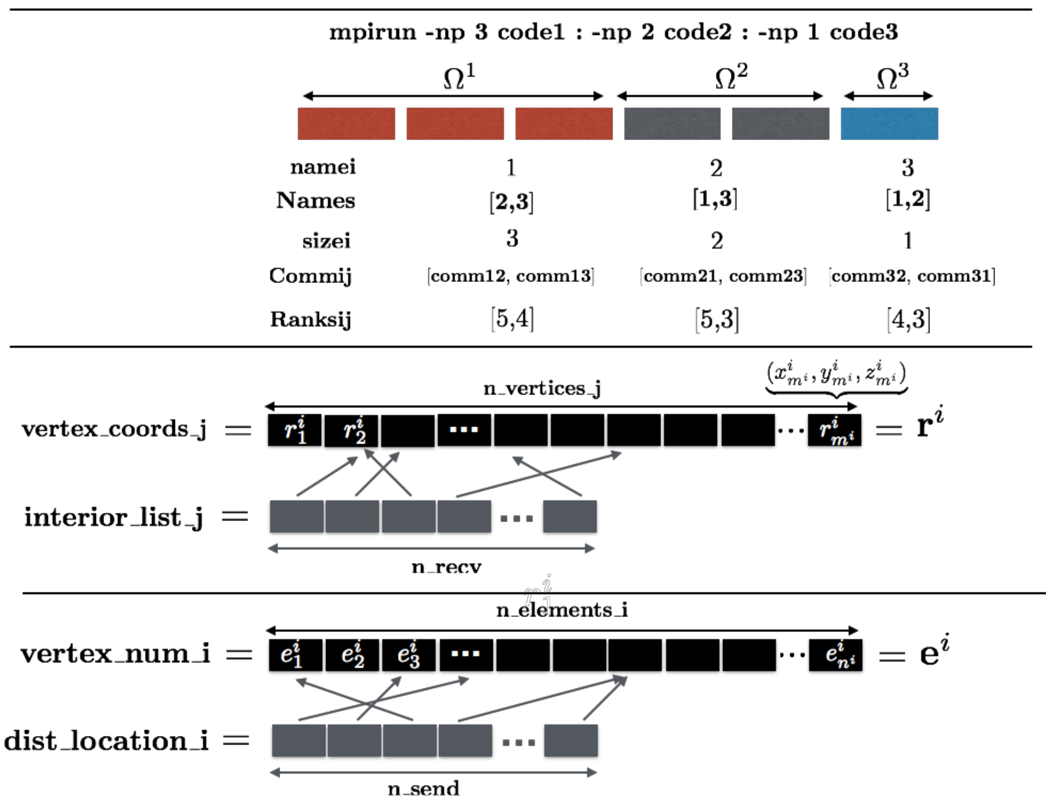
3.2.2 Application programming interface

Communication mechanism Figure 3.8(b) shows the syntax used to execute different instances of the codes involved in a coupled simulation. Each code instance use a SPMD execution mode, so that each code i is responsible for independently solving a given domain Ω_i using a determined number of processors p_i. Once the coupled simulation is lunched, the number of requested processors is associated to each code. For this, local- and inter- communicators must be created. The local-communicators are used by each code to exchange information between the local processors p_α associated to each partition α. Inter-communicators are those responsible of exchange information between processors p^l_a and p^k_b allocating the coupled sub-domains Ω^l_a and Ω^k_b respectively, see Section 3.2.1. The communicators are created by the PLE++ API during the initialization of the coupling, see Listing 3.1. After PLE++ initialization (line 4), the set of communicators **Commij** is created (line 11). Each communicator **Commij** has associated a group of processors belonging to a particular coupling. The different couplings among the existing code instances are determined by the list of identifiers **Names**. Each identifier of the list **Names** corresponds to the string **namei** used to tag each one of the solver codes. For the case depicted in Figure 3.8(b), the solver identified as **1** has two communicators, **comm12** and **comm13**, each of them with 5 and 4 processors respectively.

Localization As stated in the previous section, for each pair of overlapping partitions Ω^α and Ω^β, the overlapped elements (cells or points) of partition Ω^α must be associated to the overlapped elements of partition Ω^β, see Figure 3.8 and Figure 3.9, Listing 3.2, line 3. A local mesh structure must be associated to a set of points in each partition. The cells defined by the vertices **vertex_coords_i** and their connectivities **vertex_num_i** are used as a base mesh to find the points of the other partition, which overlaps with the local mesh. Furthermore, the set of points **vertex_coords_j** are the coordinates of the local mesh, which is visible to be located by the other partition. These points, for instance, can be defined by *the centres of the cells*, *the gauss points of the cells*, or even equal to **vertex_coords_i**. The other partition searches through all the points in the set **vertex_coords_j** overlapping its mesh. Once the points, vertices and connectivities are defined, PLE++ applies Algorithms 3.6 to look for the set of local cells **dist_locations_i** holding the set of localized points **dist_coords_j**. Note that the set of localized points **dist_coords_j** is a *subset* of **vertex_coords_j** established by the other parti-



(a)



```
mpirun -np 3 code1 : -np 2 code2 : -np 1 code3
```

	Ω^1	Ω^2	Ω^3
namei	1	2	3
Names	[2,3]	[1,3]	[1,2]
sizei	3	2	1
Commij	[comm12, comm13]	[comm21, comm23]	[comm32, comm31]
Ranksij	[5,4]	[5,3]	[4,3]

(b)

Figure 3.8: Localization procedure. (a) Partitions (1), tree data structure (2), barycentric coordinates (3), and particle source in cell interpolation (4). (b) Launch syntax, sketch of the vectors related to the PLE++ API.

```
from mpi4py import MPI
from plepp  import PLEPP
...
CD = PLEPP()
CD.set_app_name(namei);
CD.set_world_comm(MPI.COMM_WORLD)
...
local_comm = CD.set_mpi_comms()
local_size = local_comm.Get_size()
...
Commij = {namej:CD.get_mpi_commij(namej) for namej in Names if not
    namei==namej }
Ranksij = {namej:commij.Get_rank() for commij in  Commij }
...
```

Listing 3.1: Defining communicators

```
...
CD.locator_create(local_comm, commij, tolerance)
CD.locator_set_mesh(n_vertices_i, n_elements_i, vertex_coords_i,
    vertex_num_i, n_vertices_j, vertex_coords_j)
...
```

Listing 3.2: Defining geometry

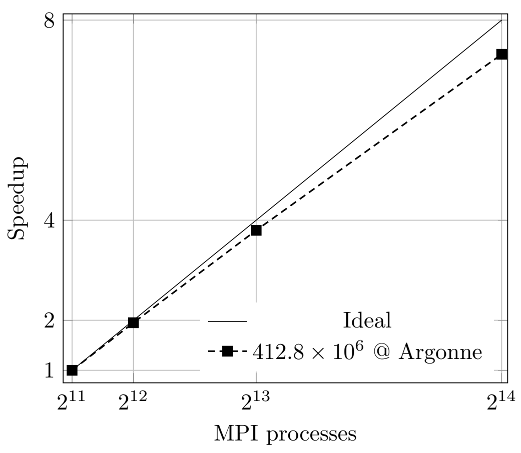
```
...
n_send = CD.get_n_dist_points()
...
# integer, n_send
dist_locations_i = CD.locator_get_dist_locations(  )
...
# double, n_send * dim
dist_coords_j = CD.locator_get_dist_coords()
...
# double, n_send * dof
var_ij = interpolation(dist_coords_j, propa_i, vertex_coords_i,
    vertices_num_i)
...
# double, n_recv * dof
var_ji = CD.locator_dexchange(var_ij)
...
```

Listing 3.3: Sending

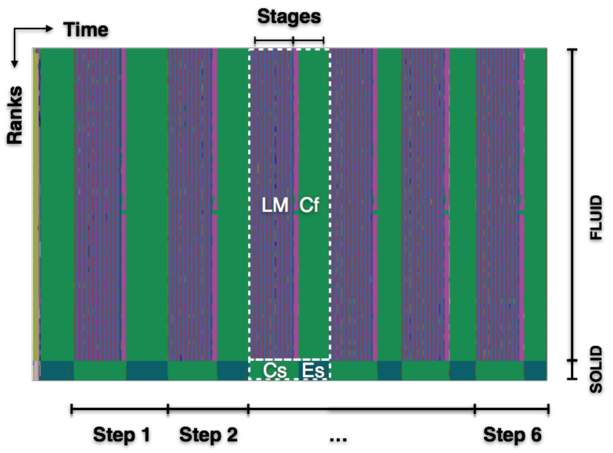
```
...
n_recv = CD.get_n_interior();
...
# double, n_recv * dof
var_ji = CD.locator_dexchange(var_ij)
...
# integer, n_recv
interior_list_j  = CD.locator_get_interior_list( )
propb_i[interior_list_j-1] = var_ji[1:n_recv]
...
```

Listing 3.4: Receiving

Figure 3.9: PLE++ application programming interface (API).



(a)



(b)

Figure 3.10: (a) Strong-scaling curve for the fluid partiton (without coupling) and (b) Trace of coupling case (flat-plate experiment).

tion. In addition, each localized point of **dist__coords__j** is associated to only one cell of **dist__locations__i**. However, a cell can hold more than one point.

Data exchange Once PLE++ creates the connection between the cells and points of each partition within the overlapping region, it exchanges (send and receive) the data related to the physical variables involved in the coupling scheme, see Figure 3.9 and Listing 3.3 and 3.4. Before the exchange, each cell of **dist__locations__i** is used to interpolate the local physical values **propa__i** on each localized point of **dist__coords__j**. The result of the interpolation is stored in the variable **var__ij**. Afterwards, PLE++ sends this variable from the local partition to the other partition. The values **var__ji** received by the other partition are then used to enforce the values of the property **propb__i** in its own domain.

After the exchange of information, each partition uses the received data to compute the solution. Once this is done, the last two stages are repeated when necessary.

3.3 Application cases

Two cases are briefly discussed below. The first addresses the parallel implementation of a staggered coupling scheme. The second case describes the implementation of a heterogeneous simulation on GPU and CPU.

3.3.1 Parallel performance analysis

In order to apply the approaches described along this chapter to large-scale problems, two factors must be taken into account: (1) the performance of the computations for each individual partition and (2) the performance associated to the selected coupling scheme. This section addresses the parallel implementation of the coupling scheme for a confined premixed jet flame described in Section 5.4.3. In particular, the analysis presented here is limited to show the availability of the partitions to achieve a good performance, and also to show the possible limitations of the coupling performance. In Section 6, a strategy to run partitioned multi-physics applications on extreme scale architectures with an optimal parallel performance will be presented.

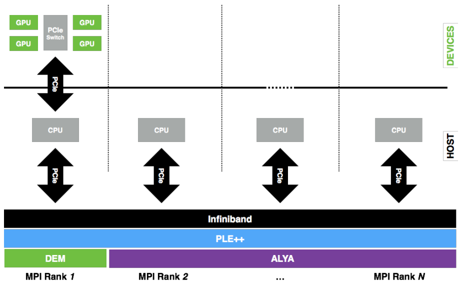
In general, a reduction in parallel performance when computing single partitions usually leads to a full breakdown of the performance of the coupling scheme, despite the algorithm can be very efficient. Because of that, the analysis of the parallel performance of the code without coupling is the first step to be verified. Figure 3.10 (a) shows the strong-scaling curve performed on the supercomputer Vesta at the Argonne National Laboratory. The curve shows an excellent performance of the code up to 16384 MPI processes on the modelling of the fluid partition for the confined premixed jet flame. The efficiency achieved for this problem was around 91%. As the solid partition uses the same numerical framework as the fluid, the focus can now be restricted to the features of the coupling scheme.

The parallel implementation of the staggered coupling scheme is now considered. The coupling execution consisted of 64 MPI processes divided between the fluid and solid partitions (60 and 4, respectively). The number of time steps performed were limited to six. Figure 3.10 (b) shows the parallel execution of the case unfolds over time (the trace) obtained by the parallel performance tool HPCToolkit [46]. The figure shows the time line of each MPI process with colours in the vertical direction showing different computing stages. In the case of the fluid, the first stage (LM) represents the solution of the low Mach equations, while the second stage (C_f) corresponds to the coupling. At the same time, there are also two stages in the solid case, the coupling (C_s) and the solution of the energy equation (E_s).

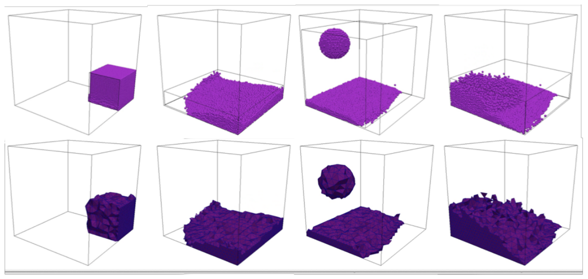
As stated above, in the sequential strategy, the domains are solved one after the other, which can be clearly seen on the trace. Each time step sequence starts with the solution of the low Mach equations in the fluid partition (LM), while the solid partition is waiting for the solution (C_s). Once the solution of the fluid partition is achieved, the solid partition performs its own solution (E_s). It should be noted that between the solution stages LM and E_s no overheads due to the exchange of information were introduced into the algorithm. This approach requires that both partitions wait during the calculation stage of the other partition introducing a limitation in the maximum performance that can be achieved when the number of MPI processes are not selected appropriately.

3.3.2 Heterogeneous computing

Examples where heterogeneous simulations on GPUs and CPUs have been employed to model fluid–solid systems can be found in [47–50]. Although different strategies can be used to solve each partition, in general most of them are based on CPUs to solve the fluid and GPUs to solve the particles. Among these examples, those that make use of heterogeneous CPU–GPU clusters are the particular interest, see Figure 3.11(a). In this context, the data is transferred along the computer nodes by a high performance low latency interconnection network, such as Infiniband (IB) and then throughout the PCI-Express (PCIe) topology until reaching the GPUs [51,52]. In that sense, the MPI can be used for the communication among distributed processes. Multiple GPUs using a MPI–based parallelization can rely on either traditional or CUDA-aware MPI implementation. In the case of a traditional MPI implementation, the bunch of data should be transferred via **cudaMemcpy** and **MPI_Send/MPI_Recv** [53, 54]. The **cudaMemcpy** instruction moves the data from the CPUs to the GPUs and vice versa, while **MPI_Send** and **MPI_Recv** are used to send and receive that data among processors.In the particular case of the CUDA particles simulator described above, once all the system has been updated, the positions and velocities of the particles are transferred to PLE++. After this step, the localization is done. The Figure 3.11(b) shows the results obtained in an example were localization was performed by using 16384 particles and 56401 tetrahedron cells. A total of 110 steps were employed. The localization was carried out in each time step. The simulation has been executed in a NVIDA GeForce 320M with 768 CUDA Cores and a processor 1.86 GHz Intel Code 2 Duo, CUDA 4.6 and OpenMPI 1.6.



(a)



(b)

Figure 3.11: (a) Localization procedure for a heterogeneous simulation on GPU and CPU. In this case, the GPU and Alya transfer the data throughout interconnection network and PCI-Express. PLE++ performs the localization and mapping. In the sketch the GPU code uses the first MPI thread while the rest is occupied by the Alya code. (b) Time evolution of the particles and tetrahedron cells involved on the localization.

3.4 Related work

In resent years a great deal of effort has been made to develop coupling tools specialized on the solution of partitioned multi-physics application. Examples of these developments are the libraries MUI (Multiscale Universal Interface) [18], preCICE (Precise Code Interaction Coupling Environment) [17]. and CWIPI (Coupling With Interpolation Parallel Interface) [15, 16]. As PLE++, these libraries provide a communication layer for exchanging information between two or more parallel codes. Of particular interest are these three libraries due to the fact that their developing is continuous, and because they have shown to be appropriate to solve large-scale multi-physics problems.

MUI uses a Smoothed Particle Hydrodynamics (SPH) simulation as an example to perform a strong scalability test [18]. Such example involves two overlapping SPH partitions, each of them containing the same quantity of fluid particles ($\sim 1.4 \times 10^6$). The results show that, by using a concurrent execution of the SPH solvers and allocating the same number of MPI processors to each domain, it is possible to achieve a parallel efficiency of $\sim 80\%$ when the total number of processors range from 2 to 2^8. When the MPI ranks move from 2^8 to 2^{10} the efficiency drops to 40%.

The parallel performance of preCICE is studied via a strong scalability for two coupled configurations. Firstly, a cubic fluid domain is artificially divided into two halves and used to demonstrate that the coupling implementation is able to maintain the trend of a linear strong scaling in the solution of hyperbolic conservation laws. The second coupling configuration consists of a partitioned fluid structure interaction case where the number of processors in the solid partition varies from 3 to 2^7 , while the number or processors dedicated to solve the fluid domain remains fixed at 2^{10} . This is done in order to identify the best relation between the number of processors used for each partition. The study claims that, for this particular case, such relation is achieved when six processors are assigned to the solid partition, in other case the scalability tend to decrease.

Regarding CWIPI, and as part of the OpenPALM software [11], parallel performance studies has been mainly focused on partitioned conjugate heat transfer problems. Exchanges times between solvers as function of the number of cores of the coupled system has been studied in [55], or [16] where the analysis of a coupled combustion chamber with a total of ~ 40 millions of elements shows the impact of imbalanced repartitions of cores among the solvers. In [5] the parallel and staggered coupling schemes are applied to the study of a cooled turbine blade with about 7 millions of elements. This last example claims that existence a relation in the distribution of the processors which encourages the load balance for a parallel coupling scheme exists. Such relation associates the execution times T_s and T_f of the solid and the fluid solvers, respectively, with the number of processors p_f and p_s assigned to each solver, so that $p_s/p_f = T_s/T_f$.

As will be shown later (Chapter 6) the parallel performance studies related to PLE++ lead to comparable results as those obtained by MUI, preCICE and CWIPI. It is due to the fact that they share similar features in their development. Firstly, all of them have been designed to encourage flexibility and scalability. In this sense, each of them have developed its own application programming interface (API) that usually allows to define the *communication, coupling schemes*, and *data mapping* to be used. On the other hand, despite of the fact that the most of them are able to solve the same problems (for instance, of all them are able to solve conjugate heat transfer problems) the main difference between the libraries is the application target.

3.5 Summary and remarks

This chapter presents an overview of the main challenges related to the implementation of a coupling tool with the capability of dealing with partitioned multi-physics systems for extreme scale architectures. Firstly, execution models of MPI and their relation with the two coupling schemes are introduced. After that, the workflow of the coupling tool developed here is described. Then, two application cases are discussed. The first case addresses the parallel performance analysis of a fluid-solid thermal coupling, while the second case describes the implementation of a heterogeneous simulation involving GPU and CPU.

The execution sequence of the parallel and staggered schemes are based on the Multiple-program-multiple-data (MPMD) execution mode. This execution model solves each of the partition solvers involved in the coupling as a Single-program-multiple-data (SPMD) execution mode. A partitioned multi-physics approach based on the MPMD execution mode requires extra communication between processors due to the fact that partitions are independently executed. This execution of the partitions is the source of the parallel coupling problem (transmission and transformation of data between partitions).

The Parallel and Locator Exchange Library++ (PLE++) presented here provides flexibility and scalability and allows the definition of different coupling schemes along with appropriate methods to deal with the parallel coupling problem. PLE++ is a C++ environmental library with the capability of allowing the communication between parallel applications written in C/C++, Fortran or Python. The workflow of PLE++ overcomes the parallel coupling problem by using three main stages. Firstly, the set of MPI communicators to be used by each partition, localization algorithm as a peer-to-peer communication layout, and the exchange of data.

Finally, the application cases introduced here show that the PLE++ library provides scalability and flexibility. The trace of a staggered coupling approach shows that no overheads due to the exchange of information are introduced into this approach. The second case proves that PLE++ can be used in heterogeneous simulations involving GPUs and CPUs. In this case, a CUDA particles simulator is coupled to the HPC-Alya code. Positions and velocities of the particles are then transferred from the GPU to the CPU through the PLE++.

4

Contact of deformable bodies

This chapter addresses the development of a parallel algorithm to deal with the problem of *contact between deformable bodies*. In general, the solution of a contact problem involves two main stages: contact resolution and contact search. Contact resolution refers to how constrains arising from contact are enforced, while contact search identifies regions on the surface of each body where contact takes place. In this work, an iterative coupling approach is used for the *contact resolution* of the frictionless interaction between two bodies, whereas the *contact search* is performed by the PLE++ library.

One of the main contributions of this chapter is to show how the parallel localization procedure implemented in the PLE++ library is crucial for the development of the contact algorithm described in this thesis, for details see [31]. It is worth noting that this development has been performed in collaboration with a specialist in computational mechanics, Matías Ignacio Rivero. The *novel* algorithm here described was developed taking profit of his knowledge in the physics of contact as well as of the advantages presented by localization procedure. Details are given as follows. Firstly, basic physical principles behind the contact of deformable bodies are revised in Section 4.1. The fundaments of computational mechanics applied to deal with the contact resolution are briefly discussed in Section 4.2. The iterative Dirichlet-Neumann approach used in this thesis for the solution of frictionless contact problems is described in Section 4.3. Case studies are given in Section 4.4, including a validation case, and a practical application of industrial interest.

4.1 Physical background

This section introduces basic physical principles behind the interaction between two bodies in contact: (1) non-penetration constrain, and (2) equilibrium of forces between bodies. These physical principles are introduced using two examples. Firstly, a simple case describing the modelling of *elastic* particles is presented. Afterwards, the interaction between deformable and rigid bodies is used to show how the contact constrains define the shape that a deformable body must reach.

4.1.1 Elastic particles

The interaction between non-spherical rigid particles is modelled by considering each of them as *elastic*. [56]. The main characteristic of this approach is that the contact forces are computed based on the magnitude of overlap. When two particles are in contact, it is possible to assume that a repulsive force $\mathbf{F}_{ab}$ appears as result of the overlapping δ_n between them. The magnitude of this force can be obtained as proportional to the overlap and a spring stiffness constant k

$$\mathbf{F}_{ab} = -k_n \delta_n \mathbf{n}_{ab} \tag{4.1.1}$$

where $\mathbf{n}_{ab} = (\mathbf{r}_b - \mathbf{r}_a)/|\mathbf{r}_b - \mathbf{r}_a|$ stands for the unit vector between the center of the particles $\mathbf{r}_a$ and $\mathbf{r}_b$. More sophisticated models can even include a damping proportional to the normal relative velocity vector between the particles, and tangential forces [35].

Without loss of generality, lets suppose the case of two identical spherical particles of radius R_a, see Figure 4.1. The repulsive force $\mathbf{F}_{ab}$ appears only when the distance r_{ab} between their centers is smaller than $2R_a$, so that

$$\mathbf{F}_{ab} = \begin{cases} -k\delta_n \mathbf{n}_{ab} & \text{for } r_{ab} \leq 2R_a \\ 0 & \text{for } r_{ab} > 2R_a \end{cases} \tag{4.1.2}$$

In order to maintain the overlapping parameter as a *non-negative number*, it must be calculated as $\delta_n = 2R_a - r_{ab}$ instead of $r_{ab} = 2R_a + \delta_n$. Considering the above,

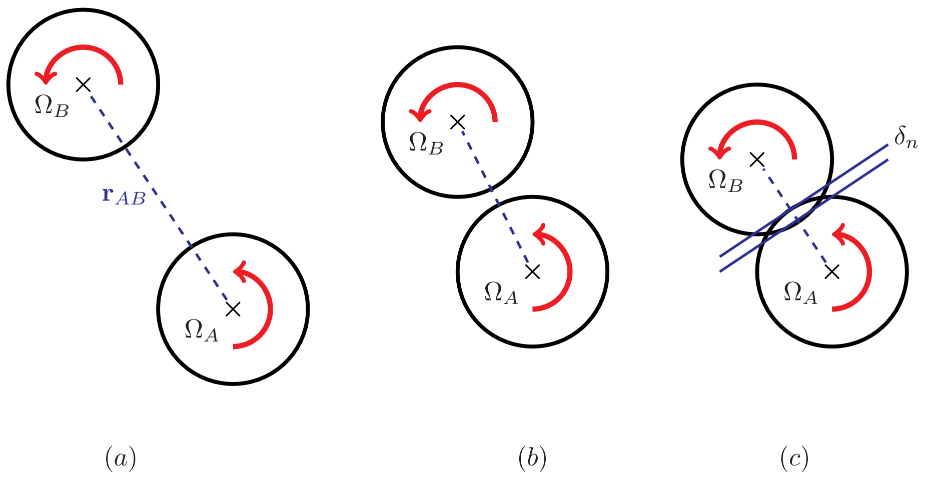


Figure 4.1: Soft-sphere model of a particle-particle interaction.

the Equation 4.1.1 can be rewritten as

$$\mathbf{F}_{ab} = \begin{cases} -k\delta_n \mathbf{n}_{ab} & \delta_n > 0 \\ 0 & \delta_n = 0 \\ 0 & \delta_n < 0 \end{cases} \tag{4.1.3}$$

i.e., when the overlapping is positive ($\delta_n \geq 0$), a force ($\mathbf{F}_{ab} \leq 0$) opposed to the movement is applied, otherwise ($\delta_n < 0$) the force is always zero.

In more general cases of the discrete element method, for instance, non-spherical particles, the magnitude of the force is calculated as proportional to the overlapping area or volume between these particles [57], see Figure 4.2. In these cases, Equation 4.1.2 is even valid if an appropriate stiffness constant is used.

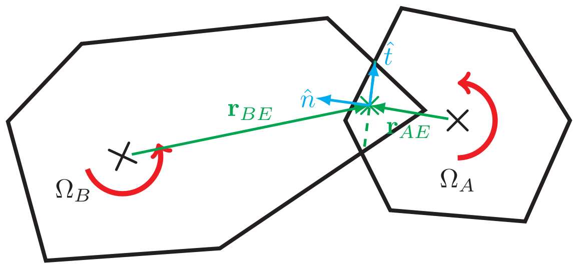


Figure 4.2: Soft-nonspherical model of a particle-particle interaction. Some of the variables are need to compute the relative velocity v_{AB} of the particles at the contact point: $\mathbf{r}_A$ and $\mathbf{r}_B$ are the vectors from the centroids C_1 and C_2 of the particles to the force point P.

4.1.2 Unilateral contact

Like the soft particles, the contact forces for deformable bodies are also computed based on the overlap between them. However, in this case, the value of the stiffness is not a constant, but a distribution of forces depending on the displacements across the contact interface.

Figure 4.3 depicts the interaction between a deformable body and a static undeformable body. Once the bodies overlap, the *non-penetration constrain* defines the shape that the deformable body must reach across its contact interface. Thus, in order to achieve this shape, the contact interface of the deformable body must be displaced. The magnitude of these displacements can be calculated similarly as the overlapping parameter δ_n. A first approach is to use the difference between the initial configurations of the bodies as displacements. As result of enforcing these displacements, the distribution of the forces inside the deformable body must change. These forces are transmitted from one body to another by a *compressible* force, normal to the contact interface, and a friction force acting tangentially. It is worth noting that, the action-reaction principle requires that the normal forces acting through the contacting surface are equal and opposite.

4.2 Formulation of multi-body contact problem

The previous section describes basic physical principles behind the contact of deformable bodies. This section briefly introduces the fundamentals of the computational mechanics applied to model contact problems, see [56, 58–64] for details. The modelling presented here involves the use of a master-slave approach, which neglects the friction, and where the conservation of momentum is solved by the finite element method, see Figure 4.4. The partitioned multi-physics approach whereby the contact conditions are enforced is described later in Section 4.3.1.

4.2.1 Constrains

Non-penetration

The overlapping parameter introduced above is generalized here as a *gap vector* $\mathbf{g}$. Given two bodies, the gap vector measures the distance from a point $\mathbf{r}$ on the surface $\partial\Omega$ of a *master* body to a point $\boldsymbol{\rho}$ allocated on the surface S of a *slave* body

$$\mathbf{g}(\boldsymbol{\rho} \in S, \partial\Omega) = \boldsymbol{\rho} - \mathbf{r} \tag{4.2.1}$$

The value of the *penetration* g_n can be defined as the *closest* distance to the *master* body, and, therefore, in the direction of the normal $\boldsymbol{\nu}$ of the master surface [58]

$$g_n(\boldsymbol{\rho} \in S, \partial\Omega) = (\boldsymbol{\rho} - \mathbf{r}) \cdot \boldsymbol{\nu}(\mathbf{r}) \tag{4.2.2}$$

Thus, the mathematical *condition for non-penetration* is given as $g_N \geq 0$.

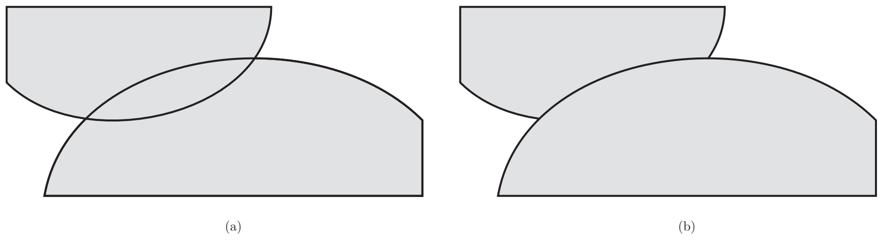


Figure 4.3: (a) Penetration. (b) Deformation.

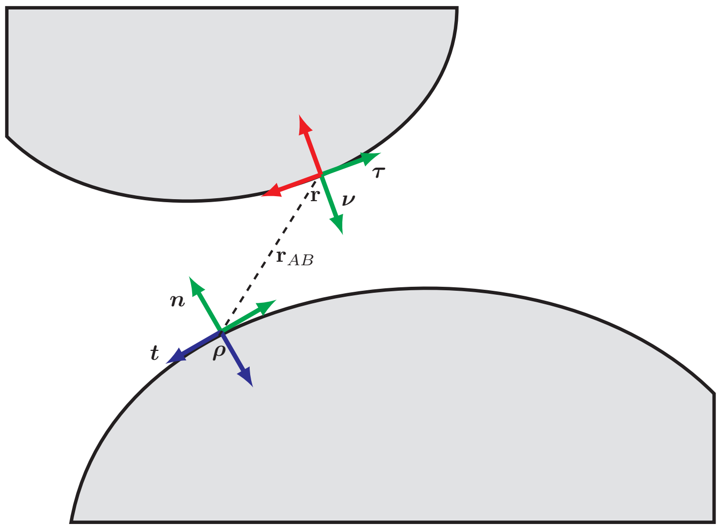


Figure 4.4: Master-slave approach. Gap vector $\mathbf{g}$ is measured from surface $\delta\Omega$ of the master (top) body, to the surface S of the slave (bottom) body. Vectors $(\boldsymbol{\nu}, \boldsymbol{\tau})$ and $(\boldsymbol{n}, \boldsymbol{t})$ stand for normal and tangential directions on the surface of each body. Components of the stresses are given as $(\boldsymbol{n}\sigma_n, \boldsymbol{\sigma}_t)$.

Equilibrium

Likewise it occurs for elastic particles and unilateral contact, reacting forces appear as result of the penetration g_n. These forces are divided into normal and tangential. The normal force $\mathbf{p}$, generally compressible, acts along a common normal $\mathbf{n}$ to the contact surface, while the tangential force $\mathbf{q}$ results in a friction force acts on the tangential plane.

In the case of deformable bodies, normal and tangential forces are transmitted from one surface to another by contact tractions: *pressure* (normal traction) and *friction* (shear traction). In turn, these tractions are used to calculate the resulting deformations $\mathbf{d}$, and the internal distribution of stresses $\boldsymbol{\sigma}$.

The action-reaction principle requires that the normal forces acting through a *frictionless* contact interface are equal and opposite at any moment, i.e., in the contact point $\mathbf{r} \in \partial\Omega$ the normal force $\mathbf{p}$ acts on both interfaces S and $\partial\Omega$, so that

$$\mathbf{p} = \mathbf{p}_S + \mathbf{p}_{\partial\Omega} = 0, \tag{4.2.3}$$

leads to the *equilibrium of stresses* throughout the contact interfaces

$$\mathbf{n}_S \cdot \boldsymbol{\sigma}_S \cdot \mathbf{n}_S + \mathbf{n}_{\partial\Omega} \cdot \boldsymbol{\sigma}_{\partial\Omega} \cdot \mathbf{n}_{\partial\Omega} = 0. \tag{4.2.4}$$

In order to find the pressure distribution at any point of the contact surface of a given profile, the solution of an integral equation for the pressure is required [62]. This is discussed in the next section.

4.2.2 Mathematical modelling

Following the procedure presented in Section 2.1.1, the transmission conditions for the contact problem can be found once the contact constrains are introduced in the weak form of the equation of linear momentum.

Solid mechanics problem

The *conservation of linear momentum* is the base of solid mechanics. It is a generalization of the Newton's law of motion to a deformable solid, for details see [65,66]. The equation is given as

$$\mathcal{L}\,\mathbf{d} = \rho\frac{\partial^2 \mathbf{d}}{\partial t^2} - \nabla\cdot\boldsymbol{\sigma} \quad = \quad \mathbf{f} \quad \text{in} \quad \Omega \in \mathbb{R}^3 \tag{4.2.5}$$

where ρ, $\mathbf{d} = \mathbf{x} - \mathbf{X}$, $\boldsymbol{\sigma}$ and $\mathbf{f}$, stand for material density, displacement field, stress field, and volume force density for a solid partition Ω, respectively. Stress boundary conditions are given by the traction condition $\boldsymbol{\sigma}\cdot\mathbf{n} = \mathbf{t}_N$, while displacement boundary conditions are given by $\mathbf{d} = \mathbf{d}_D$, where $\mathbf{d} : \Omega\cup\Gamma\times[0,T] \rightarrow \mathbb{R}^3, T > 0$. Additionally, initial conditions for displacements and velocities are required $\mathbf{d}(.,0) = \mathbf{d}_0$ in Ω, $\partial\mathbf{d}(.,0)/\partial t = \mathbf{d}_0$, respectively.

Weak form

The weak form (in the current configuration) of Equation 4.2.5 reads:

$$\text{find } \mathbf{d} \in \mathcal{U}\text{:} \quad \mathcal{A}(\mathbf{d},\boldsymbol{\eta}) \quad = \quad F(\boldsymbol{\eta}) \quad \forall\boldsymbol{\eta} \in \mathcal{V} \tag{4.2.6}$$

where

$$\begin{cases} \mathcal{A}(\mathbf{d},\boldsymbol{\eta}) & \equiv & \int_\Omega \left(-\rho\partial\mathbf{d}/\partial t\cdot\boldsymbol{\eta} + \boldsymbol{\sigma}\cdot\nabla\boldsymbol{\eta}\right) d\Omega \\ F(\boldsymbol{\eta}) & \equiv & \int_\Omega \mathbf{f}\cdot\boldsymbol{\eta}\; d\Omega + \int_{\partial\Omega}[\mathbf{n}\cdot\boldsymbol{\sigma}]\cdot\boldsymbol{\eta}\; d\Gamma \\ \mathcal{U} & \equiv & \{\mathbf{d} \in H^1(\Omega) : \;\; \mathbf{d} = \mathbf{d}_D \text{ on } D\,\} \\ \mathcal{V} & \equiv & \{\boldsymbol{\eta} \in H^1(\Omega) : \;\; \boldsymbol{\eta} = 0 \text{ on } D\,\} \end{cases}$$

By using *arbitrary test displacements* $\delta\mathbf{d} = \delta(\mathbf{x} - \mathbf{X}) = \delta\mathbf{x}$ instead of test functions $\boldsymbol{\eta}$, and applaying appropriate boundary conditions, Equation 5.2.12 can be written as

$$\begin{aligned} -\int_\Omega \rho\partial\mathbf{d}/\partial t\cdot\delta\mathbf{d}\; d\Omega + \int_\Omega \boldsymbol{\sigma}\cdot\nabla\delta\mathbf{d}\; d\Omega \quad - & \int_{\partial\Omega} \mathbf{t}_N\cdot\delta\mathbf{d}\; d\Gamma \\ + & \int_\Omega \mathbf{f}\cdot\delta\mathbf{d}\; d\Omega = 0 \end{aligned} \tag{4.2.7}$$

Now, in order to find appropriate transmission conditions (the boundary conditions to be used in the coupling) for the contact problem, the constrains must be introduced to the weak form of the linear momentum equation. Details are given below.

Contact constrains At any point across the contact interface, the displacements depend on the contact tractions and vice versa. The mechanics of contact introduces the non-penetration constrain (Equation 4.2.2) inside the weak form of the conservation of linear momentum (Equation 5.2.12), in order to find the distribution of displacements and tractions throughout the contact interface.

Following the procedure presented in Equation 2.1.4 (Section 2.1.1), the integral through the boundaries can be divided into three parts

$$\begin{aligned} \int_{\partial B} \mathbf{n} \cdot \boldsymbol{\sigma} \cdot \delta \mathbf{d} \, d\Gamma &= \cancel{\int_D \mathbf{n} \cdot \boldsymbol{\sigma} \cdot \delta \mathbf{d} \, d\Gamma} + \int_N \mathbf{t}_N \cdot \delta \mathbf{d} d\Gamma \\ &+ \int_{C^1+C^2} \mathbf{n} \cdot \boldsymbol{\sigma} \cdot \delta \mathbf{d} \, d\Gamma \end{aligned} \tag{4.2.8}$$

The contribution of the Dirichlet boundary D is zero. The integral on the Neumann boundary N is different from zero when traction conditions $\mathbf{t}_N$ are given. The last part corresponds to the contact interfaces C^1 and C^2 of each partition. By splitting the contribution of this integral into two parts, and along with the equilibrium of stresses, the gap vector $\mathbf{g}$ can be introduced

$$\begin{aligned} \int_{C^1+C^2} \left[\mathbf{n} \cdot \boldsymbol{\sigma} \cdot \delta \mathbf{d} \right] d\Gamma &= \int_{C^1} \mathbf{n}^1 \cdot \boldsymbol{\sigma}^1 \cdot \delta \mathbf{d}^1 \, d\Gamma^1 + \int_{C^2} \mathbf{n}^2 \cdot \boldsymbol{\sigma}^2 \cdot \delta \mathbf{d}^2 \, d\Gamma^2 \\ &= \int_{C^1} \mathbf{n}^1 \cdot \boldsymbol{\sigma}^1 \cdot \delta(\mathbf{d}^1 - \mathbf{d}^2) \, d\Gamma^1 \\ &= \int_{C^1} (\mathbf{n}^1 \sigma^1_n + \boldsymbol{\sigma}^1_t) \cdot \delta(\mathbf{g}) \, d\Gamma^1 \end{aligned} \tag{4.2.9}$$

where the last relation results of projecting the traction $\mathbf{n}^1 \cdot \boldsymbol{\sigma}^1$ as normal and tangential components to the master surface C^1.

Frictionless contact Equation 4.2.9 integrates the conservation of linear momentum and the contact constrains. In the particular case of contact *without* friction, it is possible to show that $(\mathbf{n}^1\sigma_n + \boldsymbol{\sigma}_t) \cdot \delta(\mathbf{d}^1 - \mathbf{d}^2)$ reduces to $\sigma^1_n \, \delta g_n$, so that

$$\int_{C^1+C^2} \left[\mathbf{n} \cdot \boldsymbol{\sigma} \cdot \delta \mathbf{d} \right] d\Gamma = \int_{C^1} \sigma^1_n \, \delta g_n \, d\Gamma^1 \tag{4.2.10}$$

where $\delta g_n = \mathbf{n}^1 \cdot (\delta \mathbf{d}^1 - \delta \mathbf{d}^2)$ is the first variation of the normal gap g_n, and $\mathbf{n}^1 = \mathbf{d}^1 - \mathbf{d}^2$ represents the closest distance from a slave node to the master surface. Thus, by using Equations 5.2.13 and 4.2.10, the weak form of the frictionless contact problem can be expressed as

$$\begin{aligned} -\int_\Omega \rho \partial \mathbf{d}/\partial t \cdot \delta \mathbf{d} \, d\Omega &+ \int_\Omega \boldsymbol{\sigma} \cdot \nabla \delta \mathbf{d} \, d\Omega - \int_N \mathbf{t}_N \cdot \delta \mathbf{d} \, d\Gamma \\ &+ \int_{C^1} \underbrace{\sigma_n \, \delta g_n}_{\psi} \, d\Gamma + \int_\Omega \mathbf{b} \cdot \delta \mathbf{d} \, d\Omega = 0. \end{aligned} \tag{4.2.11}$$

Transmission conditions

From the weak form of the equation of linear momentum (Equation 4.2.11), transmission conditions can be established. In Equation 4.2.9, the integrals corresponding to the contact interfaces are combined. The resulting integral is simplified for the case of frictionless contact in Equation 4.2.10. For this particular case, the problem is to find the normal traction σ_n allowing the nullification of the gap g_n, so that when contact constrains are satisfied, the integral on C^1 tends to zero. In this work, the coupling approach uses the gap g_n to enforce Dirichlet-type boundary conditions in one body (partition), while Neumann conditions $\mathbf{n} \cdot \boldsymbol{\sigma} = \sigma_n$ are imposed on the other body. The next section briefly describes the parallel algorithm of contact introduced here. Detail are widely described in the Ph.D. thesis [31].

4.2.3 Contact resolution: Treatment of contact constrains

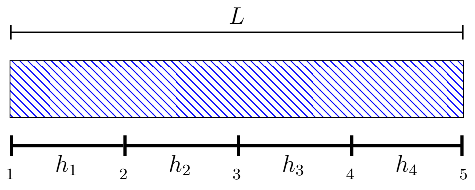
In this section, two contact models used to enforce the contact constrains are briefly discussed. The objective is to show the relation that these models have with the iterative Dirichlet-Neumann approach described in the next section. It is worth noting that the arguments given in this section introduce a novel form to understand how the methods used for the treatment of the contact constrains are related. The details about each method can be seen in [61, 65].

Lagrange multipliers and penalty methods are numerical models widely applied for the solution of contact problems [61, 65]. Both methods are used to introduce the term given by Equation 4.2.10 into the discretization of Equation 5.2.13, so that contact constrains can be modelled. In [61], a system consisting of two bars separated by a gap g is used to illustrate how the Lagrange multiplier method can be applied when contact takes place. Here, this example is modified in order to show how the iterative approach described in Section 2.2, the Lagrange multiplier method, and the penalty method are related.

Figure 4.5 (a) shows an axially loaded bar whose solution is found by the finite element method. The resulting system of equations $\mathbf{K}\ \mathbf{d} = \mathbf{f}$ is shown. Four discrete elements of length h_i and five nodes are used in the discretization. Now, lets suppose that a discrete element is removed (arbitrarily h_3), see Figure 4.5 (b). The resulting configuration consists of two bars separated by a gap g. Another consequence is that the system of equations must be modified. The new system is now composed by two parts $\mathbf{K}_{iA}\mathbf{d}_A + \mathbf{K}_{iB}\mathbf{d}_B = \mathbf{f}_i$ each of them corresponding to bar i. Fundamentally, $\mathbf{K}_{ii}\mathbf{d}_i$ stands for what takes place *inside* each bar i, while off-diagonal sub-matrices $\mathbf{K}_{ij}$ stand for *interactions* between bar i and j. In the simplest case, when bars are insulated to each other, off-diagonal sub-matrices are zero, so that systems $\mathbf{K}_{ii}\mathbf{d}_i = \mathbf{f}_i$ can be solved independently. On the other hand, when bars are in contact, the term $\mathbf{K}_{ij}\mathbf{d}_j$ needs to be considered. Three situations are now analysed.

In Figure 4.5 (b) sub-matrices $\mathbf{K}_{ij}$ are explicitly shown for two bars separated by a gap g. Bar Ω_A is composed of two discrete elements and three nodes, while bar Ω_B is composed of one element and two nodes. Gap between them is of length h_3. Because these two bars arise as consequence of removing the element h_3 from the original bar it can be assumed that the sub-matrices $\mathbf{K}_{ij}$ can be found removing appropriate terms from the original matrix $\mathbf{K}$. The symbol ξ_3 stands for the effect that the removed element h_3 could cause. Two cases are evident. In the original system $\mathbf{K}$, $\xi_3 = AE/h_3$. When bars are insulated to each other, $\xi_3 = 0$. From these limits, it can be concluded that the value of ξ_3 must be determined in order to perform the contact between these two bars. This is the essential concept behind the penalty method, see [61, 65]. The Lagrange multiplier method can also be inferred from this example. For this case, the products inside all sub-matrices $\mathbf{K}_{ij}\mathbf{d}_j$ are explicitly performed, so that terms $\pm\xi_3 g$ appear for the nodes being in contact. The Lagrange multiplier method considers these terms as extra unknowns, so that a new

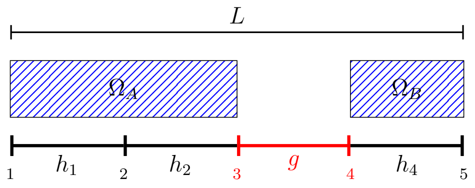
(a)



[I]

$$\begin{bmatrix} AE/h_1 & -AE/h_1 & 0 & 0 & 0 \\ -AE/h_1 & AE/h_1+AE/h_2 & -AE/h_2 & 0 & 0 \\ 0 & -AE/h_2 & AE/h_2+AE/h_3 & -AE/h_3 & 0 \\ 0 & 0 & -AE/h_3 & AE/h_3+AE/h_4 & -AE/h_4 \\ 0 & 0 & 0 & -AE/h_4 & AE/h_4 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{bmatrix} \longleftarrow \mathbf{Kd} = \mathbf{f}$$

(b)



[II]

$$\left[\begin{array}{ccc|cc} AE/h_1 & -AE/h_1 & 0 & 0 & 0 \\ -AE/h_1 & AE/h_1+AE/h_2 & -AE/h_2 & 0 & 0 \\ 0 & -AE/h_2 & AE/h_2+\xi_3 & -\xi_3 & 0 \\ \hline 0 & 0 & -\xi_3 & AE/h_4+\xi_3 & -AE/h_4 \\ 0 & 0 & 0 & -AE/h_4 & AE/h_4 \end{array}\right] \left[\begin{array}{c} d_1 \\ d_2 \\ d_3 \\ \hline d_4 \\ d_5 \end{array}\right] = \left[\begin{array}{c} f_1 \\ f_2 \\ f_3 \\ \hline f_4 \\ f_5 \end{array}\right] \longleftarrow \left[\begin{array}{c|c} \mathbf{K}_{AA} & \mathbf{K}_{AB} \\ \hline \mathbf{K}_{BA} & \mathbf{K}_{BB} \end{array}\right] \left[\begin{array}{c} \mathbf{d}_A \\ \hline \mathbf{d}_B \end{array}\right] = \left[\begin{array}{c} \mathbf{f}_A \\ \hline \mathbf{f}_B \end{array}\right]$$

[III]

$$\left[\begin{array}{ccc|ccc} AE/h_1 & -AE/h_1 & 0 & 0 & 0 & 0 \\ -AE/h_1 & AE/h_1+AE/h_2 & -AE/h_2 & 0 & 0 & 0 \\ 0 & -AE/h_2 & AE/h_2 & 0 & 0 & \pm 1 \\ \hline 0 & 0 & 0 & AE/h_4 & -AE/h_4 & \mp 1 \\ 0 & 0 & 0 & -AE/h_4 & AE/h_4 & 0 \\ 0 & 0 & \mp 1 & \pm 1 & 0 & 0 \end{array}\right] \left[\begin{array}{c} d_1 \\ d_2 \\ d_3 \\ \hline d_4 \\ d_5 \\ \xi_3 g \end{array}\right] = \left[\begin{array}{c} f_1 \\ f_2 \\ f_3 \\ \hline f_4 \\ f_5 \\ \pm g \end{array}\right]$$

Figure 4.5: Axially loaded bar example. (a) Discretization of equilibrium equation by using four discrete elements of length h_i and five nodes. (b) Discretization under supposition of that the element h_3 is removed so that a gap g arises. The bar is supposed of length L and constant cross-section A. By using Hooke's law $\sigma_{11} = E\varepsilon_{11}$ as constitutive model, Equation 4.2.5 reduces to $E\partial^2 d/\partial x^2 + f/A = 0$ in equilibrium. In the first case, the finite element discretization results on the system of equations $\mathbf{K}\ \mathbf{d} = \mathbf{f}$. For the second case, the matrices show the treatment that three different methods give to the contact constrains.

system of equations is established. For instance, rows three and four in matrix II yield to $-AEd_2/h_2 + AEd_3/h_2 + \xi_3 g = f_3$, and $AEd_4/h_4 - AEd_5/h_4 - \xi_3 g = f_4$, with $g = d_3 - d_4$, giving rise to row and column six in matrix III.

In the Iterative Dirichlet-Neumann algorithm described in the next section, the systems $\mathbf{K}_{ii}\mathbf{d}_i = \mathbf{f}_i$ are solved independently, off-diagonal sub-matrices $\mathbf{K}_{ij}$ are assumed as zero, while their effects are modelled through boundary special boundary conditions on the contact interface.

4.3 Methodology

This section summarizes the *contact resolution* proposed in [31] for the frictionless interaction between two deformable bodies. It is worth noting that the methodology presented here has been developed as a collaborative work where the PLE++ library which is crucial for the parallel contact search (localization) as well as for the transmission and transformation of the data (exchange). See Chapter 3 for details of localization and exchange.

The contact resolution presented here is based on two works introduced for the numerical solution of contact problems: the Method of partial Dirichlet Neumann boundary conditions (PDN), and the Contact algorithm of Dirichlet-Neumann type (CDN) [63, 67]. The first case is focused on the contact between a rigid surface and a deformable body. In the second case, the contact between linear elastic bodies is addressed. Summarizing, the method described here avoids the use of Lagrange multipliers widely applied in contact mechanics [59, 62]. One of the main drawbacks of Lagrange multipliers is the fact that the matrix system to be solved changes as the contact evolves. By contrast, the iterative method described here overcomes this drawback by enforcing the contact constrains through the use of Dirichlet and Neumann boundary conditions. Details are described in the rest of this section.

4.3.1 Iterative Dirichlet-Neumann algorithm

As shown in Section 2.1.1, the variational form of a given equation can be used to define the appropriate transmission conditions required by an iterative domain decomposition method. In Equation 4.2.11, the integral through the contact interface incorporates the concepts introduced in Section 4.1.2. Firstly, non-penetration constrain imposes the shape that a deformable body can reach. This is done by enforcing Dirichlet boundary conditions that ensure $\mathbf{n}^1 \cdot (\mathbf{d}^1 - \mathbf{d}^2) = 0$. Once Equation 4.2.5 is solved along with Dirichlet conditions, the normal traction σ_n to be applied on the other body can be calculated from the stress field. This fulfils the action-reaction principle $\mathbf{n}^1 \cdot \boldsymbol{\sigma}^1 \cdot \mathbf{n}^1 + \mathbf{n}^2 \cdot \boldsymbol{\sigma}^2 \cdot \mathbf{n}^2 = 0$ between these bodies. Thus, a contact problem can be solved by an iterative domain decomposition method where Dirichlet and Neumann boundary conditions can be applied sequentially. Algorithm 4.7 along with Figure 4.6 depicts this iterative process introduced here in detail.

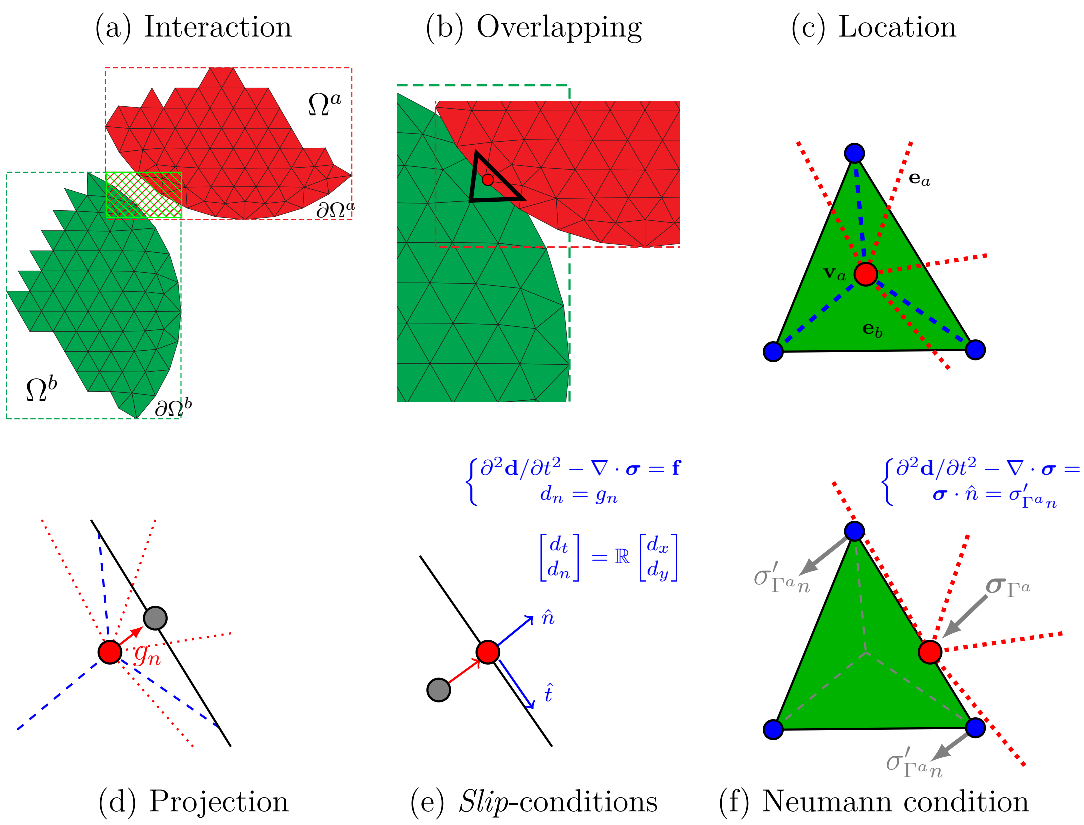

(a) Interaction
(b) Overlapping
(c) Location
Ω^a
$\partial\Omega^a$
Ω^b
$\partial\Omega^b$
$\mathbf{e}_a$
$\mathbf{v}_a$
$\mathbf{e}_b$
$\begin{cases}\partial^2\mathbf{d}/\partial t^2 - \nabla\cdot\boldsymbol{\sigma} = \mathbf{f} \\ d_n = g_n\end{cases}$
$\begin{bmatrix}d_t\\d_n\end{bmatrix} = \mathbb{R}\begin{bmatrix}d_x\\d_y\end{bmatrix}$
$\begin{cases}\partial^2\mathbf{d}/\partial t^2 - \nabla\cdot\boldsymbol{\sigma} = \mathbf{f} \\ \boldsymbol{\sigma}\cdot\hat{n} = \sigma'_{\Gamma^a n}\end{cases}$
g_n
$\hat{n}$
$\hat{t}$
$\sigma'_{\Gamma^a n}$
$\boldsymbol{\sigma}_{\Gamma^a}$
$\sigma'_{\Gamma^a n}$
(d) Projection
(e) *Slip*-conditions
(f) Neumann condition

Figure 4.6: Parallel algorithm for contact (search and resolution). Slip and Neumann conditions correspond to the boundary conditions enforced through the coupling interface Γ for steps (2) and (3) of Algorithm 4.7. Starting guess stands for the update position of Ω^a.

Initially, the iterative method supposes that the bodies Ω^a and Ω^b are far from each other. As the distance between them decreases, the localization process described in Section 3.2 takes place. When the bounding boxes around each body overlap, the bodies *can* interact, see Figure 4.6(a). The interaction *starts* as soon as the bodies overlap (Figure 4.6(b)). This overlap consists of finding the set of vertices $\mathbb{V}_a$ on the surface $\partial\Omega^a$ of the deformable body Ω^a, which are *localized* inside of any element $\mathbb{E}_b$ on the surface $\partial\Omega^b$ of Ω^b (Figure 4.6(c)). Once the localization algorithm 3.7 has a vertex $\mathbf{v}_a \in \mathbb{V}_a$ associated inside an element $\mathbf{e}_b \in \mathbb{E}_b$, then the gap g_n can be calculated as the distance from the vertex $\mathbf{v}_a$ to the boundary face of $\mathbf{e}_b$. Because this boundary face can be approximated as a 3D plane passing by the boundary vertices of $\mathbf{e}_b$, then the distance from this plane to $\mathbf{v}_a$ defines g_n, which in turn, defines the displacement to be enforced as Dirichlet condition in Ω^a (Figure 4.6(d)). This boundary condition must be enforced in such a way that the vertex $\mathbf{v}_a$ can only move in parallel to the boundary plane of $\mathbf{e}_b$. This is similar to impose

```
Lets $\mathbf{d}_0^{(0)}$ denote a starting guess
  1. For $k = 0, 1, ...,$ until convergence do:
  2. Solve for $\mathbf{d}^{(k)} \in \Omega^a$ as follows:
```

$$\begin{cases} \mathcal{L}\,\mathbf{d}^{(k)} &= \mathbf{f} & \text{in } \Omega^a \subset \mathbb{R}^d \\ \boldsymbol{\sigma}^{(k)} \cdot \mathbf{n} &= \mathbf{t}_N & \text{on } \partial\Omega^a_N \\ \mathbf{d}^{(k)} &= \mathbf{d}_D & \text{on } \partial\Omega^a_D \\ \mathbf{d}^{(k)} &= \mathbf{d}_0^{(k)} & \text{on } \Gamma^a_D \end{cases} \tag{4.3.1}$$

```
  3. Solve for $\mathbf{d}^{(k)} \in \Omega^b$ as follows:
```

$$\begin{cases} \mathcal{L}\,\mathbf{d}^{(k)} &= \mathbf{f} & \text{in } \Omega^b \subset \mathbb{R}^d \\ \boldsymbol{\sigma}^{(k)} \cdot \mathbf{n} &= \mathbf{t}_N & \text{on } \partial\Omega^b_N \\ \mathbf{d}^{(k)} &= \mathbf{d}_D & \text{on } \partial\Omega^b_D \\ \boldsymbol{\sigma}^{(k)} \cdot \mathbf{n} &= \boldsymbol{\sigma}^{(k)}_{\Gamma^a n} & \text{on } \Gamma^b_N \end{cases} \tag{4.3.2}$$

```
  4. Update:
```

$$\mathbf{d}_0^{(k+1)} = \mathbf{d}_0^{(k)} + \theta(\mathbf{d}_s^{(k)} - \mathbf{d}_f^{(k)}) \qquad \text{on } \Gamma \tag{4.3.3}$$

```
  5. Endfor
Output: $(\mathbf{d}^{(k)} \in \Omega^a, \mathbf{d}^{(k)} \in \Omega^b)$
```

Figure 4.7: Iterative Dirichlet-Neumann algorithm for the contact resolution

slip boundary conditions to a fluid (Figure 4.6(e)). The displacement $\mathbf{d} = [d_x, d_y]^T$ to be imposed is divided into normal d_n and tangential d_t components respect to the boundary plane. The boundary conditions are enforced so that the normal component is maintained fixed, while any restriction is imposed in the tangential components. As result of enforcing this slip condition to Equation 4.2.5, the stress field $\boldsymbol{\sigma}_b$ in Ω^b is updated through the contact interface (Figure 4.6(f)). If an unilateral contact is considered, the process is repeated until the desired convergence is achieved. In other cases, the stress field $\boldsymbol{\sigma}_a$ is interpolated from the element $\mathbf{e}_a$ in Ω_a to the vertices $\mathbb{V}_b$ located inside Ω^b. It is worth noting that localization and exchange are need here, due to the fact that Ω^a has changed. Finally, for a frictionless contact case, the normal stress $\sigma'_{\Gamma^a n}$ is enforced as Neumann condition in Ω^b. The entire process can be repeated until the desired convergence for each case is achieved.

So far, after introducing the fundamentals related to computational mechanics, the iterative Dirichlet-Neumann algorithm for contact has been described. This algorithm results from introducing the physics enforced by the contact constrains into the weak form of the frictionless contact problem. As part of this algorithm, the contact interface must be identified. Additionally to this identification, the exchange of transmission conditions (displacement and stresses) must be performed since it is assumed that each body is independently solved. In this work, identification and exchange are performed by the PLE++ library described in Section 3.2. Now, the rest of the chapter is focused on establishing the validity of the ideas developed above, as well as describing some applications currently under development.

4.4 Case studies

The aim of this section is to show the validity of the approach presented here. Two relevant cases are presented. The first case corresponds to the contact problem, while the second case corresponds to a problem of impact. For all cases, the computational solid mechanics problem is solved using standard Galerkin method for a large deformation framework using a generalized Newmark time integration scheme. The details related to this framework developed in a total Lagrangian formulation, can be found in [68].

4.4.1 Hertzian contact: sphere on a rigid plate

An elastic ball contacting with a rigid plane is considered here, see Figure 4.8. An upwards displacement δ is applied to the rigid plane while the topmost node n_f of the sphere is fixed. The contact interaction is produced at the bottom part of the sphere and is assumed to be frictionless. Additionally to the problem setup, the numerical solution achieved is shown.

Analytical solutions for the contact traction distribution are well-known from Hertzian elastic contact problems [69]. This particular problem is characterized by the contact radius a, and the maximum normal contact traction $P_{\max}$, which are given as

$$a^3 = \frac{3Fd}{8}\frac{1-\nu^2}{E} \tag{4.4.1}$$

$$P_{\max} = \frac{3F}{2\pi a^2} \tag{4.4.2}$$

where F is the reaction force at the fixed node and d is the diameter of the sphere.

Figure 4.9(a) shows the normal stress solution for an eighth of the deformed sphere. Figure 4.9(b) shows the reacting force F at node n_f as function of the mesh size. The starting point for the resolution of this problem is a reference mesh of

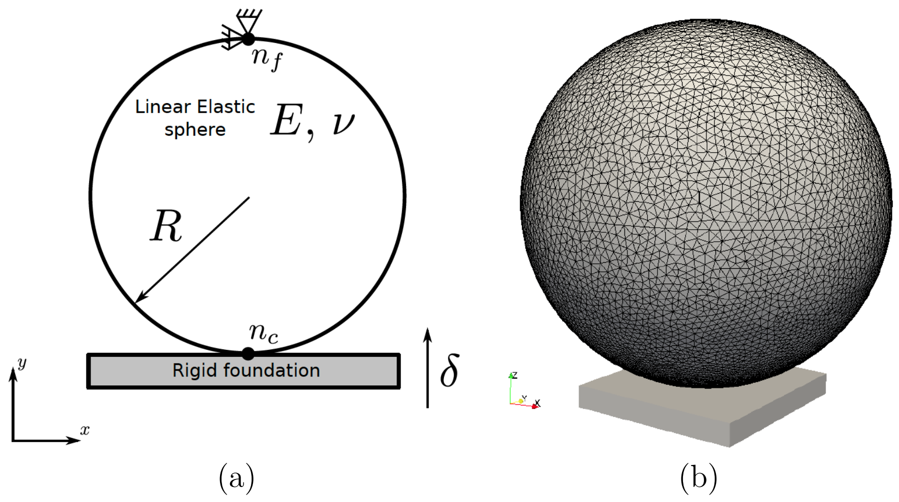


Figure 4.8: 3D Hertz contact problem. (a) problem setup. (b) mesh. Sphere of radius $R = 8m$ and material properties $E = 200N/m^2$ and $\nu = 0.32$. The top most node of the sphere n_f is fixed while an upwards displacement $\delta = 0.05m$ is applied to the rigid foundation. The contact interaction is produced at the bottom part of the sphere and is assumed to be frictionless.

32198 elements while the finer mesh used here has approximately 14.5 times more elements (465603).

By using the converged value of $F = 0.913N$, analytical solutions of the contact radius and the maximum normal contact traction can be calculated. Thus, $a = 0.292m$ and $P_{max} = 5.11N/m^2$, respectively. Figure 4.10 shows the contact zone achieved from the numerical solution, and the analytical value of a. A good agreement between numerical and analytical solutions is found.

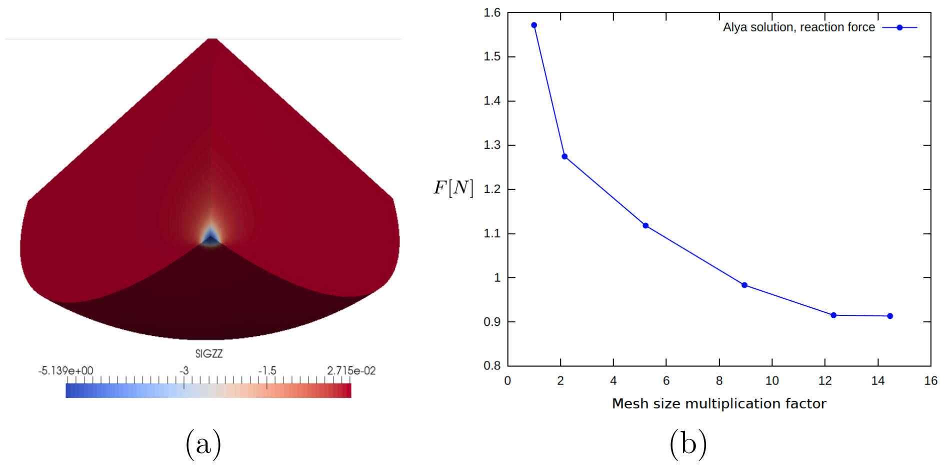


Figure 4.9: (a) (b) reacting force F an node n_f as function of the mesh size.

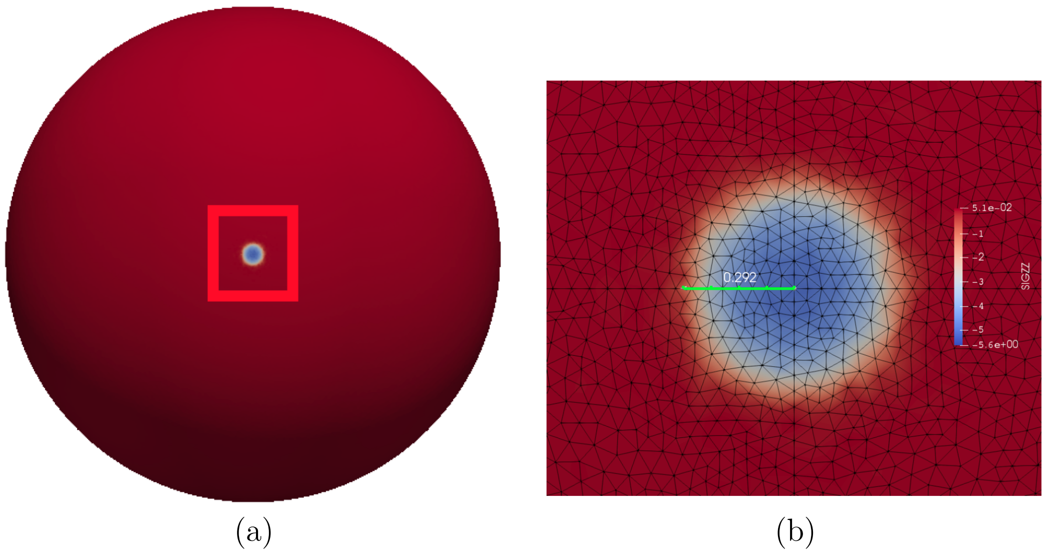


Figure 4.10: Contact zone for converged mesh. (a) View of the contact zone in the sphere. (b) Zoom in tha contact zone. Comparison of analytical (green line) and numerical solutions for the contact radius a.

Figure 4.11 shows the simulated contact pressure distribution at the contact zone for the converged mesh. As expected, the point of maximum contact pressure is located at the bottom part of the sphere, on the axis of rotation. The contact pressure at this node gives $P_{max} = 5.10 N/m^2$ which differes from the analytical value in 0.2%.

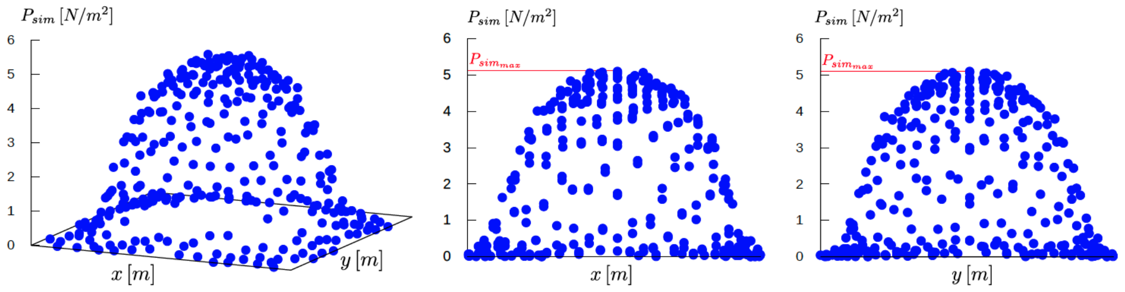


Figure 4.11: Contact pressure distribution at the contact zone.

4.4.2 Impact problem

In this section, the impact of a rigid hemisphere against a plate is studied. One of the main objectives of impact analysis is to obtain the forces as function of time and velocities of the bodies after impact.

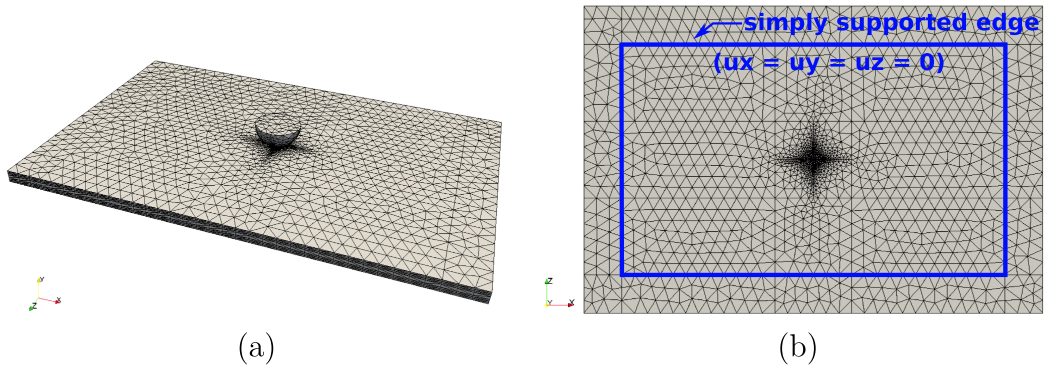


Figure 4.12: Impact setup. (a) 3d view. (b) backview. The diameter of the impactor is $d = 16mm$ and its mass is $m = 2kg$. The dimensions of the plate are $100mm \times 150mm \times 4.16mm$. Plate is simply supported ($u_x = u_y = u_z = 0$) along four internal edges leaving an inner region of $75mm \times 125mm$.

Impact set-up and boundary conditions for this problem are based on ASTM D7136/D7136M-05 standard [70], see Figure 4.13. Similar numerical and experimental analysis have been performed in [71, 72]. The plate is modelled as a transversally isotropic material assuming it as a $T700/M21$ unidirectional carbon/epoxy laminate with stacking sequence of $[0_2/45_2/90_2/-45_2]_s$. Material properties are listed in Table 4.1. The impactor is assumed as an isotropic linear elastic material with an impact energy level of $1.6J$. For the numerical solution of this problem, a mesh composed of approximately 8×10^4 elements for the impactor and 1×10^5 elements for the plate are used. The meshes have been generated in order to enforce node-matching situation at the contact interface. Finally, a total of 48 processors have been used to solve the problem.

Property		Value
E_{11}	Longitudinal Young's modulus	130 GPa
$E_{22} = E_{33}$	Transversal Young's modulus	7.7 GPa
$\nu_{12} = \nu_{13}$	Poisson's ratio	0.3
ν_{23}	Poisson's ratio	0.45 (assumed)
$G_{12} = G_{13}$	Shear modulus	4.8 GPa
G_{23}	Shear modulus	2.655 GPa

Table 4.1: Material properties

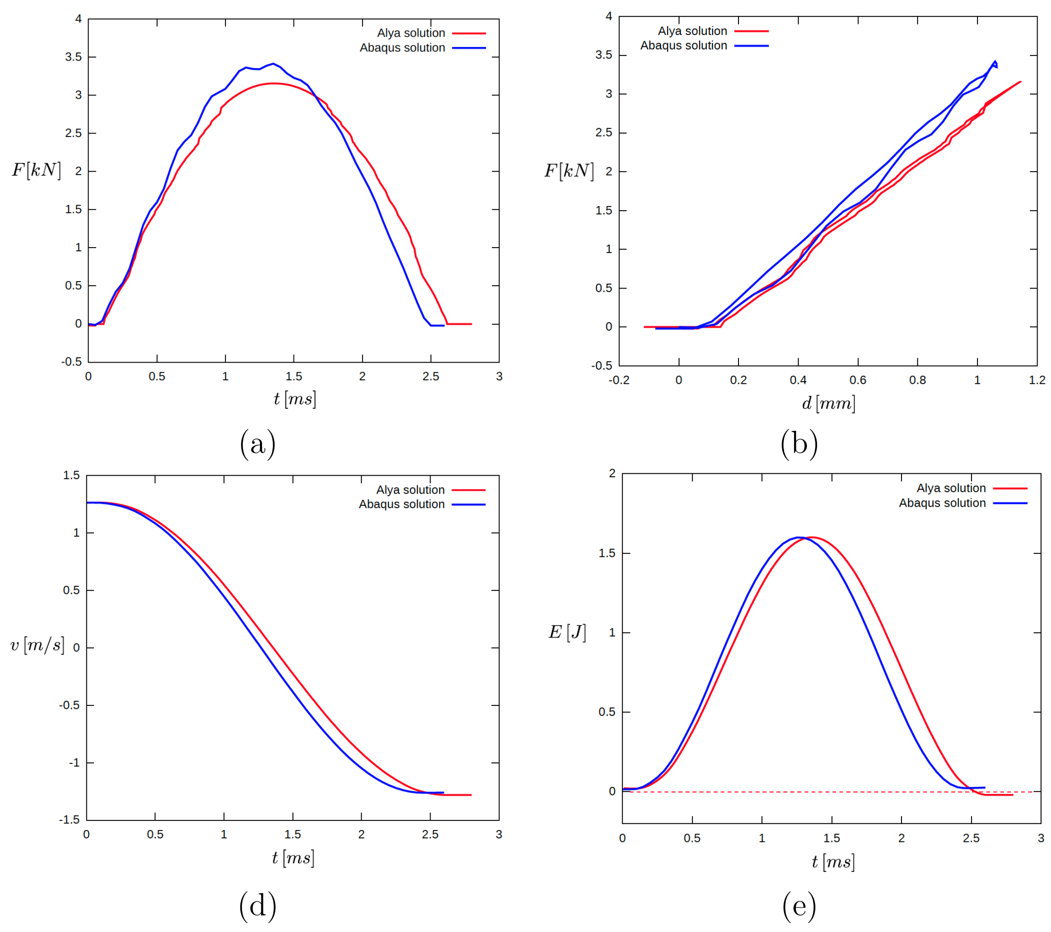


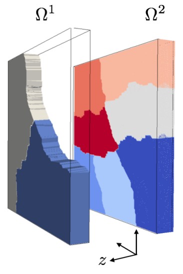
Figure 4.13: (a) contact force-time, (b) contact force-displacement, (c) velocity-time, and (d) energy-time

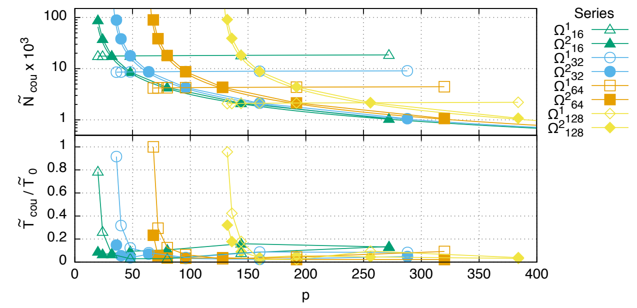
For low velocity impact, the inertia effects are relatively small [73], and hence, an implicit quasi-static solver can be used to solve such problems. The results obtained here, are compared against the ones obtained by the commercial code Abaqus [74]. This code solves the contact problem through a general implicit dynamic contact algorithm based on the node-to-segment discretization of the contact interface. For the enforcement of contact constraints, the introduction of a penalty method is used. It is worth noting that Abaqus uses a completely different approach that the one proposed in this work.

Figure 4.13 shows curves for low velocity impact tests. The Results obtained here agree well with those achieved by Abaqus. From the beginning of the impact ($t = 0$) until about $0.75ms$, a good agreement between contact force-time, contact force-displacement, velocity-time, and energy-time curves are found. It is worth mentioning that these are preliminary results, intended to evaluate the impact response of the algorithm for non-conforming meshes and further development

	Partitions	
	Ω^1	Ω^2
m_i	9	9
$N^i_{ver} \times 10^6$	1.31	1.31
$N^i_{cells} \times 10^6$	6.29	6.29
p^i	$[2^4 : 2^7]$	$[2^2 : 2^8]$

TABLE I





(a) (b)

Figure 4.14: (a) The artificial case studied. It consists on two identical rectangular cuboids with with $n^i = (2^{m_i} + 1)^2$ vertices distributed at the coupling interface (coupled points) perpendicular to the z-axis, $N^i_{ver} = (2^{m_i} + 1)(2^{m_i - 7} + 1)$ vertices, and $N^i_{cells} = 3 \cdot 2^{3(m-2)}$ cells. The colors correspond to the subdomains assigned to the processors, three (one subdomains is hidden) and seven. (b) Number of coupling points $\tilde{N}^i_{cou}$ by processor, and the time $\tilde{T}^i_{cou}$ requested for the location as function of the total number of processors p. Four test series are showed (marks) each has associated two curves, one for each partition Ω^1 and Ω^2. Each series corresponds to maintain constant the number of processors p^1 (16, 32, 64 and 128), while the processors p^2 ranges from 16 to 256. Similar parallel analysis can be found in [36,75] [76,77].

is required. This problem evidences the importance of a conservative transference of loads in Dirichlet-Neumann type contact algorithms for impact problems. For fixed interfaces, as in the case of fluid-structure interaction problems, several methods based on conservative load interpolation schemes that can deal with the information transfer between non-matching meshes have been proposed However, to the best of our knowledge, the extension of such methods to moving interfaces have not yet been reported, and their implementation in a parallel computational code is not straightforward and requires additional development. This key issue is left for future work.

4.4.3 Parallel performance analysis

In order to apply the contact approach described in Section 4.3 to large-scale problems, two factors must be considered: (1) the performance of the computations for each individual subdomain, and (2) the performance associated to the coupling scheme. This section addresses the parallel implementation of the coupling scheme for the contact approach. In particular, it is limited to show the availability of the localization algorithm employed to achieve a good performance, as well as the possible limitations of the coupling performance. When the coupled systems remain unchanged during the simulation process, the localization can be performed as a preprocessing task at the beginning of the simulation. In this case, the required time for the localization task is not important when it is compared with the rest of the simulation [78]. However, it can represent a serious drawback in the performance of a coupled system where at least one of the systems dynamically changes its position in time. This is the case of the contact of deformable bodies.

Figure 4.14(a) shows an artificial case where two geometrically identical meshes are used to exemplify how the localization algorithm operates on a parallel system. The analysis addresses the behaviour of the algorithm when the number of processors p^i and p^j for each partition changes, while the number of vertices N^i_{ver} and cells N^i_{cells} used in the discretization of each partition is maintained constant. See Figure 4.14(a) Table I for details.

Figure 4.14(b) shows the number of coupling points $\tilde{N}^i_{cou}$ by processor, and the time $\tilde{T}^i_{cou}$ requested for their the location as function of the total number of processors $p = p^1 + p^2$. The tilde is used to indicate the median of the values. Four test series are showed. Each of them correspond to maintaining the number of processors p^1 (2^4, 2^5, 2^6 and 2^7) constant, while the processors p^2 range from 2^2 to 2^8. Furthermore, each serie has two curves associated, which corresponds to each partition.

The results show that the coupling points $\tilde{N}^1_{cou}$ of the partition Ω^1 remain constant for each test serie. For the partition Ω^2, the coupling points $\tilde{N}^2_{cou}$ decrease as the total number of cores increases. As consequence, the location times $\tilde{T}^2_{cou}$ corresponding to Ω^2 remain relatively constant, while $\tilde{T}^1_{cou}$ decreases as the total number of cores increases. It is worth noting that, the difference between the lowest ($\sim 1 \times 10^5$) and the highest ($\sim 1 \times 10^7$) values achieved by $\tilde{N}^2_{cou}$ (about two orders of magnitude) produces a decrease of $\sim 100\%$ in $\tilde{T}^2_{cou}$. The above suggests that a suitable selection of the processors distribution is necessary to achieve optimal location times.

4.5 Summary and remarks

In this chapter, a method to address contact between deformable bodies has been described. This method considers the use of a partitioned coupling approach to simulate two deformable bodies interacting through a common interface in which momentum is exchanged. In order to show the validity of the proposed approach, two relevant cases are presented. The first case involves a contact problem, while the second case corresponds to a problem of impact. Finally, an artificial case is used to exemplify how the localization algorithm operates on a parallel system.

In order to validate the coupling approach and tool, an elastic ball contacting with a rigid plane is firstly considered. The contact pressure obtained by the proposed coupling approach is compared with the analytical solution calculated in [69]. The results show an excellent agreement with the theory. Regarding the impact problem, a rigid hemisphere is impacted against a carbon/epoxy laminate. Results are compared with these obtained from the comercial code Abaqus [74]. It is worth noting that Abaqus makes use of a penalty parameter for the enforcement of contact constraints, resulting in a different modelling strategy. These preliminary results show some differences achieved by using Abaqus. Finally, two identical rectangular cuboids are used to perform a parallel analysis of the localization algorithm used to search the contact interface. Results show that an appropriate distribution of processors used in the simulation encourages the time employed to identify the contact interface. Chapter 6 gives a detailed study related to times employed to exchange data and execute the solvers.

To conclude, the iterative coupling approach described here has been successfully validated and applied to impact problems. Additionally, a performance analysis shows that the iterative Dirichlet-Neumann approach used here to model contact problems properly executed in parallel architectures. The future work, considers the use of this methodology in the solution of more general applications, its validation, as well as the investigation of its limitations.

5

Conjugate heat transfer

This chapter is focused on describing a methodolgy to address *conjugate heat transfer* problems using a partitioned approach. Here, an iterative process is used to enforce the effects related to the exchange of thermal energy through the fluid-solid interface. Additionally to this iterative process, a methodology to deal with the time-disparity arising form the difference in the temporal scales between fluids and solids is also presented.

The rest of the chapter describes the details related to the iterative coupling approach and the time-disparity methodology as follows: In Section 5.1, a convective heat transfer problem is used to introduce basic concepts related to heat transfer in solids and fluids. Section 5.2 describes the mathematical details related to solve the coupled partitions. Section 5.3 deals with the numerical approach used to solve the coupling system. Section 5.4 describes briefly the results achieved using the proposed coupling approach, as well as the methodology used to overcome the time-disparity. Remarks and conclusions are given in last section.

5.1 Physical background

This section introduces basic concepts widely used in heat transfer, particularly *convective heat transfer*. The study of convective heat transfer problems is based on *heat transfer* and *fluid mechanics* principles. These principles will be introduced using two examples. The first example is related to *forced convection*, while the second is related to *transient heat conduction*. In convective heat transfer problems the Fourier's law of conduction along with the Newton's law of cooling are used to approximate the effect that a convective fluid has on a solid. In a transient heat transfer problem, Biot and Fourier numbers can be used to estimate the behaviour of a solid subjected to different conditions. The relations between these four ingredients allows understanding the solid-fluid thermal coupling discussed later.

5.1.1 Forced convection

This section describes the calculation of the convective heat transfer coefficient using the Newton's law of cooling, and the Fourier' law of conduction, along with an appropriate calculation of the fluid velocity and temperature distributions.

The *hydrodynamic boundary layer* is the result of the friction due to relative movement of a fluid with respect to a non-slipping surface. Figure 5.1 depicts the case when a laminar viscous flow with a free stream velocity $\mathbf{v}_\infty = (v_\infty, 0)$ comes into contact with a isothermal flat plate. From the contact point O, and through the fluid-solid interface, the fluid velocity $\mathbf{v}_i = (v_x, v_y)$ is zero. As the normal distance y from the interface increases, the x-velocity component v_x of the fluid also increases until it reaches the free stream velocity $\mathbf{v}_\infty$ again. The normal distance δ in which this takes place defines the *thickness* of the hydrodynamic boundary layer. For the flat plate, it can be shown *numerically* that this thickness is related to the Reynolds number $Re_x = x\rho V_f/\mu$, and the distance x to the contact point, as $\delta \simeq 5x\ Re_x{}^{-1/2}$ [79].

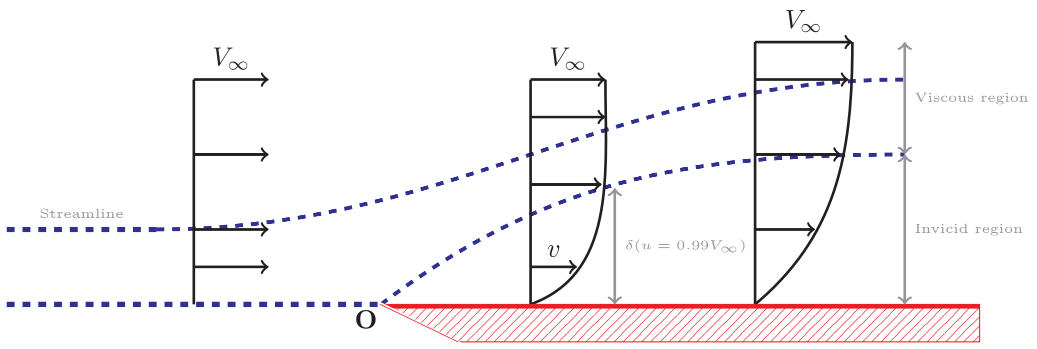


Figure 5.1: Hydrodynamic boundary layer

As the hydrodynamic boundary layer, the difference in temperature between the flow and solid also results in the development of a *thermal boundary layer* along the fluid-solid interface. Figure 5.2 depicts the case when the free stream temperature T_∞ is longer than the interface temperature T_i. In general, the *thermal boundary layer thickness* δ_t is achieved when the difference of temperature $\Delta T = T_\infty - T$ is zero. This difference increases as the thickness δ_t decreases, until it achieves a maximum $\Delta T = T_\infty - T_i$ on the fluid-solid interface. From the integral boundary layer energy equation for the laminar flow over a flat plate [79], it can be demonstrated that hydrodynamic and thermal boundary layers are related by $\delta_t/\delta = \sqrt[3]{13/(14Pr)}$, where $Pr = \nu/\alpha = (\mu/\rho)/(k/(\rho c_p))$ stands for the *Prandtl number*, and x_0 marks the point where the heating of the plate starts.

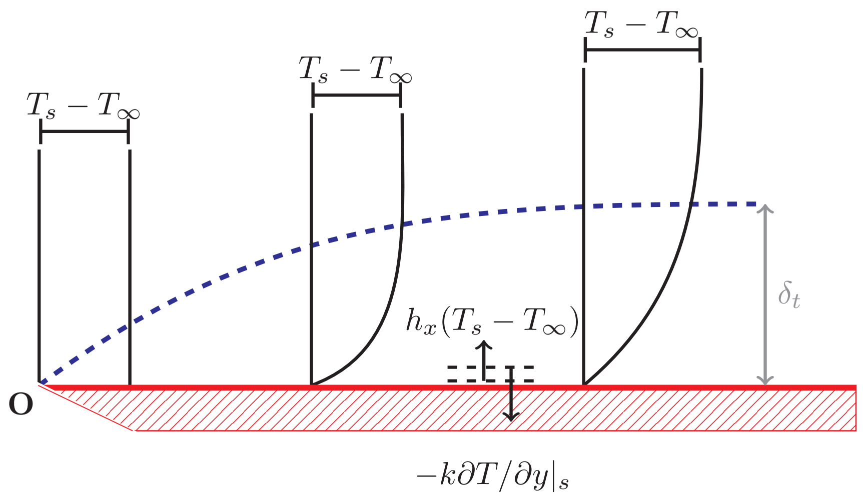


Figure 5.2: Thermal boundary layer.

The *Newton's law of cooling* establishes that the difference of temperatures between fluid and solid at the interface leads to a heat flowing from the higher to the lower temperature region. Thus, from Newton's law, the *heat flux density* due to convection (the rate of heat transfer per unit of area)

$$\dot{q} = \frac{\dot{Q}}{S} = h(T_i - T_\infty) \quad [\mathrm{J/s\text{-}m^2}] \tag{5.1.1}$$

relates the *rate of heat transfer* $\dot{Q}$ through the surface S, due to the difference of temperature $T_\infty - T_i$. Under certain conditions this relation can be used to determine the wall temperature variation T_i, the heat flux $\dot{q}$ from the wall to the fluid, or an appropriate value for the *coefficient of convective heat transfer* h.

Because the fluid velocity is zero at the fluid-solid interface, the rate of heat transfer at the wall can be attributed to conduction in the solid in absence of sources. The *Fourier's law of heat conduction* can be used to determine the corresponding rate of heat transfer per unit of area $\dot{q}$ transferred through the wall interface. With the wall along x-coordinate, Fourier's law states that heat flux density due to conduction can be expressed as

$$\dot{q}_y = -k\frac{\partial T}{\partial y} \tag{5.1.2}$$

where k is the thermal conductivity in J/s-m-K.

In the case of a laminar flow over a flat plate, an energy balance can be used to obtain the relation between the coefficient of heat transfer and the Fourier's law [79].

Thus, at the fluid-solid interface

$$\dot{q} = h(T_i - T_\infty) = -k(\partial T/\partial y)_i. \tag{5.1.3}$$

In particular, it can be demonstrated that for the flat plate under consideration, the heat flux density can be calculated from its temperature profile $(T-T_s)/(T_f-T_s) = 3y/(2\delta_t) - 1/2(y/\delta_t)^3$, which in turn, is calculated from the equation of the thermal boundary layer. Thus, the coefficient of heat transfer in the case of a laminar flow over a flat plate is given by $h = 3k/(2\delta_t)$.

Summarizing, in order to quantify the convective heat transfer coefficient h, firstly, it is necessary to find the temperature profile $T = T(x, y)$ through the fluid-solid interface, along with the hydrodynamic δ and thermal δ_t boundary layers. This can be performed using the Prandtl's boundary layer equations under certain assumptions (the flow is steady, incompressible, and with viscosity constant; the shear in the tangent direction, and the vertical pressure gradient are negligible). Once this is done, the Fourier's law of conduction $-k\partial T/\partial y$ can be used to calculate the heat flux $\dot{q}$ from the wall to the fluid. Finally, the convective heat transfer coefficient is given by the Newton's law of cooling as $h = \dot{q}/(T_i - T_\infty)$. It is worth noting that, identifying accurate values of the coefficient of heat transfer h is one of the principal challenges in convective heat transfer.

5.1.2 Transient heat transfer

While the transfer of heat in a fluid is usually dominated by convection, the heat transfer in a solid is a diffusive process. This can lead to differences of orders of magnitude between the temporal scales of the fluid and solid. For transient heat transfer problems, the *Fourier number* can be used to provide an estimation of the order of magnitude of the diffusion time process. Additionally, if the problem involves the interaction with a fluid, the *Biot number* can be used to estimate the effect that *convection* can perform on the solid.

The Fourier number $Fo = \alpha t/L^2$ arises from the non-dimensional form of the *Fourier equation*

$$\frac{\partial T'}{\partial Fo} + \nabla \cdot (-\nabla T') = 0 \tag{5.1.4}$$

Here, the apostrophe [$'$] indicates the dimensionless form of the temperature T, and $\alpha = k/(\rho c_p)$ represents the *thermal diffusivity.* ρ, c_p, and k stand for density, calorific capacity, and conductivity, respectively. L represents a characteristic length scale used to obtain the dimensionless position gradients.

The Biot number $Bi = hL/k$ appears when Newton's law of cooling is considered as boundary condition in the non-dimensional form of the *Fourier equation*

$$\int_\Omega \frac{\partial T'}{\partial Fo} d\Omega - \int_{\Gamma_k} \nabla T' \cdot d\mathbf{\Gamma} + \int_{\Gamma_h} Bi\ (T' - T'\infty) d\Gamma = 0 \tag{5.1.5}$$

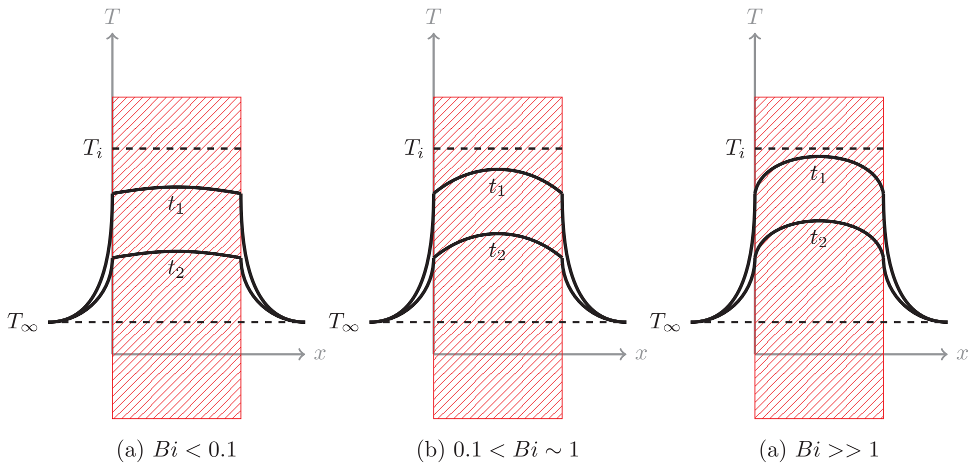


Figure 5.3: Relation between the Biot number and the temperature profile. Temperature gradients for slabs are presented as function of both space and time. In the first slab, thermal conductivity is so high that temperature anywhere within the slab is assumed to be the same. In the second slab, thermal conductivity is smaller by a factor of 10. We see that temperature varies noticeably within the slab. In the third slab, thermal conductivity is smaller by a factor of 100. In this case, temperature varies markedly within the slab [80].

Here, the integral is performed through a solid body Ω, with Dirichlet or Neumann boundary conditions on Γ_k, and convective boundary conditions applied particularly on Γ_h.

In order to study the effect that convective heat transfer has into a solid, some *simplifications* can be considered [80]. A widely used simplification assumes steady-state conditions, i.e., $Fo \to \infty$. Another simplification assumes that the temperature is exclusively a function of time. An example of that is the *lumped analysis*. In this case, the temperature changes only with the time, then the dimensionless form of Fourier equation can be spatially integrated. By considering an *arbitrary* solid of surface S, characteristic length L, volume $V \approx S\ L$, and initial conditions $T = T_0$, then the temperature T at any time t can be obtained as

$$\frac{T - T_f}{T_0 - T_\infty} = \exp\Big(-t/\tau\Big) = \exp\Big(-Bi\ \alpha\ t/L^2\Big) = \exp\Big(-BiFo\Big) \quad (5.1.6)$$

where $\tau = (\rho c_p V)/(h\ S)$ is the *thermal time constant*, hS the *convective resistance*, and $\rho c_p V$ the *lumped thermal capacitance*. Thus, the Biot number can be used to quantify the effect of convection on the solid. Temperature profiles within the solid respect to the surface temperature are shown in Figure 5.3. At a given time t_i, the condition $Bi < 0.1$ states that the temperature T of the body remains spatially uniform, i.e., $(T - T_\infty)/(T_i - T_\infty) \approx 0$. On the other hand, if $Bi > 1$, the change in temperature $T - T_\infty$ within the solid due to convection becomes important.

5.1.3 Coupled problem

An accurate solution of problems where conduction and convection are combined, requires the exact calculation of the convective heat transfer coefficient. Firstly, the solution of a solid exchanging heat with a fluid involves the calculation of the convective heat transfer coefficient h. It is worth noting that, rather than a constant, the coefficient h is distributed on the solid-fluid interface. Thus, accurate solutions of the temporal evolution and the spatial distribution of the temperature in the solid can be achieved if these spatial and temporal dependences of h are considered. On the other hand, if, for instance, either the solid or the fluid evolves with time, the temperature T_i and the conduction heat flux density $\dot{q}$ on the interface, as well as the fluid temperature T_∞, will also evolve, so that a new distribution of h must be considered. Additionally, the heat transfer coefficient is influenced by the physical properties of the fluid and solid when they are function of the temperature. From the above, it is possible to conclude that due to these issues, an exact calculation of the distribution of the convective heat transfer coefficient represents a difficult challenge.

The coupling approach proposed in this thesis makes use of boundary conditions of fourth kind to perform an *exact* solution of convective heat transfer problems [81]. Instead of simply using the convective heat transfer coefficient, a boundary condition of fourth kind uses the temperature and heat flux distributions through the common interface defined by fluid and solid partitions. The use of these boundary conditions in the solution of fluid-solid thermal coupling problems is detailed in the following sections.

5.2 Formulation for CHT problems

The previous section introduces basic physical principles behind convective heat transfer problems: the convective heat transfer coefficient, the Biot number and the Fourier number. This section will briefly introduce mathematical details related to the fluid and solid solvers, see [30] for details. The modelling presented here involves the use of a low-Mach approximation of the Navier-Stokes equations for the fluid, while the solid is solved using a general conduction equation. These equations are solved using the finite element method as implemented in the Alya code [24]. The partitioned multi-physical approach whereby the coupling boundary conditions are enforced will be described in Section 5.3.1.

5.2.1 Mathematical modelling

General conduction equation

The governing equation describing the temperature distribution of a rigid solid is given by the general conduction equation

$$\mathcal{L}_s T \equiv \rho c_p \partial T/\partial t + \nabla \cdot (-\kappa \nabla T) \quad = \quad q_v \quad \text{in } \Omega^s \subset \mathbb{R}^3 \tag{5.2.1}$$

where ρ, c_p, κ, and q_v are density, specific calorific capacity, conductivity, and volumetric heat source, respectively. Additionally, initial conditions $T(.,0) = T_0$ in the solid partition Ω^s complement to the Dirichlet $T = T_D$ on $\partial\Omega^s_D \subset \partial\Omega^s$, and Neumann $\mathbf{n}\cdot\kappa\nabla T = \psi_N$ on $\partial\Omega^s_N \subset \partial\Omega^s$, boundary conditions, so that $\partial\Omega^s = \partial\Omega^s_D \cup \partial\Omega^s_N$ and $\partial\Omega^s_D \cap \partial\Omega^s_N = \emptyset$.

Low-Mach equations

The governing equations describing the fluid flow correspond to the low Mach number equations given by the zeroth-order Navier-Stokes equations

$$\mathcal{L}_{lm}\boldsymbol{U} \equiv \begin{cases} \partial\rho/\partial t + \nabla\cdot(\rho\mathbf{u}) & = \quad 0 \\ \partial(\rho\mathbf{u})/\partial t + \nabla\cdot(\rho\mathbf{u}\mathbf{u}) + \nabla p & = \quad \nabla\cdot\boldsymbol{\tau} \\ \partial(\rho E)/\partial t + \nabla\cdot(\rho H\mathbf{u}) & = \quad \nabla\cdot(-\boldsymbol{q}) \end{cases} \quad \text{in } \Omega^f \subset \mathbb{R}^3 \tag{5.2.2}$$

where ρ, $\mathbf{u}$, p are density, velocity field, and pressure, respectively. The total energy E and the enthalpy H are given by

$$E \quad = \quad \frac{1}{\gamma - 1} T \tag{5.2.3}$$

$$H \quad = \quad E + \frac{p}{\rho} = c_p T \tag{5.2.4}$$

Moreover, $\boldsymbol{\tau}$ indicates the sum of the molecular and Reynolds stress tensor components. According to the Boussinesq approximation, one has:

$$\boldsymbol{\tau} \quad = \quad \mu(\nabla u + (\nabla u)^T) - \frac{2}{3}\mu\nabla\cdot\mathbf{u}\mathbf{I} \tag{5.2.5}$$

The heat flux vector components, $\boldsymbol{q}$, are given by the Fourier law

$$\boldsymbol{q} = -\kappa\nabla T \tag{5.2.6}$$

The transport properties are expressed in terms of the molecular viscosity μ and the conductivity κ with the Sutherland law

$$\mu = C_1 \frac{T^{3/2}}{T + C_2}$$

$$\kappa = C_3 \frac{T^{3/2}}{T + C_4} \tag{5.2.7}$$

where C_1-C_4 are constants for a given gas. For air at moderate temperatures, $C_1 = 1.458 \times 10^{-6} kg/(msK^{1/2})$, $C_2 = 110.4K$, $C_3 = 2.495 \times 10^{-3}(kgm)/(s^3K^{3/2})$, $C_4 = 194K$. Finally, the system is closed by the equation of state that relates the density ρ and temperature T as

$$p_0 \quad = \quad \rho\, R\, T \tag{5.2.8}$$

where p_0 and R stand for a reference pressure and the specific gas constant, respectively.

Additionally, initial conditions $\boldsymbol{U}(.,0) = \boldsymbol{U}_0$ in the fluid partition Ω^f complement to the Dirichlet $\boldsymbol{U} = \boldsymbol{U}_D$ on $\partial\Omega^f_D \subset \partial\Omega^f$, and Neumann $\boldsymbol{\psi}_N$ on $\partial\Omega^f_N \subset \partial\Omega^f$, boundary conditions, so that $\partial\Omega^f = \partial\Omega^f_D \cup \partial\Omega^f_N$ and $\partial\Omega^f_D \cap \partial\Omega^f_N = \emptyset$.

Energy equation

The energy equation from the Navier-Stokes equations, is the one directly interacting with the conduction equation of the solid. This equation, rewritten in terms of the temperature reads

$$\mathcal{L}_f T \equiv \frac{\partial(\rho c_p T)}{\partial t} + \nabla \cdot (\rho c_p \mathbf{u} T) \quad = \quad \frac{dp}{dt} + \nabla \cdot (\kappa \nabla T) \tag{5.2.9}$$

Note that, this equation, along with the general conduction equation, is a convective-diffusive reaction equation. In this sense, and as shown below, most of the conclusions given in Section 2.1.1 can be applied for the solution of a conjugate heat transfer problem.

Likewise in the case of the Fourier equation, the non-dimensional form of the energy equation highlights important parameters that must be taken into account. The non-dimensional form reads

$$\frac{\partial(\rho' T')}{\partial t} + \nabla \cdot (\rho' \mathbf{u}' T') \quad = \quad \frac{\gamma - 1}{\gamma}\frac{dp'}{dt} + \frac{\kappa}{Re_\infty Pr_\infty}\nabla^2 T' \tag{5.2.10}$$

where the apostrophe $[']$ indicates the dimensionless form of the variables. The Reynolds number $Re_\infty = \rho U L/\mu$ and the Prandtl number $Pr_\infty = c_p\mu/k$ represent the ratio of inertial forces $\rho U^2 L$ to viscous forces $\mu U L$, and the ratio of the momentum diffusivity μ/ρ to heat diffusivity α, respectively. The product $Re_\infty Pr_\infty$ is called Péclet number Pe. It represent the ratio of the rate of advection to the rate of diffusion. A convective time scale t_f can be approximated as the ratio L/U of given characteristics length L and velocity U.

Finite element method

Unless indicated, the governing equations are solved using the Finite Element method with the Variational Multiscale Stabilization (VMS) approach [82] for the spatial discretization and with a second order Crank-Nicholson scheme for the time integration. The discretization of the low-Mach number equations yields a coupled algebraic system of the form

$$\begin{pmatrix} A_{nn} & A_{nt} \\ A_{tn} & A_{tt} \end{pmatrix} \begin{pmatrix} U_n \\ U_t \end{pmatrix} = \begin{pmatrix} b_n \\ b_t \end{pmatrix} \tag{5.2.11}$$

which is solved iteratively through a Gauss-Seidel method at each linearization step within a time loop. The diagonal submatrix A_{nn} is related to the discretization of continuity and momentum (fluid motion), while the submatrix A_{tt} is associated to the energy equation. The off-diagonal submatrices A_{nt} and A_{tn} take into account the coupling between the terms of the fluid motion and the energy equation. The vectors $[U_n\ U_t]^T$ and $[b_n\ b_t]^T$ represent the unknowns and right-hand side terms of the individual subsystems, respectively. The momentum and continuity equations are solved independently, applying the iterative Orthomin solver for the pressure-Schur complement [83].

Weak formulation

The weak form of Equation 5.2.1 reads:

$$\text{find } T \in \mathcal{U}: \quad \mathcal{A}(T,\boldsymbol{\eta}) \quad = \quad F(\boldsymbol{\eta}) \quad \forall \boldsymbol{\eta} \in \mathcal{V}$$

where

$$\left\{ \begin{array}{rcl} \mathcal{A}(T,\boldsymbol{\eta}) & \equiv & \int_\Omega (\boldsymbol{\eta}\, \partial(\rho c_p T)/\partial t \; d\Omega + k\nabla T \cdot \nabla \boldsymbol{\eta})\, d\Omega \\ F(\boldsymbol{\eta}) & \equiv & \int_\Omega \boldsymbol{\eta}\, q_v \; d\Omega + \int_{\partial\Omega} \boldsymbol{\eta}\; k\nabla T \cdot \boldsymbol{n} \; d\Gamma \\ \mathcal{U} & \equiv & \{T \in H^1(\Omega):\; T = T_D \text{ on } \partial\Omega_D \;\} \\ \mathcal{V} & \equiv & \{\boldsymbol{\eta} \in H^1(\Omega):\; \boldsymbol{\eta} = 0 \text{ on } \partial\Omega_D \;\} \end{array} \right. \tag{5.2.12}$$

Similarly, for Equation 5.2.9:

$$\left\{ \begin{array}{rcl} \mathcal{A}(T,\boldsymbol{\eta}) & \equiv & \int_\Omega (\boldsymbol{\eta}\, [\partial(\rho c_p T)/\partial t + \nabla\cdot(\rho c_p \mathbf{u} T)] + \nabla\boldsymbol{\eta} \cdot k\nabla T)\, d\Omega \\ F(\boldsymbol{\eta}) & \equiv & \int_\Omega \boldsymbol{\eta}\; dp/dt \; d\Omega + \int_{\partial\Omega} \boldsymbol{\eta}\; k\nabla T \; d\Gamma \\ \mathcal{U} & \equiv & \{T \in H^1(\Omega):\; T = T_D \text{ on } \partial\Omega_D \;\} \\ \mathcal{V} & \equiv & \{\boldsymbol{\eta} \in H^1(\Omega):\; \boldsymbol{\eta} = 0 \text{ on } \partial\Omega_D \;\} \end{array} \right. \tag{5.2.13}$$

By applying appropriate boundary conditions, these equations can be written as

$$\begin{aligned} \int_{\Omega^f} \boldsymbol{\eta} \left[\partial(\rho c_p T)/\partial t\right] d\Omega \;&+\; \int_{\Omega^f} \nabla\boldsymbol{\eta}\cdot k\nabla T \; d\Omega \qquad (5.2.14) \\ &+\; \int_{\partial\Omega^f_N} \boldsymbol{\eta}\; \psi_N \; d\Gamma - \int_{\Omega^f} \boldsymbol{\eta}\; q_v \; d\Omega = 0 \end{aligned}$$

$$\begin{aligned} \int_{\Omega^s} \boldsymbol{\eta} \left[\partial(\rho c_p T)/\partial t + \nabla\cdot(\rho c_p \mathbf{u} T)\right] d\Omega \;&+\; \int_{\Omega^s} \nabla\boldsymbol{\eta}\cdot k\nabla T \; d\Omega \qquad (5.2.15) \\ &+\; \int_{\partial\Omega^s_N} \boldsymbol{\eta}\; \psi_N \; d\Gamma - \int_{\Omega^s} \boldsymbol{\eta} \frac{dp}{dt} \; d\Omega = 0 \end{aligned}$$

The weak form of both, energy and general conduction equations, enable understanding the iterative Dirichlet-Neumann algorithm widely applied to conjugate heat transfer problems. This is completely related to the fact that the nature of these two equations is similar to those found in the heterogeneous domain decomposition example discussed in Section 5.3.2. Details are shown in the next section.

Transmission conditions

Partitioned multi-physics problems make use of boundary conditions that ensure the coupled system is well posed. These boundary conditions can be found, following the procedure described in Sections 2.1.1 and 2.1.2. Once the weak form is established, and given a partition Ω^i, the integral across its boundary $\partial\Omega^i$ is divided into three disjoint parts: $\partial\Omega^i_D$, $\partial\Omega^i_N$ and Γ^i. In the first two parts, *physical* Dirichlet and Neumann ψ^i_N boundary conditions are enforced. In the third part, transmission conditions ψ^i_Γ are imposed on the coupling interface Γ^i. Thus, from the weak form of the fluid partition Ω^f (Equation 1.1.11) the boundary integral reads

$$\int_{\partial\Omega^f} -\boldsymbol{\eta}\, k\nabla T \cdot \boldsymbol{n}\, d\Gamma = \int_{\partial\Omega^f_N} \boldsymbol{\eta}\, \psi^f_N\, d\Gamma + \int_{\Gamma^f} \boldsymbol{\eta}\, \underbrace{-k\nabla T \cdot \boldsymbol{n}}_{\psi_{\Gamma^f}}\, d\Gamma \tag{5.2.16}$$

The same process applied to the solid partition Ω^s (Equation 1.1.12) results in

$$\int_{\partial\Omega^s} -\boldsymbol{\eta}\, k\nabla T \cdot \boldsymbol{n}\, d\Gamma = \int_{\partial\Omega^s_N} \boldsymbol{\eta}\, \psi^s_N\, d\Gamma + \int_{\Gamma^s} \boldsymbol{\eta}\, \underbrace{-k\nabla T \cdot \boldsymbol{n}}_{\psi_{\Gamma^s}}\, d\Gamma \tag{5.2.17}$$

These equations demonstrate that the normal component of the density flux $-k\nabla T$ can be used as transmission condition to couple both partitions Ω^f and Ω^s. Furthermore, these integrals are similar to those found for the diffusion-transport-reaction equation, see Equation 2.1.6. It can be concluded that the iterative Dirichlet-Neumann algorithm shown in Figure 2.3 can be applied to conjugate heat transfer problems. Although this iterative algorithm can fulfil the constrains imposed by the domain decomposition methods other algorithms can be used to improve the convergence and the stability of the algorithm. These alternative algorithms will be briefly discussed in the next section.

5.3 Methodology

The iterative algorithm proposed in this thesis to deal with conjugate heat transfer problems is described here.

5.3.1 Iterative Dirichlet-Neumann algorithm

In the last section, the weak form of the energy and conduction equations lead to the conclusion that conjugate heat transfer problems can be solved as an *heterogeneous* domain decomposition problem. This means solving iteratively two independent problems. An early approach introduced in [84] proposes to solve the fluid partition using Dirichlet conditions, while Neumann conditions are applied to the solid partition, respectively. Most recent developments suggest the use of the Biot number in order to determine efficient transmission conditions (Dirichlet–Robin or Neumann–Robin conditions) at the fluid-solid interface for weakly transient heat transfer problems [85, 86]. Even though, these methods have been successfully applied to the solution of real configurations, e.g. [87], their use involve the assumption that one partition is solved by a *steady state* approach, while the other is modelled as a *transient* simulation. For the cases studied in this work, these assumptions have not been required to obtain accurate and stable solutions using a general approach, so that the original Dirichlet-Neumann algorithm was chosen.

From the above, the iterative Dirichlet-Neumann algorithm is based on the following procedure. Firstly, lets suppose that initial and boundary conditions are applied to Equation 1.1.1, along with an initial guess T_0 as Dirichlet condition on the fluid-solid interface Γ_s of Ω^s, so that

$$\begin{cases} \mathcal{L}_s T &= q_v & \text{in } \Omega^s \subset \mathbb{R}^d \\ -\kappa \nabla T \cdot \mathbf{n} &= \psi_N & \text{on } \partial\Omega^s_N \\ T &= T_D & \text{on } \partial\Omega^s_D \\ T &= T_0 & \text{on } \Gamma^s_D \end{cases} \tag{5.3.1}$$

The updated temperature field T is used to calculate the normal flux $\psi_{\Gamma^s_D} = -\kappa\nabla T \cdot \mathbf{n}$ crossing the interface Γ^s_D. Once this is done, the low-Mach equations $\mathcal{L}_f \mathbf{U}$ are solved enforcing the normal flux $\psi_{\Gamma^s_D}$ as Neumann condition in the energy equation $\mathcal{L}_f T$. Thus, on the interface Γ_f of Ω^f

$$\begin{cases} \mathcal{L}_f T &= q_v & \text{in } \Omega^f \subset \mathbb{R}^d \\ -\kappa \nabla T \cdot \mathbf{n} &= \psi_N & \text{on } \partial\Omega^f_N \\ T &= T_D & \text{on } \partial\Omega^f_D \\ -\kappa \nabla T \cdot \mathbf{n} &= \psi_{\Gamma^s_D} & \text{on } \Gamma^f_N \end{cases} \tag{5.3.2}$$

As described in Section 1.1.1, throughout the fluid-solid interface Γ, the value of the temperature T_f calculated on Γ^f_N can be different from the temperature T_s obtained on Γ^s_D. In this case, the procedure should be repeated as many times as necessary through an iterative process where the value of the guess field T_0 is updated as a combination of T_s and T_f with a relaxation factor $0 < \theta \leq 1$ to accelerate the convergence. The resulting algorithm is given in Figure 5.4.

Lets $T_0^{(0)}$ denote a starting guess

1. For $k = 0, 1, ...,$ until convergence do:
2. Solve for $T_s^{(k)}$ as follows:

$$\begin{cases} \mathcal{L}_s T_s^{(k)} &= q_v & \text{in } \Omega^s \subset \mathbb{R}^d \\ -\kappa \nabla T_s^{(k)} \cdot \mathbf{n} &= \psi_N & \text{on } \partial\Omega_N^s \\ T_s^{(k)} &= T_D & \text{on } \partial\Omega_D^s \\ T_s^{(k)} &= T_0^{(k)} & \text{on } \Gamma_D^s \end{cases} \tag{5.3.3}$$

3. Solve for $T_f^{(k)}$ as follows:

$$\begin{cases} \mathcal{L}_f T_f^{(k)} &= q_v & \text{in } \Omega^f \subset \mathbb{R}^d \\ -\kappa \nabla T_f^{(k)} \cdot \mathbf{n} &= \psi_N & \text{on } \partial\Omega_N^f \\ T_f^{(k)} &= T_D & \text{on } \partial\Omega_D^f \\ -\kappa \nabla T_f^{(k)} \cdot \mathbf{n} &= \psi_{\Gamma_D^s} & \text{on } \Gamma_N^f \end{cases} \tag{5.3.4}$$

4. Update:

$$T_0^{(k+1)} = T_0^{(k)} + \theta(T_s^{(k)} - T_f^{(k)}) \qquad \text{on } \Gamma \tag{5.3.5}$$

5. Endfor

Output: $(T_s^{(k)}, T_f^{(k)})$

Figure 5.4: Iterative Dirichlet-Neumann algorithm for conjugate heat transfer

5.3.2 Time scale disparity

Due to the physical nature of a conjugate heat transfer problem, the temporal scales between fluids and solids are usually very different. While the heat transfer in a solid is a diffusive process, a fluid is usually dominated by convection. The time scale disparity can be measured by the *solid–fluid time scale ratio* as t_s/t_f. As shown above, the convective time scale t_f is given by the ratio of the characteristic length L and velocity U of the flow. For a solid, the factor L^2/α coming from the Fourier number provides an estimation of the order ot magnitude of the diffusion time process. As a result, the time scale ratio can be calculated as

$$\frac{t_s}{t_f} = \frac{L_s^2/\alpha_s}{L_f/U_f} = \frac{\alpha_f}{\alpha_s}\frac{L_s^2}{L_f^2} Re_f Pr_f \tag{5.3.6}$$

which, as shown later, leads to differences of orders of magnitude between their temporal scales, specially as the Re_fincreases. Numerically, it means that the fluid can require several time steps to generate a change in the solid. i.e., temporal perturbations in the fluid are almost negligible in the solid. In order to deal with this drawback, different approaches can be used. In [87], several time steps are used before updating the fluid partition, while a *steady state* approach is used to solve the solid partition. Another approach is to consider the flow field as a sequence of steady states, so that *transient* calculations are performed in the solid, while the fluid is *steady* [85]. The synchronization in physical time between fluid and solid can be taking into account by enhancing the material conductivity of the solid. It supposes that the ratio of thermal conductivities is almost set to unity [28, 88]. This last strategy will be tested in this work, along with a proposed methodology to address the time scale disparity.

5.4 Case studies

This section describes the results of the coupling methodology presented above. Firstly, two validation cases are presented along with two practical applications. The heat transfer between a thin plate and an incompressible flow is studied in both validation cases. In the first case, the solution obtained with the methodology proposed here is compared against an analytical solution. Additionally, a methodology to overcome the time scale disparity is presented. In the second validation case, the cooling of a steel plate with constant air stream is analysed using the time scale disparity methodology introduced. Regarding the practical applications, a premixed impinging jet flame and a confined premixed jet flame are studied.

5.4.1 Flat-plate

The heat transfer between a thin plate and an incompressible flow is studied here. This problem has been widely analysed so that results can be readily validated [86, 89]. The 2D geometry is depicted in the Figure 5.5. The fluid is characterized by the Reynolds number $Re_\infty = \rho_\infty U_\infty L/\mu_\infty$ based on the length L of the plate, and where the index ∞ is used to indicate the properties of the fluid at region 0 (inlet). At the bottom, there are three succeeding regions. In region 1, symmetric and adiabatic boundary conditions are assumed. In the next region, a wall temperature T_w and no-slip conditions are prescribed. In the last region, symmetric and adiabatic boundary conditions are also applied. Finally, the rest of the domain is supposed to be adiabatic with open boundary conditions.
This conjugate heat transfer problem is now used to validate the methodology proposed in this work. Firstly, velocity and temperature profiles for the fluid partition are obtained. Afterwards, the conjugate results are compared against the analytical solution given by [89]. Finally, results for the conjugate heat transfer benchmark

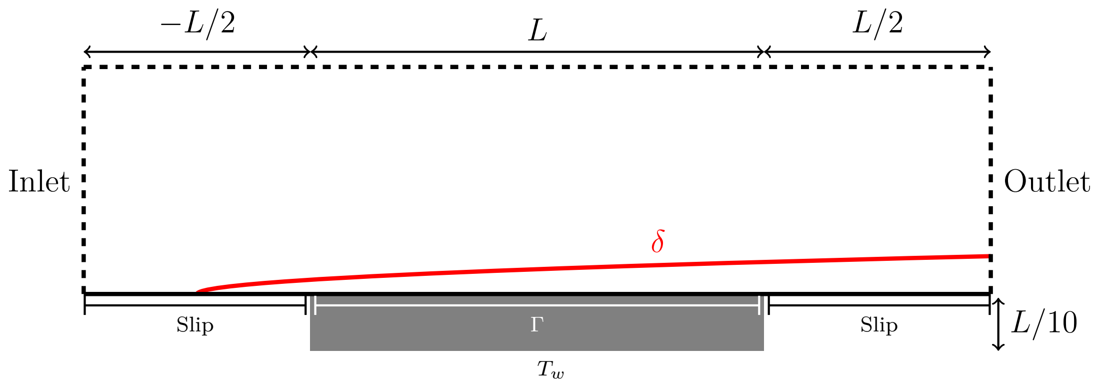


Figure 5.5: Computational domain for flat plate.

proposed in [90] are presented. Note that, this benchmark constitutes a severe test case since it comprises high velocities along with real material properties for the solid.

Fluid partition

A laminar boundary layer flow ($Re_\infty \lesssim 2 \times 10^4$-$10^6$) with *constant* air properties ($Pr = 0.71$) is assumed for validation proposes.

Figure 5.6 shows the normalized temperate distribution $\theta = (T - T_w)/(T_\infty - T_w)$ and the thermal boundary layer thickness δ_T corresponding to $Re_\infty = 100$ when the steady state is achieved. As discussed in Section 5.1.1, the value of this thickness is proportional to the hydrodynamic boundary layer δ, which is defined by the relation $\sqrt{Re_x}\; x/y \approx 5.0$ [79].

In Figure 5.7(a), the velocity U/U_∞ is compared against four solutions given by well-established correlations. An excellent agreement is obtained for results corresponding to $2(y/\delta) - 2(y/\delta)^2$ as well as for the numerical solution of the Blasius differential equation [91–93] Note that the profile $(3/2)(y/\delta) - (1/2)(y/\delta)^2$ is frequently used along with an integral heat transfer equation to obtain a temperature profile approximation in many engineering applications [79, 89]. This temperature profile is compared with the normalized temperature θ in Figure 5.7(b). The numerical results show that the best agreement of θ is achieved with the Blasius profile.

Thermal coupling

The flat-plate considered in the last section is solved here using a four-kind boundary condition approach in region 3. The results are compared with the analytical solution proposed in [90]. The expression for the analytical temperature profile is

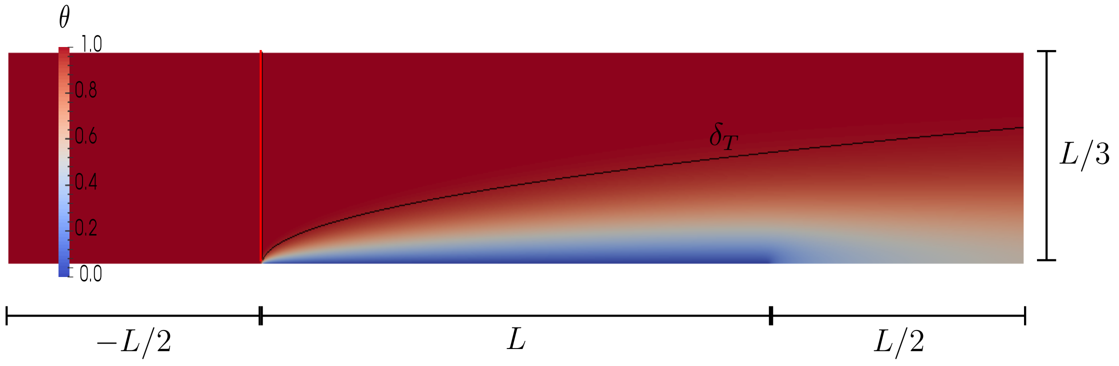


Figure 5.6: Normalized temperature distribution $\theta = (T - T_w)/(T_\infty - T_w)$ on the fluid partition. The curve marks the thickness δ_T of the thermal boundary layer. The wall temperature is $T_w = 0.9T_\infty$ with a Reynolds number fixed to 1×10^3.

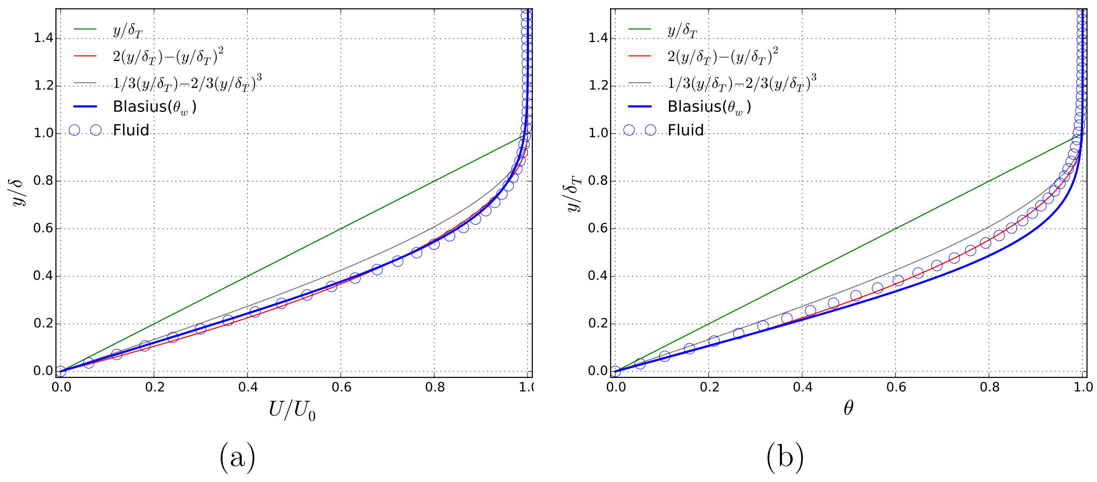


Figure 5.7: Velocity (left) and temperature (right) profile at $x = L/2$. $\delta_T = \delta/\sqrt[3]{Pr}$

given by

$$\vartheta = T - T_\Delta = \begin{cases} \vartheta_\infty & 1 \le y/\delta_t \\ \vartheta_w + 3/2(\vartheta_\infty - \vartheta_w)(y/\delta_t) - 1/2(\vartheta_\infty - \vartheta_w)(y/\delta_t)^3 & 0 \le y/\delta_t < 1 \\ \vartheta_w + \vartheta_w(y/\Delta) & -1 \le y/\Delta < 0 \end{cases} \tag{5.4.1}$$

where $\vartheta_w = \vartheta_0 z/(1+z)$, $z = 3\kappa_\infty\Delta/(2\kappa_s\delta_t)$. It is worth noting that for this case, the wall temperature T_w depends on three properties from the solid: (1) isothermal temperature T_Δ imposed on its bottom wall, (2) conductivity κ_w, and (3) thickness Δ. The ϑ_0 represents the temperature of the plane with zero thermal resistance [81].

The numerical solution of the coupled system has been performed using fluid and solid matching meshes consisting of 119529 and 9728 quadrilateral elements, respectively. For this specific simulation, the properties of solid and fluid are assumed to be constant. Additionally, the conductivity and thermal diffusivity of fluid and solid are supposed to be equal, i.e., $\kappa_s/\kappa_\infty = 1$, and $\alpha_s/\alpha_\infty = 1$. Finally, the temperature T_Δ in the bottom of the solid ($y = L/10$) is fixed to $0.9T_\infty$. Under these assumptions, the normalized temperature distribution $\theta = (T - T_\Delta)/(T_\infty - T_\Delta)$ along with the boundary layer thickness δ_T are shown in Figure 5.8. Figure 5.6 shows a comparison of the temperature distribution against the one obtained for the uncoupled fluid. The boundary layer thickness is decreased due to the heating introduced by the coupling in region 1, see Figure 5.9.

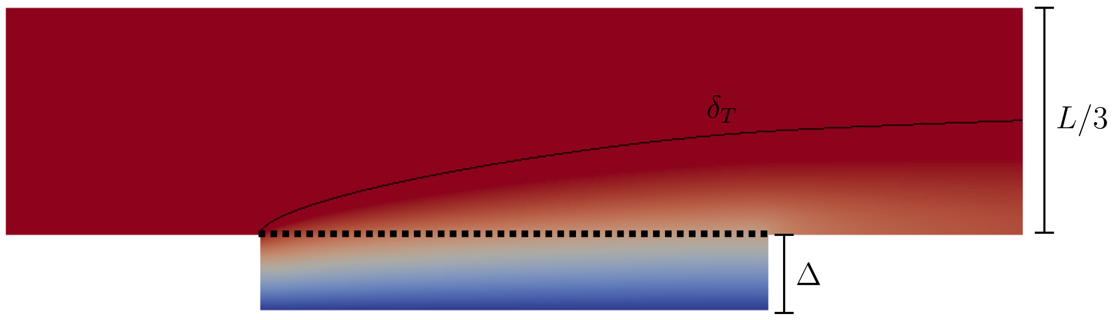


Figure 5.8

Velocity and temperature profiles at $x = L/2$ for this coupled problem are shown in Figure 5.9. Additionally to the fluid partition ($y/\delta_t \geq 0$), the solid has also been included ($-\Delta/\delta_t \leq y/\delta_t < 0$). The results show the velocity profile is practically unaffected, while the temperature profile changes. Of particular interest is that from Equation 5.4.1, the effect of the heating/cooling is to move and scale the original temperature profile $F(y/\delta_t)$, so that $(\vartheta - \vartheta_w)/(\vartheta_\infty - \vartheta_w) = (3/2)(y/\delta_t) - (1/2)(y/\delta_t)^3$. From Figure 5.9, it can be seen that these effects are well captured by the methodology used in this work, even for the Blasius profile. Another important aspect is the temperature profile in the solid. Given as a line of slope $\vartheta_w(\delta_t/\Delta)$, which achieves the wall temperature ϑ_w on the coupling interface ($y/\delta_t = 0$).

So far, the results have shown an excellent agreement with the theory. In what follows, it will be shown that the continuity of the temperature and the heat flux is achieved throughout the coupling interface. After that, the effect of changing the conductivity ratio κ_s/κ_∞, and the thermal diffusivity ratio α_s/α_∞ will be analysed.

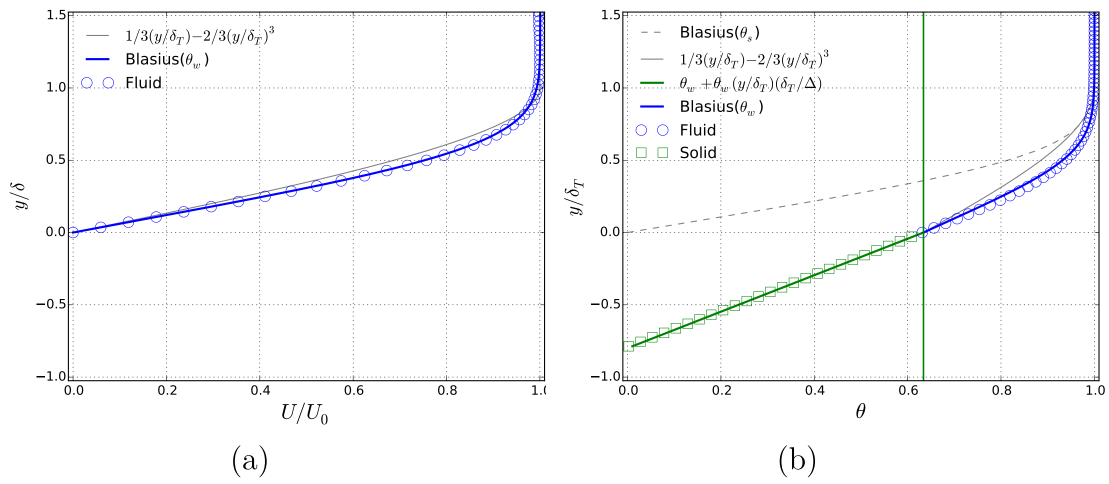


Figure 5.9: Velocity (left) and temperature (right) profile at $x = L/2$.

Continuity of the transmission conditions Figure 5.10 shows the continuity of the transmission conditions achieved when the iterative Dirichlet-Neumann algorithm 5.4 described in Section 5.3 is applied to the solution of the conjugate flate-plate. In this case, the fluid partition is defined as the Dirichlet partition, while the solid partition is considered as Neumann. Thus, the fluid partition is firstly solved in parallel using an initial guess T_0, then the results obtained for the current k-iteration are used to calculate the normal flux $\phi_{\Gamma^s_D}$ to be enforced as Neumann boundary condition on the coupling interface Γ^s_D of the solid. Once the solid solver has finished, the temperature $T_f^{(k)}$ on Γ^s_D is used to calculate the temperature $T_0^{(k+1)}$ to be imposed on the fluid partition. The process is repeated as many times as necessary to achieve a given convergence ϵ so that $|T_s^{(k)} - T_f^{(k)}| < \epsilon$. In this case, the maximum number of sub-iterations is fixed to 50, with a convergence $\epsilon < 1 \times 10^{-5}$, relaxation factor $\theta = 0.8$ and variable time steps. Figure 5.11 shows that, by means of this iterative algorithm, the continuity of the transmission conditions (temperature T_Γ and heat flux density ϕ_Γ) through the interface Γ is ensured at every time step.

Temporal evolution The evolution of the temperature in a point $p = (L_0/2, 0.0)$ until the steady state is reached is shown in Figure 5.11. Before starting the coupling ($t < 0$), the fluid is simulated using $T_\Gamma = 0.9$ as Dirichlet condition. At the beginning of the coupling ($t = 0$), the fluid partition is assumed to be in steady state, with an uniform temperature $T = 0.9$ in the whole solid partition. A total of six curves are shown in the figure. Three for the evolution of the fluid $y/\Delta = \{0.0, 0.05, 0.10\}$, and three for the solid $\{-0.10, -0.05, 0.0\}$. The overlapping beetween two of these six curves provides the continuity of the transmission conditions through the coupling

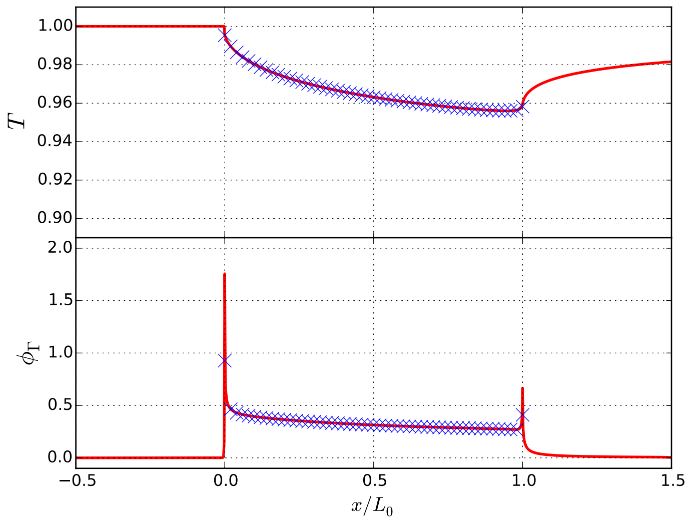


Figure 5.10: Temperature T and heat flux ϕ_Γ for solid (marks) and fluid (line) partitions through the coupling interface Γ when the steady state is achieved.

interface at any time. After the initialization, the curves show an increase of the temperature, until the steady state ($t \sim 10$) is achieved. For the particular case of the curves at the interface ($y/\Delta = 0$), the temperature has changed from 0.9 to $\sim$ 0.963. Note that when Equation 5.1.6 is used to fit the evolution of the temperature in $y/\Delta = 0$, a value for thermal time constant $\tau \sim 4.028$ is obtained. Another observation is that τ is directly related to the Biot number and to the thermal diffusivity, i.e., $\tau = \Delta^2/(Bi\ \alpha)$. The thermal diffusivity of the solid can be easily calculated, however in order to find a Biot number, it is necessary to characterise the heat coefficient using its mean value, for instance. It is due to the fact than h, rather that a constant value, is a number changing along the coupling interface. Thus, when the mean $\bar{h} = 8.203$ is used, the Biot number can be estimated as $Bi \sim 1.651$. Additionally, the thermal time value obtained by $\bar{h}$ ($\tau \sim 4.301$) is close to the one found by the fitting of Equation 5.1.6. As stated in Section 5.1.2, inside a solid whose Biot number exceeds 0.1, the temperature changes substantially in space. This effect is clearly observed in this case, since the temperature for the three curves inside the solid ($y/\Delta = \{-0.10, -0.05, 0.0\}$) have different distributions. The effect of modifying the Biot number and the thermal diffusivity inside the solid is analysed in Appendix A. The main results obtained from this approach are used to study the behaviour of the coupled system in the next section.

Thermal diffusivity effect This section addresses the effect of solid properties variations have on the fluid domain. Appendix A concludes that, given a solid,

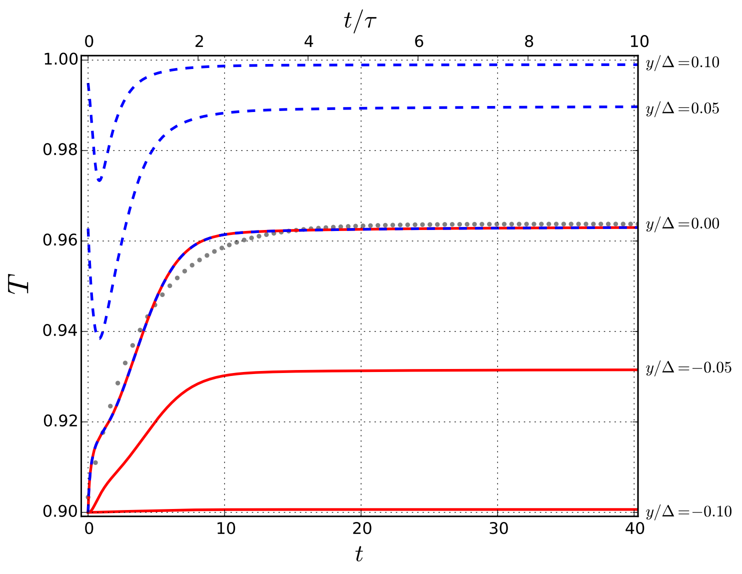


Figure 5.11: Temporal evolution of the temperature T at $x/L_0 = 0.5$ for five vertical points $y/\Delta = \{-0.10, -0.05, 0.0, 0.05, 0.10\}$. Normalised time scale $t/\tau = Bi\ \alpha\ t/L^2$ is shown in the top part. Dashed lines correspond to the evolution of the fluid, while the solid evolution is shown with continuous lines. Dotted line correspond to $T_s(t) = (T_\Gamma - T_\infty)\exp(t/\tau) + T_\infty$.

numerical simulations with the same Biot number can be considered physically and numerically equivalent. These assumptions are examined for this thermal coupling problem.

Five cases are analysed here, which are defined based on the product $\rho_s c_{ps}$, so that the thermal activity ratio K range is $\{10, \sqrt{10}, 1, 1/\sqrt{10}, 1/10\}$. For all cases, the properties of the fluid are kept constant, along with the conductivity of the solid. This is done under the assumption that, in this way, the Biot number is preserved and the simulation time t can be modified without changing the physical characteristics of the solid. The results of this analysis are shown in Figure 5.12 (a1)-(c1) and have good agreement with the ones obtained in Figure A.3. As the thermal activity ratio increases, the time to achieve the steady state decreases. In practice, all temperature curves overlap when they are plotted against the normalised time t/τ (τ is calculated for each case). Finally, the number of steps $n_{\Delta t}$ to achieve the steady state decreases as the thermal activity ratio increases. Note that the behaviour of a fluid partition is similar to the one found in the solid domain.

Figure 5.12 (a2)-(c2) shows the response of the fluid to the changes performed in the solid. When temperature curves are plotted against the normalised time t/τ, all of them tend to overlap. In general, this behaviour is observed for all cases except for $t/\tau \lesssim 1$. This difference is produced by the initialization of the simulations.

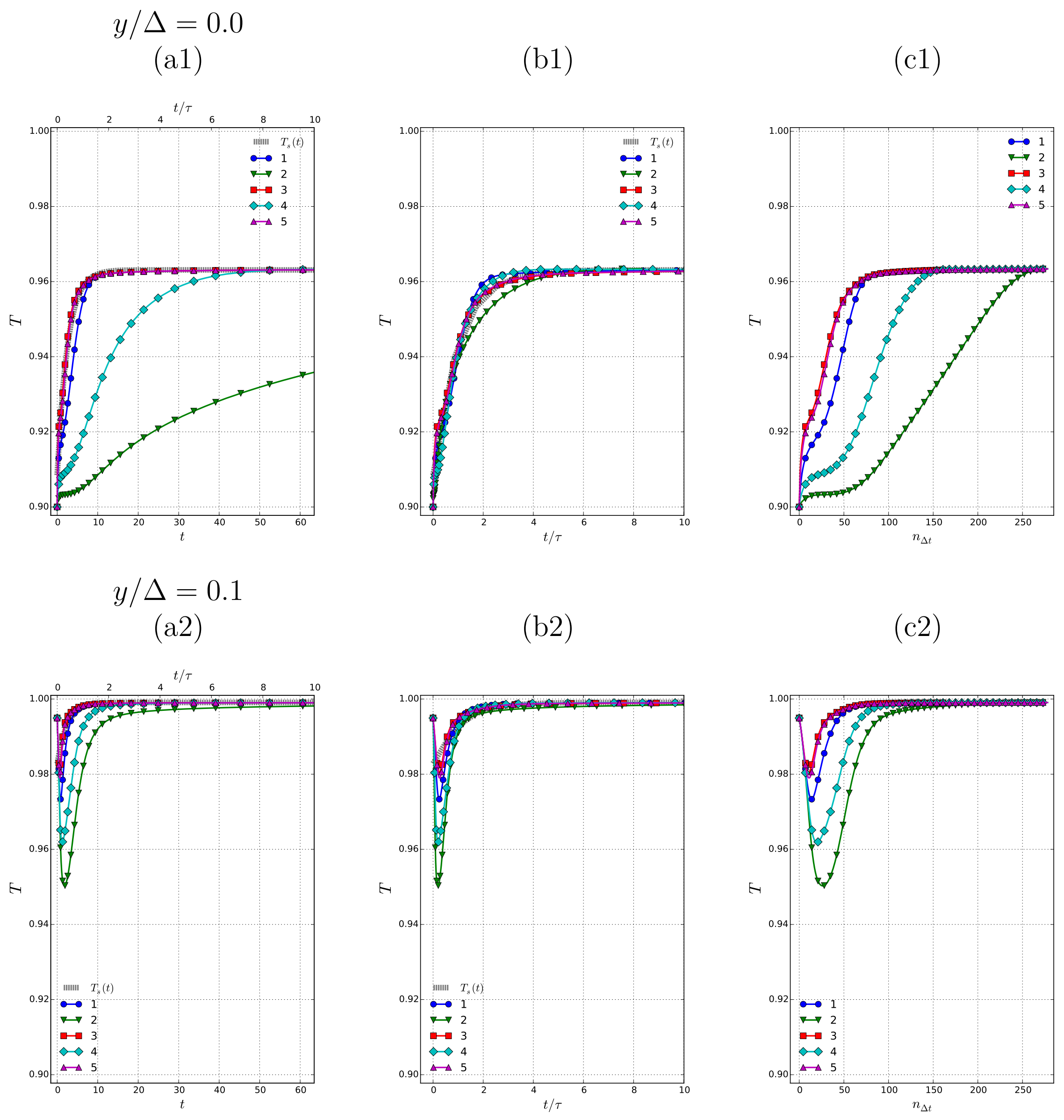


Figure 5.12: Temperature T at $x/L_0 = 0.5$ and $y/\Delta = 0.0$ as function of (a) time t, (b) normalised time t/τ, and (c) number of time steps $n_{\Delta t}$. $K = \{10, \sqrt{10}, 1, 1/\sqrt{10}, 1/10\}$, $\alpha/\alpha_\infty = \{100, 10, 1, 1/10, 1/100\}$, cases 3, 5, 1, 4, 2, respectively.

Regarding the number of steps, as the thermal activity ratio increases, more time steps are required to reach the same temperature.

5.4.2 Convective cooling of a plat plate

The problem analysed here consists of a constant air stream cooling a steel flat plate [94,95]. The air coolant entering is assumed to be a gas with temperature $T_\infty = 273$ K, Prandtl $Pr_\infty = \mu_\infty c_{p\infty}/\kappa_\infty = 0.71$. and Reynolds $Re_\infty = \rho_\infty U_\infty L_0/\mu_\infty = 4.01 \times 10^6$. Before reaching the coupling interface Γ, two succeeding regions are present. After Γ, a third region is defined. In the first and last regions, adiabatic and slip boundary conditions are applied. Isothermal and no-slip boundary conditions are also applied on the second region. Finally, for the rest of the domain, open boundary conditions are used.

Regarding the solid partition, a flat plate of length $L_0 = 0.2\ m$ and thickness $\Delta = 0.015\ m$ made of steel with properties such that $k_s/k_\infty = 1.99 \times 10^3$, and $\alpha_s/\alpha_\infty = 7.58 \times 10^{-1}$ is defined. Under these assumptions, the values corresponding to the time scale ratio and thermal activity ratio ($t_s/t_\infty = 2.1167 \times 10^4$ and $K = 4.353 \times 10^{-4}$, respectively) lead to an important disparity between the time scales. This also means that low level of fluctuations throughout the coupling interface should be expected. Finally, Neumann boundary conditions ($q = 0$) are applied on the solid, which is initially set to 900 K. Note that the coupling simulation is started once the fluid partition has achieved the steady state with constant wall temperature $T_\Gamma = 900$ K at the interface.

The solution of this problem is obtained applying the iterative Dirichlet-Neumann algorithm described in Section 5.3. A convergence study has been conducted to select matching meshes consisting of 119529 (fluid) and 9728 (solid) quadrilateral elements, ensuring sufficient resolution near the wall ($y^+ < 1$). For this specific simulation, Equation 5.2.1 for the solid is solved assuming constant properties. For the fluid, the low-Mach equations described in Section 5.2.1 are used (density, viscosity, and conductivity are function of the temperature). Regarding the coupling treatment, the time step is given by the fluid partition, and is allowed to evolve as

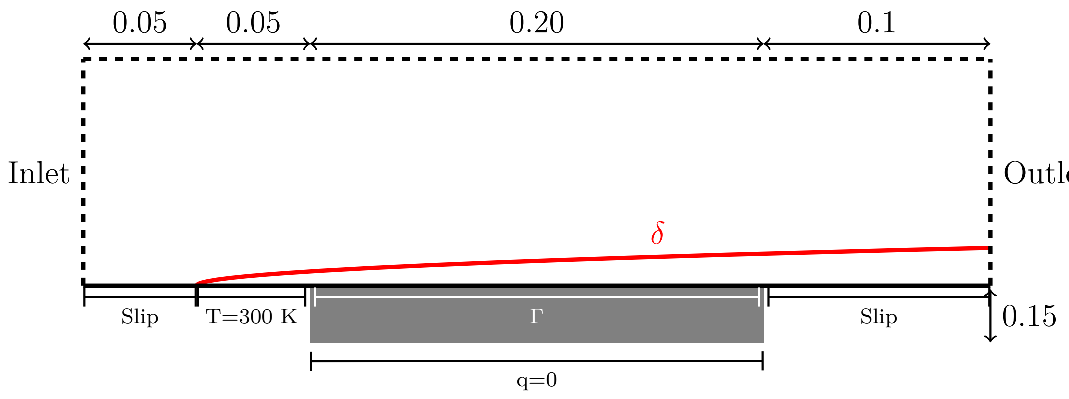


Figure 5.13: Computational domain for convectively cooled plate.

the simulation progresses. The convergence is set to 1×10^{-5}, being required less than 10 iterations for the first three time steps, and only one iteration per time step for the rest of the simulation.

Figure 5.14 shows the temperature evolution at the beginning ($x/L_0 = 0$) and ending ($x/L_0 = 1$) of the coupling interface. Results obtained are compared against the temporal evolution of temperature reported in [94]. A good agreement with the reference data is observed for the entire time history of these two interface points. Due to the fact that the non-coupled walls of the solid are adiabatic, it is expected that, the solid reaches the temperature of the air stream, as the system evolves. The effect of cooling is slightly captured in the range of time simulated. From $t = 0$ to 1, the temperature at the beginning of the interface drops faster than at the ending point. In $x/L_0 = 0$, the temperature changes from 900 to $\sim 874K$, while in $x/L_0 = 1$, the temperature drops from 900 to $\sim 895K$ (only five degrees). Thus, it is possible to conclude that many time steps should be required to simulate the cooling of the solid down to $273K$ (a difference of $\sim 600K$). In this sense, the methodology proposed in Section 5.4.1 to deal with the time scale disparity can be applied to simulate the entire cooling process reducing the number of time steps.

Figure 5.15 shows the entire cooling process at the beginning and ending point of the coupling interface. The time that the solid needs to achieve the air stream temperature seems to be $\sim 3.5 \times 10^3$ s, which agrees well with the fact that the time scale ratio t_s/t_∞ is $\sim 2 \times 10^4$. In order to simulate the entire process, the properties of the solid are modified. For this specific case, the original thermal activity ratio $K_0 = 4.343 \times 10^{-4}$ has been increased up to 4.343×10^{-2} by decreasing ten thousand

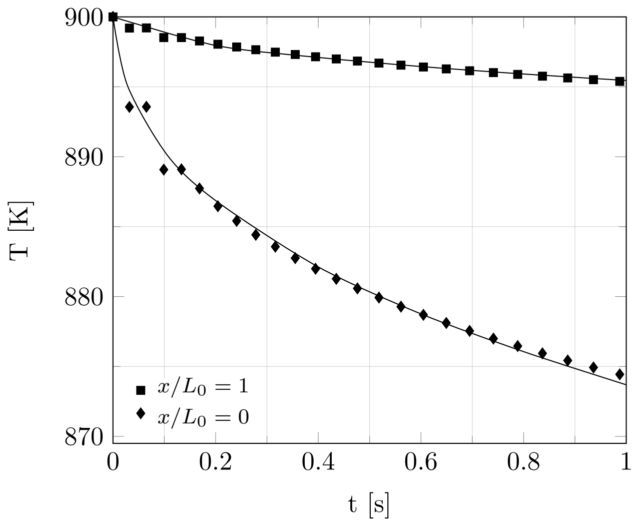


Figure 5.14: Temperature evolution at beginning (diamonds) and end (squares) of the interface Γ. A good agreement with results given by [94] is observed for the entire time history of both interface points.

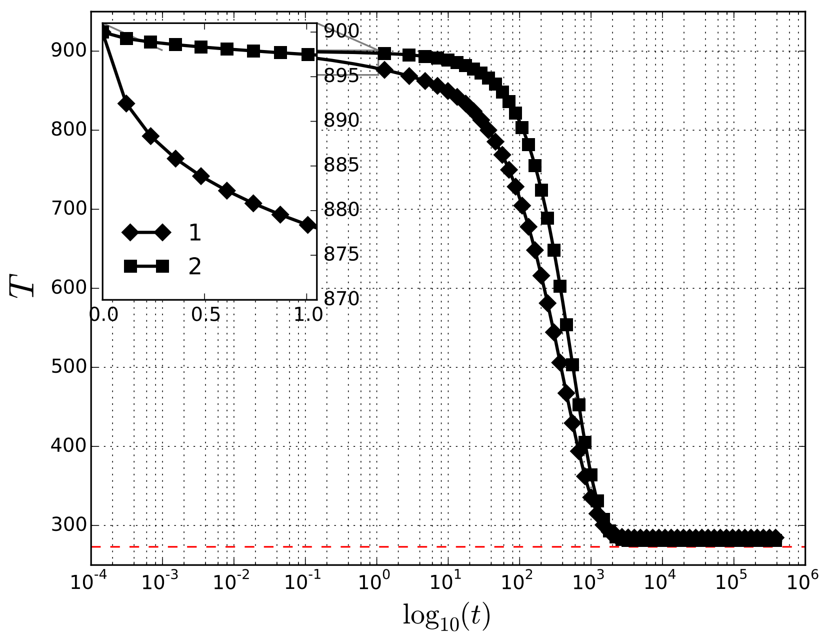


Figure 5.15: Complete temperature evolution at beginning (diamonds), and end (squares) points over interface Γ. The entire cooling process takes $\sim 3.5 \times 10^3$ s. Sub-figure shows the evolution during first second. Dashed line correspond to the temperature air $(273K)$.

times the value of $\rho_s c_{ps}$. This change should maintain the original Biot number, and with different thermal activity radio the physical behaviour of the solid partition.

In order to validate the results two simulations ($K = K_0$ and $10K_0$) have been performed. The temperature evolution obtained at the beginning point of the coupling interface for the three values of thermal activity ratio are compared in Figure 5.16. In (b), the time t corresponding to each curve plotted in (a) has been multiplied by the same factor used to obtain its respective thermal activity ratio. Thus, the curves 1,2, and 3 corresponding to K_0, $10K_0$ and $100K_0$ scaled by 1, 1×10^2, and 1×10^4, respectively.

The resulting curves overlap, so that it is possible to conclude that all of them are *physically* equivalent. Figure 5.15 (b) shows the temperature as function of number of the time steps $n_{\Delta t}$. From this figure, it can be clearly seen that given a number of steps, the change of temperature increases as the thermal activity ratio increases. This means that a larger amount of time steps can be performed as the thermal activity ratio increases. For instance, when $n_{\Delta t} = 400$, the temperature in curve 1 $(K = K_0)$ has hardly changed from 900 to $\sim 870K$, simultaneously the temperature in curve 2 $(K = 10K_0)$ is close to $650K$, while in curve 4 $(K = 100K_0)$ the steady state has been already achieved. Finally, it is worth noting that the total number

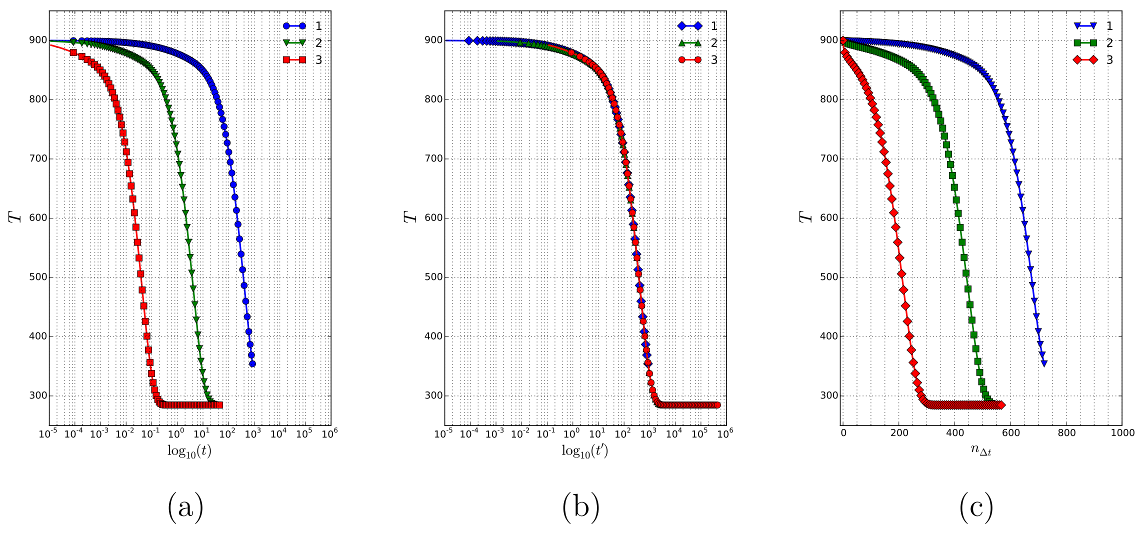


Figure 5.16: Temperature T at $x/L_0 = 0.0$ as function of (a) time t, (b) scaled time t', and (c) number of time steps $n_{\Delta t}$. $K = ((\kappa\rho c_p)_f/(\kappa\rho c_p)_s)^{1/2} = \{K_0, 10K_0, 100K_0\}$, cases 1,2,3, respectively.

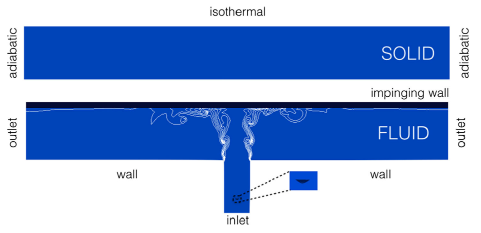
of time steps performed in each curve is different (approximately 725, 550 and 550, respectively). This is due to the fact that these three simulations have been executed during the same computational time (and with the same number of processors). As result, at the end of the simulations, curve 1 is around $350K$, curve 2 seems to have reached the steady state, while the state achieved in curve 3 is completely steady.

5.4.3 Practical applications

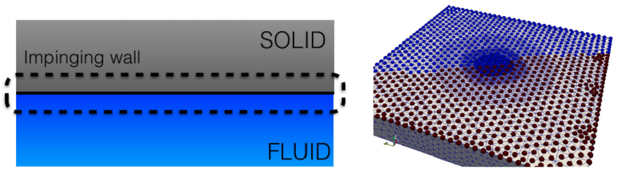
So far, the iterative Dirichlet-Neumann algorithm described in Section 5.3 has been applied in the solution of the thermal coupling between a fluid and a flat-plate. Additionally, the effect of changing the thermal properties of the solid have been studied. The results show that the validity of the iterative algorihm along with the possibility of enhancing the response of a solid when it interacts with a fluid. The rest of this section summarizes the results obtained when the algorihm described here is applied to the solution of two more complex applications: a impinging flame interacting with a wall and for the prediction of heat losses in a confined premixed jet flame.

Premixed impinging jet flame

The present study addresses the effect of the heat transfer condition at the solid wall on a premixed impinging jet flame. Different thermal conditions are imposed at the impinging plate and its effects on the flame dynamics, heat transfer, shear



(a)



(b)

Figure 5.17: (a) Computational domain and boundary conditions. (b) Interface between solid and fluid domain and matching between fluid and solid meshes. The impinging flame configuration considered in this work corresponds to a methane premixed flame with equivalence ratio $\phi = 0.8$ at $T = 298K$ impinging on a flat plate. The nozzle-to-plate distance is $H/D = 2$ and the inlet velocity is set as $30m/s$. The jet inlet diameter is D and the length of the fluid domain parallel to the plate is $20D$. The solid part is represented by a squared domain of length $20D$ and thickness $2D$. The isothermal temperature on the upper side of the solid domain is set as $T_s = 800K$. The nodes at the fluid–solid interface are matching in order to avoid interpolation errors.

stress and wall-jet development are discussed in the context of large-eddy simulation (LES). Aspects regarding heat transfer effects are summarized here, but for details refer to [28].

The computational domain with the corresponding boundary conditions is sketch in Figure 5.17(a). The velocity at the inlet is imposed with a top hat profile and the turbulence is generated in the wake of a bluff body. The inflow condition for the temperature is prescribed with $T = 298K$. All solid walls are assumed adiabatic except the impinging wall, where five different heat transfer conditions are examined. The cases are defined with an isothermal wall, constant heat flux, Robin-type boundary conditions and as a conjugate heat transfer problem. In addition, taking into account the thermal activity ratio ($K = 7 \times 10^{-5}$) and the synchronization in physical time between fluid and solid, the material conductivity of the solid was enhanced by a factor of 100.

The results reveal differences when the boundary conditions are varied at the impinging wall. The differences among the cases not only affect the mean temperature and gradients in the near wall region (Figure 5.18), but also the temperature fluctuations and dynamics of the flow (Figure 5.19). The distribution of gradients over the plate indicates that Dirichlet-type conditions favour localized temperature gradients that can be larger than Neumann-type boundary conditions at particular locations, (Figure 5.20). The analysis of the results indicate that temperature variations, gradients and fluctuations in the near-wall region also require information of the solid domain, and that can only achieved by the use of a conjugate heat transfer approach.

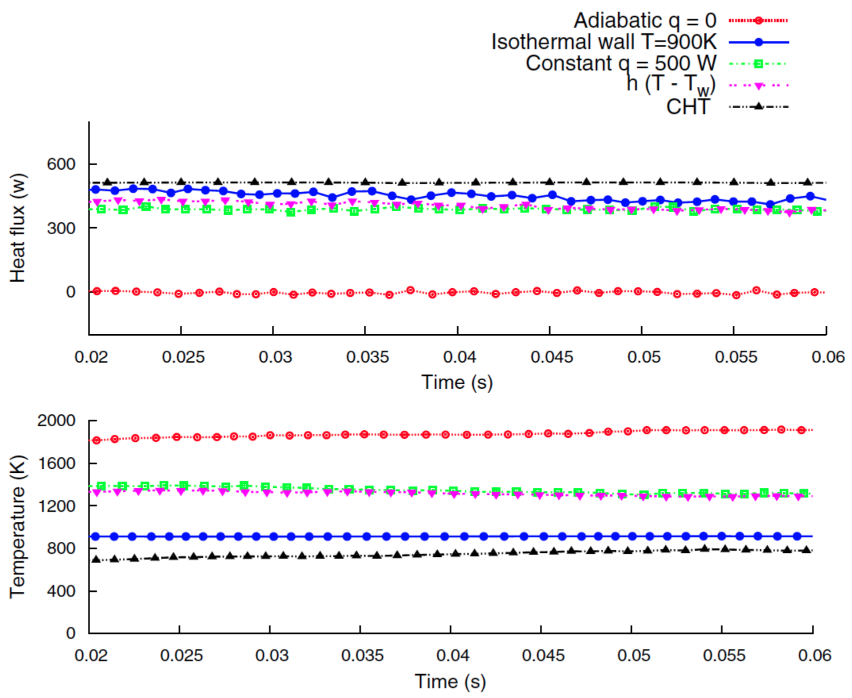


Figure 5.18: Global response of the impinging wall to flame heating. Time evolution of heat flux (top) and temperature (bottom) integrated over the plate. The temporal trends indicate that all the cases (except the adiabatic case) dissipate approximately the same amount of heat after initialization. The wall temperature for the adiabatic case reaches values around the equilibrium temperature ($T \sim 1996K$). The lower bound is given by the CHT case. The other cases are found in between and have a similar response from the perspective of global heat exchange.

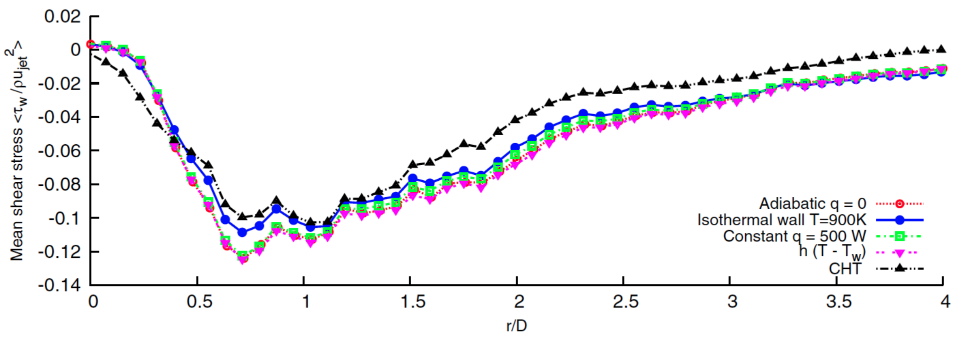
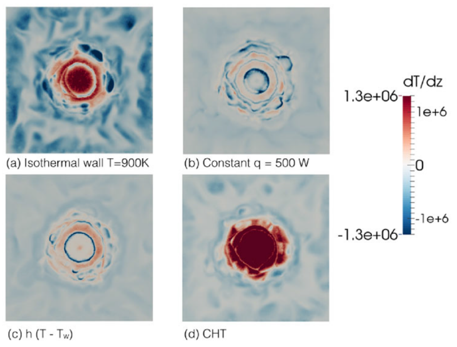


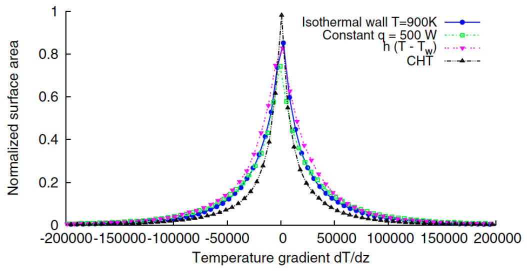
Figure 5.19: Mean wall shear stress along the radial direction. An enlargement of the thickness of the wall jet occurs due to effects of turbulent mixing and convection in the near-wall region. This effect reduces the maximum velocity of the jet and affects the shear stress in the radial direction.

Confined premixed jet flame

The presented study addresses the investigation of the heat loss of a confined turbulent jet flame in a lab-scale combustor. Here, the effect of different heat transfer



(a)



(b)

Figure 5.20: (a) Contour plots of temperature gradients at the wall, and (b) probability distribution of the temperature gradients at the impinging plane at time $t = 0.045$s. A distribution of the temperature gradient over the normalized surface area associates the magnitude of the gradients to the portion of area on which they are distributed relating the sensibility of the boundary respect to changes in flame temperature. In general, it is observed that boundary conditions based on flux prescriptions tend to widen the pdf distribution, lowering the peak of the curve and inducing more uniform temperature gradients over the solid plate. The CHT case undergoes the highest temperature gradient, while the case with constant heat flux has the minimum. The CHT and the isothermal case show narrow distributions indicating that gradients are rather large over particular sections of the plate. Dirichlet-type conditions favour localized temperature gradients that can be larger than Neumann-type boundary conditions at particular locations.

mechanisms and thermal conditions for the chamber walls of a turbulent jet flame configuration is investigated in detail by means of numerical simulations. Aspects regarding heat transfer effects are summarized here, but for details refer to [29].

An sketch of the computational domain for the fluid and solid including main dimensions in mm is shown in Figure 5.21. The test case consists of a premixed turbulent jet flame confined in a rectangular combustion chamber. A lean mixture of methane and air at equivalence ratio 0.71 is injected into the combustor through a circular pipe with diameter d. No-slip boundary conditions are set for the velocity at the walls. In order to obtain a turbulent flow field at the inlet plane of the flow domain, a precursor LES simulation is performed in a pre-processing step and the velocity components are sampled at the inlet plane of the combustor simulation at every time step.

In this work, the *dual heat transfer* (DHT) approach is used to obtain steady temperature distributions in the solid [96]. In essence, this approach differs from the the original iterative Dirichlet-Neumann algorithm described in Section 5.3 only in the fact that steady states of both partitions are used at each coupling step. Thus, fluid and solid are computed at the same time, so that each of them is fully converged. Only the fully converged solution at the interface is used as a boundary condition in the other domain. The fluid domain is computed with an initially

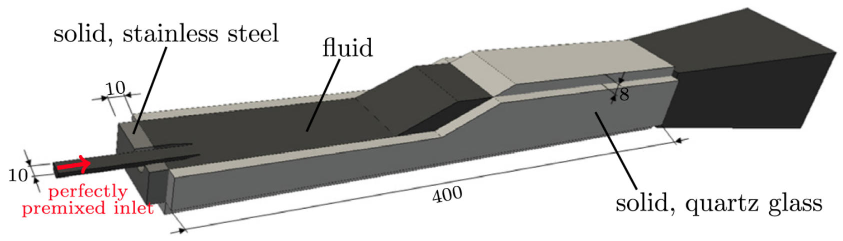


Figure 5.21: Experimental facility at the German Aerospace Center (DLR). The combustor is operated at atmospheric pressure. The premixed fuel–air mixture is injected with an inlet velocity of $90 m/s$ and a temperature of $573 K$. The walls of the combustion chamber are made of synthetic quartz glass with a thickness of $8 mm$. In the streamwise direction, the combustion chamber walls consist of two 200 mm segments with narrow ($2 \times 5 mm$) flanges of stainless steel. The burner base plate is 10 mm thick and made of stainless steel. Constant values are used for the density of Quartz glass and stainless steel (2200 kgm^{-3} and 1750 kgm^{-3}, respectively). Isothermal boundary conditions are used at the outside walls of the solid. The thermal activity ratio for the current case is about 2×10^{-3}.

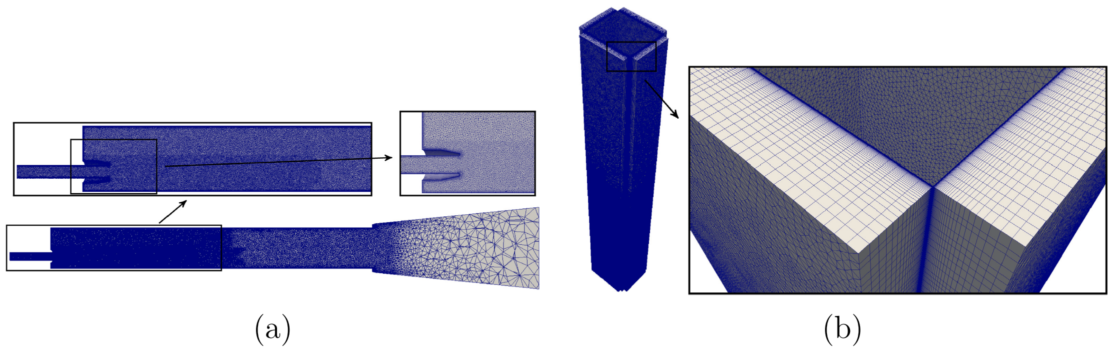


Figure 5.22: (a) The unstructured mesh for the fluid simulation consists of about 9.13 M elements. The cell size in the inlet pipe and the flame region is about 0.08d and gradually coarsens in the downstream regions towards the outlet. A total number of 10 prism layers are added at all wall boundaries in order to capture the strong gradients in the boundary layer. (b) The mesh for the structure consists of 13.44M prism elements. A total of 40 elements are distributed in the direction normal to the heat flux following a log law with a growth rate of 1.15. The nodes at the fluid–solid interface are matching in order to avoid interpolation errors.

uniform temperature distribution. The temperature field in the solid is computed under consideration of the heat flux from the fluid domain. The new wall temperature distribution from the solid domain is used as a boundary condition in the fluid domain. The loop is continued until the exchanged fields are converged. The study is divided in two main parts. Firstly, the temperature distributions obtained by the DHT approach are used to obtain RANS solutions of the flow field. Then, the influence of external walls on the gas temperature inside the combustion chamber

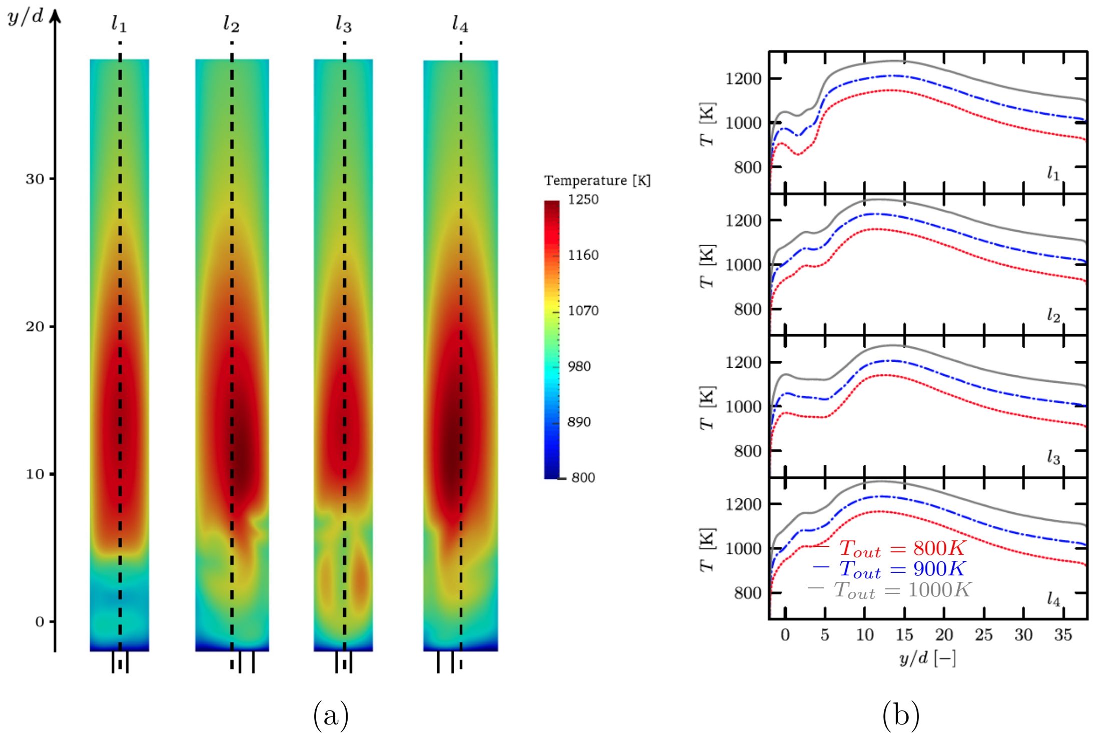


Figure 5.23: (a) The location of the inlet pipe is indicated. l_1 corresponds to the wall close to the recirculation zone, further away from the jet exit while l_3 is the short wall close to the jet exit. (b) Profiles of wall temperatures along l_1-l_4 for the different outside wall boundary conditions.

is investigated. Results are compared with the baseline case with fixed temperature at walls.

Figure 5.23 shows the temperature distribution at the fluid–solid interface when the temperature of the quartz glass fixed to $900K$. A significant spatial variation of the temperature at the interface is observed. The deviation of the temperature at the inner wall seen by the fluid is significant with a maximum deviation of about $\sim 350K$ between hot and cold regions. The comparison for the temperature range from 800 to 1000 K of the outside wall boundary is presented in Figure 5.24. When different temperatures are applied to the outer solid wall, the overall gas temperature inside the combustion chamber and the prediction of the flame length are affected. In the fluid part, the cold region is hardly affected by the variation of the outside wall boundary condition and also the flame front does not show any sensitivity to the wall condition. However, in the hot regions and the recirculation zone, the influence is the highest and the temperature reduces nearly linear with the wall boundary values. Regarding the solid, the profiles for the fluid domain are extended by addition of the temperature development across the chamber walls. The conduction through the walls leads to an almost parallel temperature development in the walls for the different boundary conditions. Although, the temperature

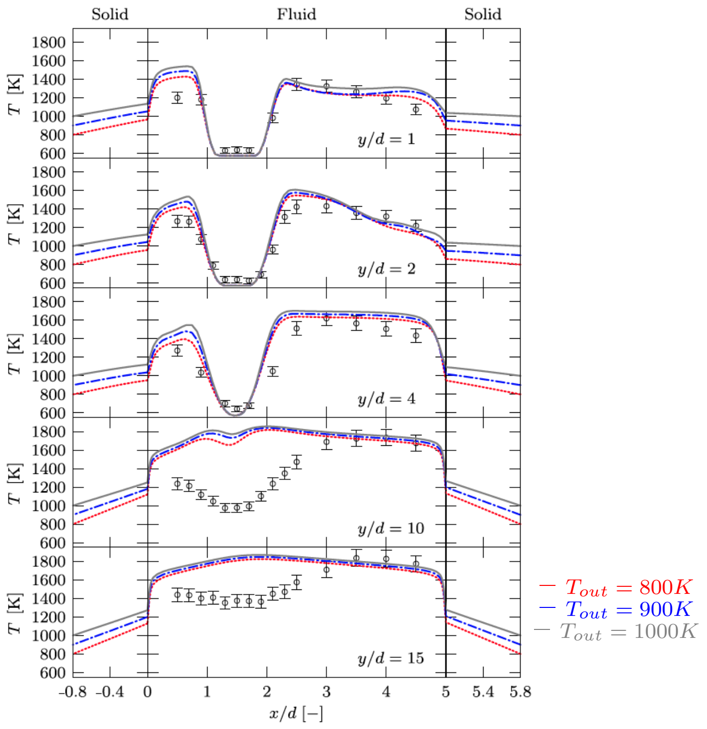


Figure 5.24: Influence of the outside wall boundary condition on the temperature distribution in fluid and solid. The profiles for the fluid domain are extended by addition of the temperature development across the chamber walls.

development is not exactly linear. This has the effect that the 100 K temperature difference at the outer walls between the different cases is reduced at the inner walls seen by the fluid.

5.5 Summary and remarks

In this chapter, a method to address conjugate heat transfer problems has been described. This method considers the use of a partitioned coupling approach to simulate a fluid interacting with a solid through a common interface in which thermal energy is exchanged. The heat transfer between a thin plate and an incompressible flow is used to validate the methodology. After that, this methodology is applied to analyse a steel flat plate cooled by an air stream at high Reynold number. Finally, the results of the application of this approach to two engineering applications using LES are summarised. The first case considers the investigation of an impinging flame configuration. In the second case, the heat loss of a confined turbulent jet flame is analysed.

In order to validate the coupling methodology, the temperature distribution obtained using the coupling approach described here is compared against analytical solutions. The results obtained have shown an excellent agreement with the theory. Continuity of the transmission conditions (temperature and the heat flux) is achieved throughout the coupling interface. The effect that the properties of the solid enforce in a fluid is also analysed. Particularly, the effect of modifying the Biot number and the thermal diffusivity inside a solid is studied. The results show that: (1) given a Biot number, an increase of the thermal activity ratio reduces the simulation time; and (2) for a particular solid, numerical simulations with the same Biot number can be considered physically equivalent. These results are applied to the case of a steel flat plate cooled by an air stream at high Reynolds number. In this case, the air coolant entering is assumed to have a Reynolds number of $\sim 4 \times 10^6$. Additionally, real properties are applied to the flat plate. Under these assumptions, an important disparity between the time scales exists (about four order of magnitude). By modifying the thermal activity ratio, the entire cooling process can be successfully simulated. The results show the validity of the iterative algorithm applied as well as the possibility of enhancing the response of a solid when interacts with a fluid.

To conclude, the iterative coupling approach described here has been successfully validated and applied to engineering applications. Additionally, the study of the properties of the solid allows accelerating the coupling process in orders of magnitude maintaining the physical properties. The ongoing work considers the use of this methodology in the solution of more general applications, its validation, as well as the investigation of its limitations.

6

Parallel Performance Analysis

In this chapter, a *load balance strategy* to run partitioned multi-physics applications on extreme scale architectures with an optimal parallel performance is presented. This strategy provides a detailed analysis of the influence of the assignment of the processors to the partitions and provides an expression that relates the performance metrics of each partition with the overall parallel efficiency of the coupled simulation.

The main contribution of this chapter is to present a load balance strategy for partitioned multi-physics applications. This strategy is based on a suitable resources distribution in which the parallel performance metrics for the overall multi-physics application are considered. The details related to this load balance strategy are described in the rest of this chapter as follows. In the first section, a general introduction of the load balance problem for partitioned multi-physics applications is presented. The metrics used to evaluate the parallel performance of bi-coupling schemes, and the proposed load balance strategy are detailed in Section 6.2. Relevant results are presented in Section 6.3, where the parallel performance of the confined premixed jet flame presented in Section 5.4.3 is analysed.

6.1 Introduction

In general, the performance of a parallel application is strongly related to two parameters, concurrency and load imbalance [97]. The concurrency refers to the number of tasks that can be executed simultaneously at any given time. The load imbalance is consequence of assigning different amount of workload to the available processors. In the particular case of partitioned multi-physical applications, a factor that can introduce limitation into the concurrency is the data dependency among the field components. Such dependencies can be a source of overhead since some components can be idle while others are involved in finalizing its own calculations. Regarding the load imbalance present in multi-physics applications, it is associated to the computational operations arising from the interaction among the field components (partitions). The main operations to consider here are the exchange of data among the field components, and also those operations related to the coupling scheme used to achieve the targeted convergence in the overall system.

In general, an improvement of the parallel performance of any application can be achieved when the idle-time produced by load imbalance is reduced at the lowest possible level, i.e, enhancing the load balance [98]. In the particular case of partitioned multi-physics applications as those studied here, the load balance can be associated to the coupling scheme. For instance, in the parallel coupling scheme, it is possible to assume that the field components running concurrently should finalize their execution at the same time. Nevertheless, this assumption is not always fulfilled. Sometimes while the slowest component finishes its execution, the fastest component waits. A possible solution to this limitation is to redistribute the load into each component, so that the idle time is minimized. In this respect, two alternatives have been identified, the resources distribution and the data distribution [99]. In the first case, the idea addresses *how to assign the available processors resources* among the components. In the second case, the goal is *how to achieve a balance in the data distribution between processors* so that, the load is balanced not only for each component, but also for the whole system. Thus, in order to maximize the parallel performance of a partitioned multi-physic application, a bottleneck that must be overcome is how to achieve a load distribution, with the computation completed in optimal execution time.

Examples where the load distribution have been considered can be found on fluid-structure-acoustic (FSAI) problems [100], where strong scaling analysis is performed for different stages of a coupling implementation. A similar example but applied to Smoothed Particle Hydrodynamics (SPH) simulations is found in [18]. Another example where exchange times between parallel solvers as function of the number of cores for a conjugate heat transfer (CHT) problem can be found in [55]. Further examples applied to CHT problems can be found in [16] where the analysis of a coupled combustion chamber shows the impact of imbalanced repartitions of cores among the solvers; and in [5] where the parallel and staggered coupling schemes are applied to the study of a cooled turbine blade. It is worth noting that this example claims that there exists a relation in the distribution of the processors that encourages the load balance for a parallel coupling scheme. Such relation associates the execution times T_s and T_f of the solid and the fluid solvers, respectively, with the number of processors p_f and p_s assigned to each solver, so that $p_s/p_f = T_s/T_f$. A last example dealing with the distribution of processors is related the performance for coupled climate systems models [101]. In this case four physical components are combined to predict weather and climate along thousands of years. In this case, a rectangular packing method [102] is used to determine the best number of processors to be assigned to each component, and how to combine these components in order to maintain efficiency and the scalability of the whole system.

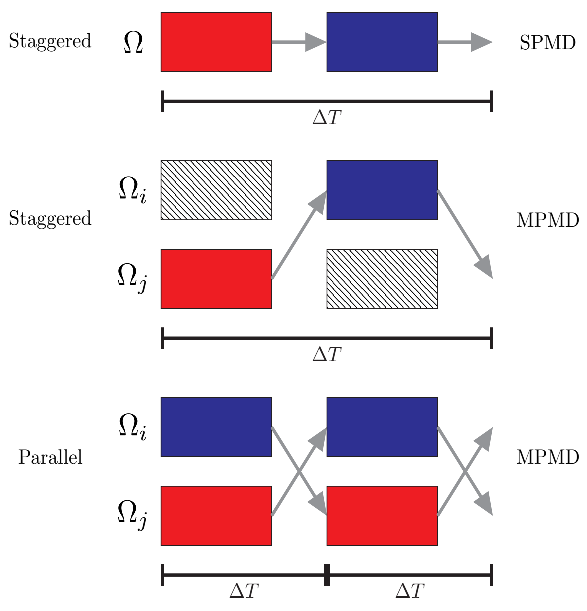


Figure 6.1: Schematic view of three coupling approaches. In staggered scheme, the solvers are executed one after the other while, in the parallel scheme, the solvers are executed concurrently to each other. The parallel execution model can involve either the use of all the processors by the solvers (Single Program Multiple Data mode), or sharing the processors between the solvers (Multiple Program Multiple Data mode).

Motivated by the objective of solving efficiently partitioned multi-physics problems described along this thesis on extreme scale architectures, the problem of *resources distribution* of multi-physics applications that only involve two coupled field components (bi-coupling schemes) is studied in this section. The goal is that each bi-coupling scheme can reach an optimal load balance for the overall multi-physics application. The main purpose is to analyse the parallel performance of each component field, but considering additionally the constrains enforced by the bi-coupling schemes. In this way, such constrains can be used to relate the run time of the component fields, and to establish a load balance for each bi-coupling scheme. In addition, a complementary restriction should be imposed in order to achieve an optimal configuration for the load balance. This restriction can arise from the resources distribution problem, i.e., how to distribute the processors among the component fields. In order to overcome this restriction, the strategy proposed here establishes that, for a given machine, the distribution of the processors resources is based on the metrics used to quantify the parallel performance of each component field. In particular, the interest is on how the execution time and the efficiency change when the number of processors vary for a *fixed* problem size. In this case, it is possible to determine the largest number of processors, which enables to maintain an optimal efficiency. As a result, this constraint imposed by the efficiency can be used to determine the resources distribution that enables reaching the optimal load balance for the overall multi-physic application, i.e., for the coupling scheme and for each component field. Summarising, the novel load balance strategy proposed here is based on a suitable resources distribution in which the parallel performance metrics for the overall multi-physics application are considered. The details related to this strategy are described in remainder of this chapter.

6.2 Methodology

Parallel metrics used to evaluate the performance of a coupling scheme along with the proposed load balance strategy are introduced in this section. Despite of the fact that, in practice, parallel performance metrics are widely used, appropriated definitions of them are hardly found. In this sense, load balance, efficiency, scalability, and speed-up are firstly defined before to introduce a geometrical interpretation for each of them. This novel interpretation allows understanding what the load balance means, its relation with the performance metrics, and finally, to be the basis of the proposed load balance strategy for coupling schemes.

6.2.1 Parallel performance metrics

The effectiveness of a parallel program can be evaluated by using metrics in performance evaluation. Additionally to the load balance (workload across processors), the study presented here makes use of the tracing as well as of widely used metrics: Efficiency, and Speed-up.

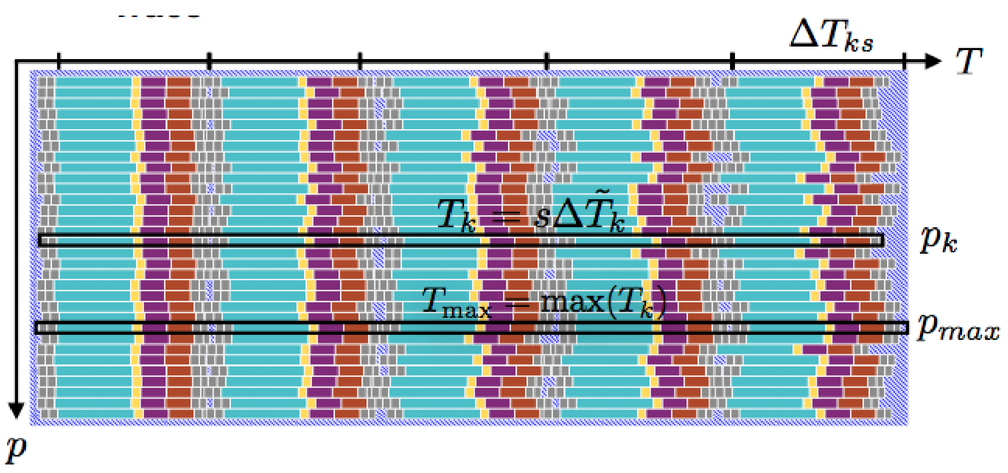


Figure 6.2: Time lines for ranks are placed from top to bottom and time flows from left to right. The colours in each time line state the task that is being executed at the time.

Tracing

A parallel activity trace shows the chronology in which the workflow of a parallel code is executed. It can be interpreted as a two dimensional diagram that shows how the parallel workflow deploys over time [46, 103]. The workflow is composed by different tasks taking place at a given time and place, see Figure 6.2.

Load Balance

For a parallel algorithm solving a problem of size n, the load balance L evaluates the fraction of utilization of a parallel system [103]. It is measured as the ratio between the time that the entire system of p processors are busy $T_{busy}(p, n)$ (running tasks or communication), and the total time that all processors are occupied $T_{total}(p, n)$ (running tasks, communication or being idle)

$$\begin{aligned} L &= \frac{T_{busy}(p,n)}{T_{total}(p,n)} = \frac{\mathrm{sum}(\mathbf{T}_{busy})}{p\max\left(\mathbf{T}_{total}\right)} \\ &= \frac{\sum_{\alpha=1}^{p} T_{\alpha,busy}}{p\max\left(T_{\alpha,total}\right)} = \frac{\sum_{\alpha=1}^{p}(\Delta T_{\alpha,run} + \Delta T_{\alpha,com})}{p\max\left(\Delta T_{\alpha,run} + \Delta T_{\alpha,com} + \Delta T_{\alpha,idle}\right)} \end{aligned} \tag{6.2.1}$$

where $\Delta T_{\alpha,run}$, $\Delta T_{\alpha,com}$, and $\Delta T_{\alpha,idle}$ stand for the time during a processor α is executing a task, communicating, or being idle, respectively. Both busy and total times are calculated as the sum of all the task executed by a given set of processors. In general, the measurements of these times are related to the machine architecture where the algorithm is being executed. It can also be related to the problem size, which, can be associated to the physics of the problem, or to the number of elements and vertices used to discretize the partitions, among others.

Work, Efficiency, and Scalability

A measure of the performance of a given algorithm is the work. For a parallel algorithm solving a problem of size n in a certain machine architecture, the work W can be determined by the product of the number of processors p used, and its running time $T = T(p, n)$ [104, 105]. Generally, the efficiency E of a parallel algorithm is defined as the ratio of the work of the sequential algorithm $W(1, n) = 1 \times T(1, n)$ to the work $W(p, n) = p \times T(p, n)$ of the parallel algorithm when it is executed using p processors [106]

$$E(p,n) = \frac{W(1,n)}{W(p,n)} = \frac{1 \times T(1,n)}{p \times T(p,n)}. \tag{6.2.2}$$

If an algorithm performs the same amount of work as the fastest known sequential algorithm, it can be called work-efficient (or just efficient).

The scalability is a property of a parallel algorithm to maintain the efficiency constant while the problem size and/or the number of processors change [105]. In particular, a strong scalable algorithm maintains the efficiency constant while the problem size n_0 is kept constant, and the number of processors is modified, i.e., $E(p, n_0) \equiv cst$. On the other hand, a weak scalable algorithm is the one keeping the efficiency constant only when the problem size increases along with the number of processors i.e., $E(p, n) \equiv cst$.

Speed-up

The speed-up S of a fixed problem size n_0 is used to measure the performance of a parallel algorithm when the number of processors p changes [42, 107]. It is given as the ratio of the running time $T(p_0, n_0)$ when p_0 processors are used, relative to the running time $T(p, n_0)$

$$S(p, n_0) = \frac{p_0 T(p_0, n_0)}{T(p, n_0)} = p\ E_{p_0} \tag{6.2.3}$$

Hereafter, unless otherwise indicated, it is assumed to solve the same problem size n_0, and then, it is omitted in the notation.

Load balance and trace area

This section introduces a *geometrical interpretation* of the parallel performance metrics introduced above. In the next sections, this novel interpretation is used to calculate values to achieve a coupled simulation with a suitable load balance. This methodology is premised on the idea that the trace can be seen as a rectangular coordinate system which can be used to depict the computation performed by a specific processor at any given time [46, 103], see Figure 6.2. Each axis in the coordinate system can be divided into units so that a point in the trace can be specified by a pair of coordinates (T, p), where T is the time, and p is the processor rank.

Thus, in the coordinate system T–p, the amount of time $\Delta T_{\alpha\beta}$ during a processor α is computing any task β is represented by an horizontal bar with length of $\Delta T_{\alpha\beta}$ and height of $\Delta p_\alpha = 1$. The set of tasks each processor executes is depicted by a horizontal time line composed by a sequence of bars. The length of this time line can be used to calculate the total computing time $T_\alpha = \sum_\beta \Delta T_{\alpha\beta}$ performed by each processor α. In general, a parallel solver uses a set of processors $\mathbf{p} = \{1, 2, .., p\}$ to compute a sequence of tasks $\boldsymbol{\tau} = \{1, 2, .., \tau\}$. Based on the above, the among of time that a parallel solver computes a sequence of tasks $\boldsymbol{\tau}$ using a set of processors $\mathbf{p}$ can be calculated as

$$\mathbf{T} = \{T_1, T_2, ..., T_p\} = \{\sum_{\beta=1}^{\tau} \Delta T_{1\beta}, \sum_{\beta=1}^{\tau} \Delta T_{2\beta}, ..., \sum_{\beta=1}^{\tau} \Delta T_{p\beta}\}. \tag{6.2.4}$$

The vector $\mathbf{T}$ contains the sum along each row of $\Delta T_{\alpha\beta}$, i.e., it holds the total length of the bars that lies at the time line of each processor. From the above, the load balance, Equation 6.2.1, can be rewritten as

$$\begin{aligned} L(p) &= \frac{\sum_{i=1}^{p} T_{i,busy}}{p \max\left(T_{i,total}\right)} \\ &= \frac{\sum_{i=1}^{p} \Delta p_i (\Delta T_{i,run} + \Delta T_{i,com})}{\sum_{i=1}^{p} \Delta p_i \max\left(\Delta T_{i,run} + \Delta T_{i,com} + \Delta T_{i,idle}\right)} = \frac{A_{busy}}{A_{total}}, \end{aligned} \tag{6.2.5}$$

and interpreted as the ratio of the area busy A_{busy} and the total area occupied A_{total}. The sum $\sum_{i=1}^{p} T_{i,busy} = \sum_{i=1,\beta=1}^{p,\tau} \Delta p_i \Delta T_{i\beta}$ can be seen as the area formed by the set of all the tasks (including communication and idle) with area $A_{i\beta} = \Delta p_i \Delta T_{i\beta}$. Similarly, the $T_{i,total} = \Delta T_{i,run} + \Delta T_{i,com} + \Delta T_{i,idle}$ and $A_i = \Delta p_i \max\left(T_{i,total}\right)$ is seen as the maximum area each processor can cover, so $p \max\left(T_{i,total}\right)$ can be constructed as the total area that the set of processors $\mathbf{p}$ can occupy.

Similarly to the load balance, the relative Efficiency can be defined from Equation 6.2.2 as

$$E_r = \frac{E_{\mathbf{p}_1}}{E_{\mathbf{p}_2}} = \frac{W_{\mathbf{p}_1}}{W_{\mathbf{p}_2}} = \frac{\sum_{i=1}^{p_1} T_i(p_1)}{\sum_{i=1}^{p_2} T_i(p_2)} = \frac{A_{\mathbf{p}_1}}{A_{\mathbf{p}_2}}, \tag{6.2.6}$$

where $W_{\mathbf{p}_1}$ and $W_{\mathbf{p}_2}$ define the work performed by parallel algorithm run with two different set of processors $\mathbf{p}_1$ and $\mathbf{p}_2$, respectively. When the algorithm is work-efficient, such algorithm must perform the same amount of work, i.e., $W_{\mathbf{p}_1} = W_{\mathbf{p}_2}$. This means that the areas $A_{\mathbf{p}_1}$ and $A_{\mathbf{p}_2}$ defined by each set of processors should be equivalent.

An ideal performance is achieved when a parallel algorithm that runs in p_0 processors and performs t_0 steps in time T_0 can perform the same t_0 steps in half of the time $T_0/2$ when two times processors $2p_0$ are used, $T_0/4$ for $4p_0$ processors, and so on. In line with the above, the work achieved for each of these cases is $W_{p_0} = (p_0)(T_0)$, $W_{2p_0} = (2p_0)(T_0/2)$, $W_{4p_0} = (4p_0)(T_0/4)$, and so on, respectively. For all cases, the areas are equivalent. In order to achieve this result, the parallel algorithm must be perfectly load balanced, with no latency, and with the communications completely overlapped [108].

6.2.2 Load balance strategy

The resources distribution problem of a coupling system involving two partitions as presented in Section 3.1.4 is analysed here. The goal is that the coupling system can achieve an optimal load balance. The approach is to analyse the parallel performance of each partition, but considering additionally the constrains enforced by the coupling scheme. The result shows that it is possible to find a resources distribution that allows achieving a load balance that equilibrates the parallel performance of the coupling system and its partitions.

The load balance strategy is based on two points. First, the constrains enforced by the coupling schemes can be used to create a link between the execution time of all partitions. Second, the metrics used to quantify the performance of the coupling schemes can be used to find a suitable distribution of the processors resources for each partition.

Coupling schemes and load balance

As said above, the load balance of a partitioned multi-physics system can be associated to the coupling scheme. In the parallel coupling scheme, usually the fastest solver must wait while the slowest finishes its execution. In the staggered scheme, a solver must wait while the other solver is being executed. For both schemes, the objective is the same, how to distribute the workload so that the computation can be completed within an optimal execution time. The workload and the execution time are related through the load balance, Equation 6.2.1. Such equation establishes that the load balance can be calculated from the execution time of each task across all processors.

Figure 6.3 sketches the trace for each of the coupling schemes analysed here. It shows the chronology of the execution of the tasks composing the workload in

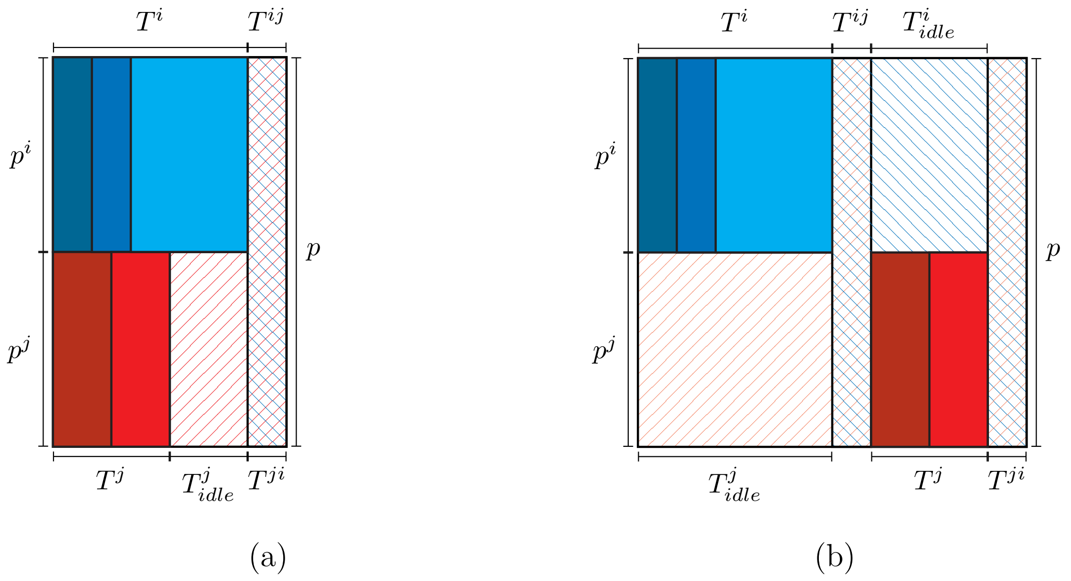


Figure 6.3: Coupling schemes traces: (a) parallel, and (b) staggered. The total number of processors $p = p^1 + p^2$ assigned to the whole coupled system $\Omega = \Omega^1 \cup \Omega^2$ are divided between the partitions $\{\Omega^1, \Omega^2\}$ involved in the coupling. For the parallel scheme, the total execution time T_{tot} can be approached either as $T^1_{tot} = T^1 + T^{12}$ or $T^2_{tot} = T^2 + T^2_{idle} + T^{21}$. Where T^1, T^2, T^{12}, T^{21}, and T^2_{idle} stand for the *median* of the execution time across all processors of the solver partition Ω^1 and Ω^2, the data exchange between Ω^1 and Ω^2, the data exchange between Ω^2 and Ω^1, and the idle time of the domain Ω^2, repectively. Likewise, for the staggered scheme, the total execution time can be calculated either as $T^1_{tot} = T^1 + T^{12} + T^1_{idle} + T^{21}$ or $T^2_{tot} = T^2_{idle} + T^{12} + T^2 + T^{21}$.

detail. Two set of tasks are performed. The first set is related to the solution of each partition. The second is constituted by operations that only take place when the partitions are coupled. The operations arising as result of the coupling are: localization, data exchange, and idle. The localization task is performed at the beginning of the coupling simulation, but its study is out of the scope of the present work. During data exchange, the interpolations, and sending/receiving data between partitions take place. The idle time mainly arises from the difference in the execution time of the solvers in the parallel scheme. On the other hand, in the staggered scheme, the limitation introduced by the lack of concurrence in the execution of the solvers is the source of the idle.

Once the tasks composing the workload of each coupling scheme are determined, the load balance can be calculated for each of them. Firstly, from Equation 6.2.5 is possible to establish that the load balance is given by

$$L = \frac{A_{busy}}{A_{total}} = \frac{\mathrm{sum}(\mathbf{A}_{busy})}{\mathrm{sum}(\mathbf{A}_{total})} = \frac{\sum_l A^l_{busy}}{\sum_l A^l_{total}}, \tag{6.2.7}$$

where the total area A_{total} and the busy area A_{busy} are calculated from the contribution A^k_{busy} and A^k_{total} given by each partition Ω^k. To begin with, the total area $A_{total} = \mathrm{sum}(\{A^1_{total}, A^2_{total}, ..., A^n_{total}\})$ covered by the partitions $\mathbf{\Omega} = \{\Omega^1, \Omega^2, ..., \Omega^n\}$ involved in the coupling can be calculated as

$$\begin{aligned} A_{total} = \sum_{l=1}^{n} A^l_{total} &= \sum_{k\in\mathbf{p}^1} \Delta p^1_k \max(T^1_{k,total}) + \sum_{k\in\mathbf{p}^2} \Delta p^2_k \max(T^2_{k,total}) \\ &\quad +... + \sum_{k\in\mathbf{p}^n} \Delta p^n_k \max(T^n_{k,total}) \\ &= p^1 \max(\mathbf{T}^1_{total}) + p^2 \max(\mathbf{T}^2_{total}) + ... + p^n \max(\mathbf{T}^n_{total}) \\ &= \max(\mathbf{T}_{total})(p^1 + p^2 + ... + p^n) \\ &= \max(\mathbf{T}_{total})\mathrm{sum}(\mathbf{p}), \end{aligned} \tag{6.2.8}$$

where $\mathbf{p}$ is a vector holding the total number of processors $p^l = \mathrm{sum}(\mathbf{p}^l)$ used by each partition Ω^l. On the other hand, $\max(\mathbf{T}_{total})$ is a *constant* value which corresponds to the maximum total time across all domains. Likewise, the area $A_{busy} = \mathrm{sum}(\{A^1_{busy}, A^2_{busy}, ..., A^n_{busy}\})$ during which the processors are busy is given by

$$\begin{aligned} A_{busy} = \sum_{l=1}^{n} A^l_{busy} &= \sum_{l=1}^{n} (A^l_{sol} + A^l_{cou}) \\ &= \sum_{l=1}^{n} (\sum_{k\in\mathbf{p}^l} \Delta p^l_k T^l_{k,sol} + \sum_{k\in\mathbf{p}^l} \Delta p^l_k T^l_{k,cou}) \\ &= \sum_{l=1}^{n} (\mathrm{sum}(\mathbf{T}^l_{sol}) + \mathrm{sum}(\mathbf{T}^l_{cou})), \end{aligned} \tag{6.2.9}$$

where $\text{sum}(\mathbf{T}^l_{sol})$ and $\text{sum}(\mathbf{T}^l_{cou})$ stand for the solution area and the coupling area defined by each partition Ω^l.

Now, from Equations 6.2.7, 6.2.8 and 6.2.9, along with the trace depicted on Figure 6.3, it is possible to obtain the load balance for each coupling scheme as

$$\begin{aligned} L = \frac{A_{busy}}{A_{total}} &= \frac{\sum_{l=1}^{n}(\text{sum}(\mathbf{T}^l_{sol}) + \text{sum}(\mathbf{T}^l_{cou}))}{\text{sum}(\mathbf{p})\max(\mathbf{T}_{total})} \\ &= \frac{\text{sum}(\mathbf{T}^i) + \text{sum}(\mathbf{T}^{ij}) + \text{sum}(\mathbf{T}^j) + \text{sum}(\mathbf{T}^{ji})}{(p^i + p^j)\max(\mathbf{T}_{total})} \\ &\equiv \frac{A^i + A^{ij} + A^j + A^{ji}}{A_{total}}, \end{aligned} \tag{6.2.10}$$

where, A^i and A^j define the solution areas, while A^{ij} and A^{ji} stand for the coupling areas of partitions Ω^i and Ω^j.

It is worth highlighting that when the coupling time is small relative to the calculation time, i.e., $T^{ij} << T^i$, the load balance can be approximated as

$$\begin{aligned} L &= \frac{A^i + A^{ij} + A^j + A^{ji}}{A_{total}} \\ &\approx \frac{A^i + A^j}{A_{total}}. \end{aligned} \tag{6.2.11}$$

Coupling schemes and performance metrics

The load balance of a coupled system can be based on the analysis of the computational and coupling times for a range of processors configurations [101, 109]. This analysis allows optimizing the global iteration time required to solve a time step of the coupled problem. Additionally to the analysis of the processors configurations, the static load balance strategy presented in this thesis considers the parallel performance metrics for the overall coupling system as well as the performance of each field component.

The Equation 6.2.11 establishes a general relation between the partitions and the total run time for each coupling scheme. Under the assumption of *ideal* parallel algorithm ($E \equiv 1$), an expression for the run time of each partition can be obtained. Thus, it is possible to relate the total run time of the schemes with the distribution of processors in each partition. This can be performed by the assumption that each partition can maintain the efficiency constant. It means that, given a problem of size n^i_0 and a set of processors $\{p^i_1, p^i_2, ..., p^i_m\}$, the work must be constant

$$A^i = p^i_1 \cdot T^i_1(p^i_1, n^i) = p^i_2 \cdot T^i_2(p^i_2, n^i) = ... = p^i_m \cdot T^i_m(p^i_m, n^i). \tag{6.2.12}$$

Thus, under the assumption of an ideal parallel algorithm, the work A^i defined by an arbitrary point (T^i_1, p^i_1) can be used to define a general relation for the running time T^i of a partition as

$$T^i(p^i, n_0^i) \quad = \quad \frac{A^i}{p^i}. \tag{6.2.13}$$

Now, for the coupling schemes depicted on Figure 6.3, under the supposition that (1) the coupling time is almost negligible, and (2) the parallel solver used by each partition can mantain the efficiency constant, it can be found that the load balance can be achieved when

$$L(p^1, T^1, p^2, T^2) = \frac{A^i + A^j}{A_{total}} = \frac{p^1 T^1 + p^2 T^2}{p\ T_{\max}} \tag{6.2.14}$$

where $p = p^i + p^j$ is the total number of processors, and $T_{\max}$ is the value given by the maximum total time across both partitions. This equation makes use of the efficiency to associate the execution time T^i of each partition with the total execution time $T_{\max}$ and the distribution of the processors resources $p = p^i + p^j$ of the coupling scheme. As demonstrated above, Equation 6.2.12, the areas A^i and A^j are constants into the range $[p_{\min}, p_{\max}]$ within which $E^i = 1$ and $E^j = 1$. As result, the load balance mainly depends on the relation between the total execution time $T_{\max}$, and the distribution of the processors p.

Coupling schemes and total run time

Figure 6.4 depicts an example of the behaviour that it is possible to find when the number of processors allocated to perform the solution of each coupling scheme is modified. The aim is to show the relation between the partitions and the processors with an example solved using the parallel and the staggered schemes. It consists of a domain Ω divided into two identical partitions Ω^1 and Ω^2 where Equation 6.2.14 is applicable. Each partition deals with the same problem of size n_0 using a number of processors p^1 and p^2, respectively.

Parallel scheme In general, for this scheme, the fastest solver must wait while the slowest solver finishes its execution. As a result, the maximum time $T_{\max}$ across the whole system is given by the execution time T^s of the slowest solver. Thus, from Equation 6.2.14, an optimal load balance $L \equiv 1$ can be achieved when

$$\begin{aligned} L = \frac{p_0^s T_0^s + p^f T^f}{p^{sf} T_0^s} \equiv 1 \rightarrow \frac{p^{sf}}{p_0^s} &= 1 + \frac{p^f T^f}{p_0^s T_0^s} \\ \frac{p^{sf}}{p_0^s} &= 1 + \frac{A_0^f}{A_0^s}, \end{aligned} \tag{6.2.15}$$

where p^f and T^f stand for the processors and run time of the fastest solver, and $p^{sf} = p^s + p^f$. It is worth noting that the optimal number of processors to be

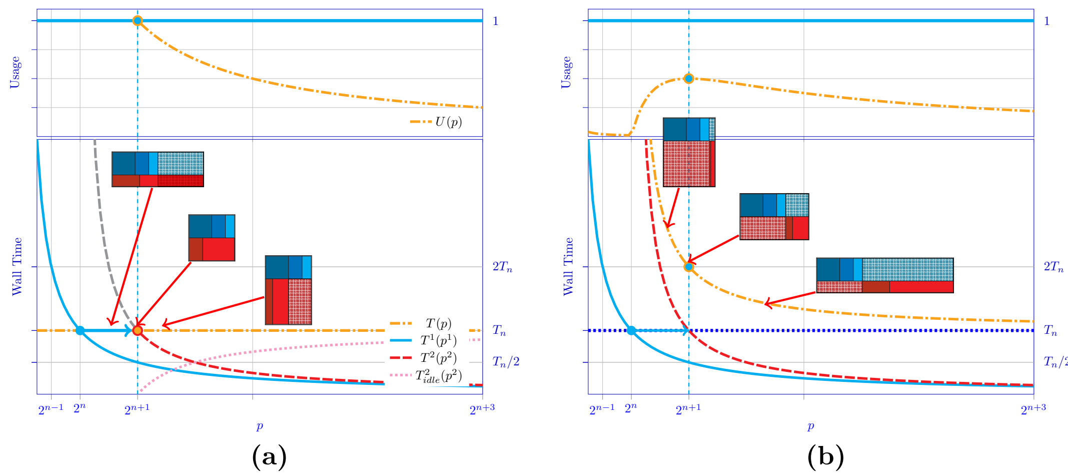


Figure 6.4: Usage and Wall time as function of the total number of processors for (a) parallel, and (b) staggered schemes. The total execution time (dash-dotted lines) is constituted by the composition of execution times in each partition (dash and dotted lines). In the parallel scheme, the execution time $T^i = T_n$ of the partition Ω^i fixes the value of the total execution time T of the coupling. In the staggered scheme, the total execution time T of the coupling is calculated as the sum of the execution time $T^i = T_n$ of the partition Ω^i, and the execution time $T^j = T^j(p)$ of the partition Ω^j. In both schemes, it is possible to achive an *optimum* load balance $L(p_o)$. Whether less processors that the optimum p_o are used, then idle time is added to the coupling. The same situation takes place when more processors that the optimum p_o are used.

assigned to the fastest solver is given by the product of the ratio of areas A^f_0/A^s_0 and the processors assigned to the slowest solver.

For the particular case depicted on Figure 6.4(a), a perfect load balance is expected when the same number of processors $p/2$ is assigned to each partition, i.e.,

$$\frac{p}{p^1} = 1 + \frac{A^2}{A^1} = 2 \quad \rightarrow \quad L = \frac{(p/2)T^1 + (p/2)T^2}{(p/2 + p/2)\ T^1} = 1. \tag{6.2.16}$$

Staggered scheme For an optimal load balance, Equation 6.2.14 can be interpreted as having the total calculation area *completely* covered by the calculation areas defined by all partitions, i.e., $A_{total} \equiv A^1 + A^2$. As a result, the area A_{idle}, arising from the difference between the execution time of the fastest and slowest partitions, has been practically eliminated from the parallel scheme. The situation in the staggered scheme is different. In this case, it is not possible to achieve an ideal load balance, due to the idle time, since a solver must wait while the other is running. As a consequence, the total calculation area always contains a fraction of idle. To overcome this limitation, a search for the extremum values (maximum or minimum) on the load balance can be made. A necessary condition for the load balance $L(p)$ to have a relative extremum value at p_{ext} is $dL(p_{ext})/dp = 0$. In order to find such extremum value, it is necessary to establish an expression for $T_{\max}$. For this scheme, the maximum total time across partitions can be calculated as the sum of the execution time of all solvers, i.e., $T_{\max} = T^1 + T^2$, since they are executed one after the other.

Based on the above, the load balance for the staggered scheme can be calculated as

$$L(p^1, p^2) = \frac{p^1 T^1 + p^2 T^2}{(p^1 + p^2)(T^1 + T^2)}. \tag{6.2.17}$$

This expression can be interpreted as the ratio of area busy and total area occupied for a particular configuration of processors $p = p^1 + p^2$. Now, when a set of processors $\{p_0, p_1, ..., p_m\}$ is considered, different ratios $\{L(p^1_0, p^2_0)$, $L(p^1_1, p^2_1)$, ..., $L(p^1_m, p^2_m)\}$ are expected. For all of them, the numerator $p^1 T^1 + p^2 T^2$ is constant. This is due to the fact that the efficiency of each partition is assumed to be constant, see Equation 6.2.12. In spite of this assumption, the total coupling area $(p^1 + p^2)(T^1 + T^2)$ is modified when the distribution of processors is changed. In order to show this, it is possible to assume that a set of processors $\{p^1_0 + p^2_0, p^1_0 + p^2_1, ..., p^1_0 + p^2_m\}$ is considered when the processors p^1_0 assigned to the domain Ω^1 are maintained constant. Thus, it is possible to demonstrate that when the expression for the load balance

$$L(p^1_0, p^2) = \frac{p^1_0 T^1_0 + p^2 T^2}{(p^1_0 + p^2)(T^1_0 + T^2)}, \tag{6.2.18}$$

is rewritten as

$$\begin{aligned} L(p) &= \frac{bc + a}{(b + p^2)(c + a/p^2)} \\ &= \frac{bc + a}{(p)(c + a/(p - b))}, \end{aligned} \tag{6.2.19}$$

where $p^1_0 T^1_0 + p^2 T^2 = bc + a$, $p = b + p^2$, and $T = c + a/(p - b)$, the extreme value p_{opt} calculated from $dL/dp = 0$ is given by

$$p_{opt} = b + \sqrt{ab/c} \quad \rightarrow \quad \frac{p_{opt}}{p^1_0} = 1 + \sqrt{\frac{A^2}{A^1}} \tag{6.2.20}$$

For the particular case depicted on Figure 6.4(a), the optimal load balance is expected when the same number of processors $p/2$ is assigned to each partition, i.e.,

$$\frac{p}{p^1} = 1 + \sqrt{\frac{A^2}{A^1}} = 2 \quad \rightarrow \quad L = \frac{a + bc}{(b + \sqrt{ab/c})(c + \sqrt{ac/b})} = \frac{1}{2}. \tag{6.2.21}$$

Wall time

Equations 6.2.15 and 6.2.20 show that using a suitable distribution of the available processors, an optimal load balance can be achieved. Furthermore, such distribution of processors are associated to a workload so that the computation can be completed within an optimal execution time.

Figure 6.4 shows this execution time T_{opt} as function of the optimum distribution of processors p_{opt}. Additionally, the effect of the use of different sets of processors $p = \{p_0, p_1, ..., p_m\}$, is also regarded in the parallel and staggered schemes. As stated above, the example consists of two identical partitions solving the same problem size n_0 by using equal set of equations.

When a total of 2^{n+1} processors are used, $p^1_{opt} = p^2_{opt} = p_{opt}/2 = 2^n$. As a result, the parallel scheme can achieve an ideal load balance ($L = 1$), while a maximum value ($L = 1/2$) is achieved by the staggered scheme, Equations 6.2.21 and 6.2.16, respectively. The assignation of 2^n processors to the partition Ω^1 establishes its running time as $T^1_n = A^1_0/2^n$, Equation 6.2.13. These values fix a point $(2^n, T^1_n)$ in the Figure 6.4. Likewise for the partition Ω^2, a point $(2^{n+1}, T^2_n)$ is defined, where $T^2_n = A^2_0/(2^{n+1} - 2^n) = A^2_0/2^n$. Thus, the total wall time in the parallel scheme is given by the point $(2^{n+1}, T^2_n) = (2^{n+1}, T^1_n)$, while for the staggered scheme the point is found in $(2^{n+1}, T^1_n + T^2_n) = (2^{n+1}, 2T^1_n)$.

Now, the effect of using different sets of processors $\{2^{n+1}+p^2_0, 2^{n+1}+p^2_1, ..., 2^{n+1}+p^2_m\}$ can be evaluated by the general expression of the wall time $T^2_n = A^2_0/(p - 2^n)$ in the partition Ω^2. Two situations are considered: (1) using less processors than the optimum, or (2) assigning more processors than the optimum.

In the parallel scheme, when less processors than the optimum are used ($2^n + p^2 < 2^{n+1}$), the total execution time T^1_n is exceeded. It is due to the fact that, in this situation the execution time T^2_n of the partition Ω^2 is slower than T^1_n. As a consequence, either less processors p^2 should be used, or a reallocation of the processors p^1 must be considered. On the other hand, when more processors than the optimum are assigned ($2^n + p^2 > 2^{n+1}$), the difference between the execution time of both partitions introduces the idle time $T_{idle} = T^1_n - T^2_n$. This difference can even reach the total execution time T_n when the processors assigned to Ω^2 are significantly greater ($p >> 2^{n+1}$). Finally, it is worth noting that as the idle time increases with the number of processors, the load balance decreases monotonically towards zero.

The behaviour of the load balance and the total execution time is different in the staggered scheme. As stated above, the load balance achieves a maximum value $L = 1/2$ when $p = 2^{n+1}$ and $T = 2T^1_n$. When $2^n + p^2 < 2^{n+1}$, the load balance decreases because the increase of the execution time T^2_n expands the total area $A_{total} = (2^n + p^2)(T^1_n + T^2)$. Likewise, the total area A_{total} expands when $2^n + p^2 > 2^{n+1}$. In both cases, the coupling areas $2^n\ T^2$ and $T^1 p^2$ occupy most of the total area, and are minimal only when $p = p_{opt}$.

6.3 Numerical tests

In this section, the load balance strategy described above is applied to the solution of the conjugate heat transfer problem presented in Section 5.4.3. This problem is selected with the aim of analysing three different aspects of the load balance strategy. First, the validation of the relations introduced in the previous section, and the effects of ranging the processors of an unique partition are discussed. Additionally to the wall time, the efficiency, the speed-up, and their relation with the load balance are also analysed. Next, the behaviour of the performance metrics referred above is analysed when the processors in both partitions are ranged under two situations: fixed problem size, and variable problem size.

6.3.1 Coupling problem

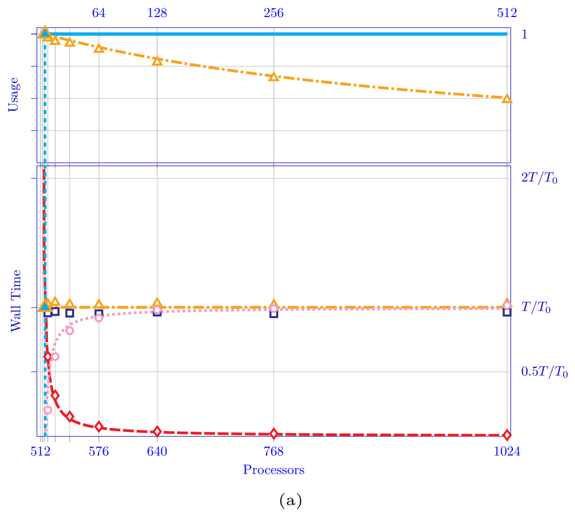
Fluid-solid thermal coupling

In this work, the confined turbulent jet flame presented in Section 5.4.3 is considered for a parallel performance analysis. The unstructured mesh for the fluid subdomain consists of 9.13 M tetrahedral elements, while the mesh for the solid subdomain consists of 13.44 M prismatic elements. The details related to the physical aspects of the problem can be found in [4].

6.3.2 Assessment of the load balance strategy

Figure 6.4 shows that each coupling scheme can achieve an optimum load balance for a particular distribution of processors, and such distributions are given by Equations 6.2.15 and 6.2.20. In this section, these relations are validated using the confined premixed jet flame described on Section 5.4.3. Figure 6.5 shows two numerical test where the distribution of processors $p = p^F + p^S$ has been ranged. The range for each partition is chosen so that each of them runs with the maximum efficiency, i.e., as close as possible to the ideal $E = 1$, Thus, for this case, the number of processors $p = p^F$ for the fluid partition is maintained constant to 512 processors, while processors for the solid partition are ranged as $p^S = \{4, 8, 16, 32, 64, 128, 256, 512\}$. Now, from the curves in Figure 6.5, it is clear that the load balance distribution, (Equations 6.2.15 and 6.2.20), agree with the measurements resulting from the wall time of the partitions.

In the parallel scheme, the load balance practically achieves an ideal load balance, i.e., $L \sim 1$, when the number of processors is $p = 516$, which corresponds to $p^F = 512$ and $p^S = 4$. As stated in Section 6.2.2, this takes place when the execution time of both partitions are equal, and then, the idle time is not introduced. Thus, as the number of processors p^S increases, the load balance drops, while the execution time of the fluid T^S decreases and the idle time T^{idle} increases. When the number



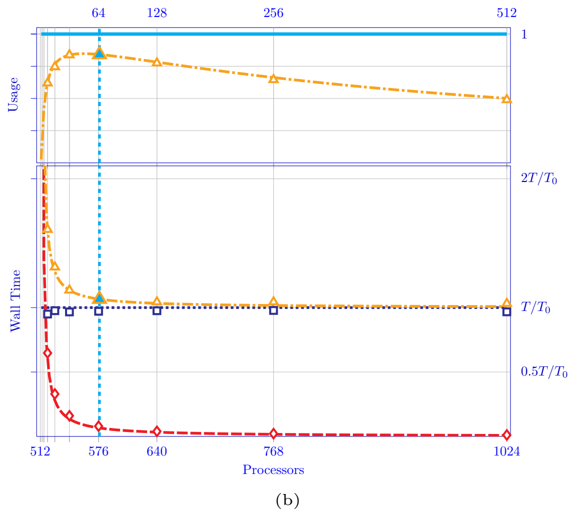


Figure 6.5: Load balance and wall time as function of the distribution of processors for (a) the parallel scheme, and (b) staggered scheme. The total number of processors $p = p^F + p^S$ ranges from 516 to 1024, with $p^F = 512$ and $p^S = \{4, 8, 16, 32, 64, 128, 256, 512\}$. The total wall time T (triangles) is composed of the wall time of the solid partition T^S (squares), and the wall time of the fluid partition $T^F = T_0$ when $p^F = 516$ (diamonds), which is taken as reference. The optimum distribution of processors (vertical dashed line) has been calculated by using Equations 6.2.15 and 6.2.20, along with the wall time measurements, which result on the relation of areas of approximately $A^F = 122A^S$, and where the optimum p^F is rounded to the nearest power of 2 i.e., $p^S_{opt} = 2^n$. With $n = \lfloor \log(p^F A^S_0/A^F_0)/\log(2) \rceil$, and $n = \lfloor \log((p^F)^2 A^S_0/A^F_0)/\log(4) \rceil$ for the parallel and staggeres schemes, respectively.

of processors of the solid is $p^S = 512$, the execution time of the solid is almost negligible and, as consequence, the load balance is $L = 0.5$. On the other hand, in the staggered scheme, when the distribution of processors is $p^F = 512$ and $p^S = 64$, the optimum load balance $L \sim 0.83$ of this problem is achieved. When the total number of processors is either smaller or larger than the optimum $p = 576$, the load balance drops again. It is worth noting that the decrease of the load balance is faster in cases where p is below the optimum than when p is higher than the optimum.

6.3.3 Performance metrics

The efficiency and speed-up for the setup described in the last section are shown in Figure 6.6. The efficiency compares the amount of *total* work that a parallel algorithm performed using different set of processors is conducted. Under the assumption that each partition maintains the efficiency constant, the area defined by each set of processors should remain constant, and the efficiency of each of them should be equal to one. In a coupled system, this can be different because the total area depends on the sub-set of processors used to execute each partition. For the setup described in the last section (where the processors allocated to a partition are

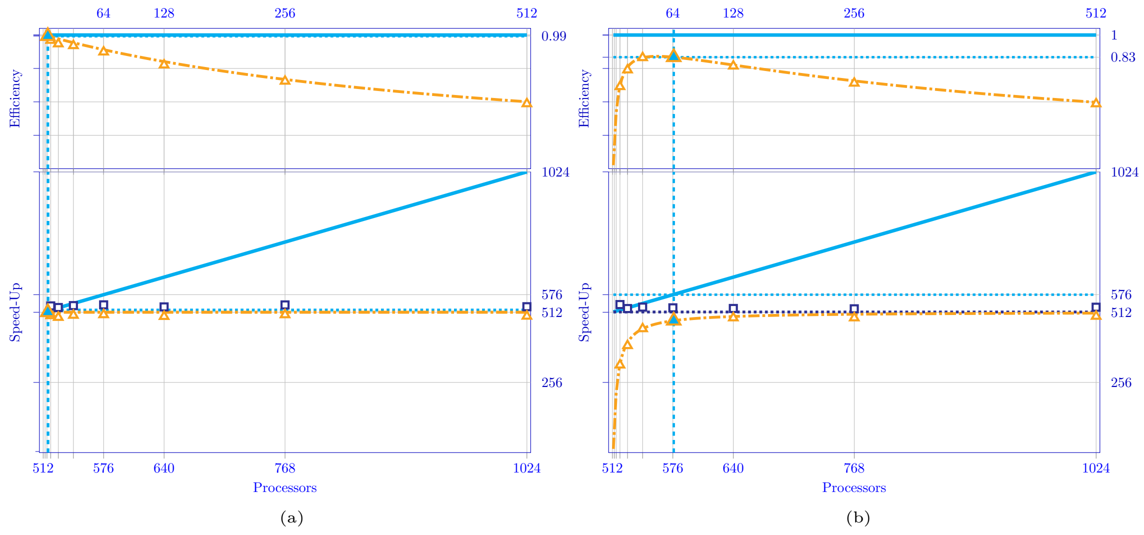


Figure 6.6: Efficiency and Speed-up as function of the distribution of processors of the overall coupling (triangles) for (a) the parallel scheme, and (b) staggered scheme. The total number of processors $p = p^F + p^S$ ranges from 516 to 1024, with $p^F = 512$ (squards) and $p^S = \{4, 8, 16, 32, 64, 128, 256, 512\}$. The speed up and the efficiency are compared with their ideal values (continuum blue lines) $E = 1$ and $S = p$, respectively. The optimum distribution of processors (vertical dashed line) has been calculated by using the wall time measurements and Equations 6.2.15 and 6.2.20. The corresponding values of efficiency $E = 0.83$, and speed up $S = 512$ are given by Equations 6.3.1 and 6.3.2.

maintained constant, while the processors for the other partition are ranged) a first chance here considered is to calculate the efficiency respect to the distribution of processors corresponding to the ideal load balance. Another chance considered is to use the same values employed to calculate the efficiency of a given partition as reference. In the first case, the efficiency is equal to one when the optimum distribution of processors is achieved, while in the latter case, the efficiency of the chosen partition is taken as reference. For the setup described in the last section, the values related to the fluid partition $A_0^F = p_0^F T_0^F$ are used as reference to calculate the coupling efficiency.

From the Equation 6.2.6, the efficiency for the parallel scheme can be calculated as $E^p = p_0^F T_0^F / (p\, T)$, which for the ideal load balance (Equation 6.2.15) is given by

$$E^p_{opt} = \frac{p_0^F}{p_{opt}} = \frac{1}{1 + A_0^S / A_0^F}. \tag{6.3.1}$$

These relations show how the efficiency of the fluid partition is affected by the coupling. When the ratio of areas tends towards zero $A_0^S / A_0^F \to 0$, the efficiency of the original partition is recovered, otherwise, the efficiency drops. On the other hand, due to the fact that the efficiency E^p is independent of the total time, the speed up of the parallel sheme S^p is equal to the number of processors assigned to the fluid partition, i.e., from Equation 6.2.3, $S^p = p_0^F$, see Figure 6.6(a).

Similarly to the parallel scheme, the efficiency of the staggered scheme can be calculated using the expression $E^s = p_0^F T_0^F/(p\ T)$, while the speed-up is given by $S^s = p_0^F T_0^F/T$, see Figure 6.6(b). Note that a maximum value of efficiency E^s_{opt} can be achieved when the distribution of processors is the optimum, i.e,

$$E^s_{opt} = \frac{p_0^F T_0^F}{p_{opt} T_{opt}} = \frac{1}{(1+\sqrt{A_0^S/A_0^F})^2}. \tag{6.3.2}$$

This expression is calculated using the Equation 6.2.20, and considering that the total time is given by $T = T_0^F + p_0^S T_0^S/(p - p_0^F)$. Under the assumption of negligible coupling time, this expression for the total time relates the total coupling area $p\ T$ with the area of each partition (see Figure 6.3)

$$(T - T_0^F)(p - p_0^F) = p_0^S T_0^S \to \frac{p\ T}{p_0^F T_0^F} = (1 + \frac{T_0^S}{T_0^F})(1 + \frac{p_0^S}{p_0^F}), \tag{6.3.3}$$

so that the optimum load balance is obtained by

$$\frac{p_{opt} T_{opt}}{p_0^F T_0^F} = \left(1 + \sqrt{\frac{A_0^S}{A_0^F}}\right)\left(1 + \sqrt{\frac{A_0^S}{A_0^F}}\right). \tag{6.3.4}$$

This expression indicates that, for an optimum distribution of processors, the areas defined by the partitions are distributed so that the ratio of running times, and the ratio of allocated processors are equivalent, i.e., $T_0^S/T_0^F = p_0^S/p_0^F \equiv \sqrt{A_0^S/A_0^F}$. From Equations 6.3.2 and 6.3.4, it can be interpreted as the efficiency, the area of the reference partition $p_0^F T_0^F$ to the ratio of the optimum coupling area $p_{opt} T_{opt}$, is equivalent to the area enclosed in a square of side $1 + \sqrt{A_0^S/A_0^F}$.

6.3.4 Weak scaling

Given a particular problem characterized by the ratio A_0^S/A_0^F, Equation 6.2.20 allows calculating the optimum distribution of processors $p = p^F + p^S$ for a staggered scheme. Thus, a change in the distribution of processors is expected only when such ratio is modified. On the other hand, Equation 6.3.2 indicates that a maximum value of efficiency E^s_{opt} can be achieved, and its value is maintained constant while A_0^S/A_0^F remains unchanged. In general, this assumption is preserved just as much as the efficiency of the partitions remain close to one. As a result, the efficiency calculated in the latter section remains constant despite of the fact that the number processors p_0^F used as reference changes. The effect of changing the value of the processors allocated for the fluid partition, is shown in Figure 6.7.

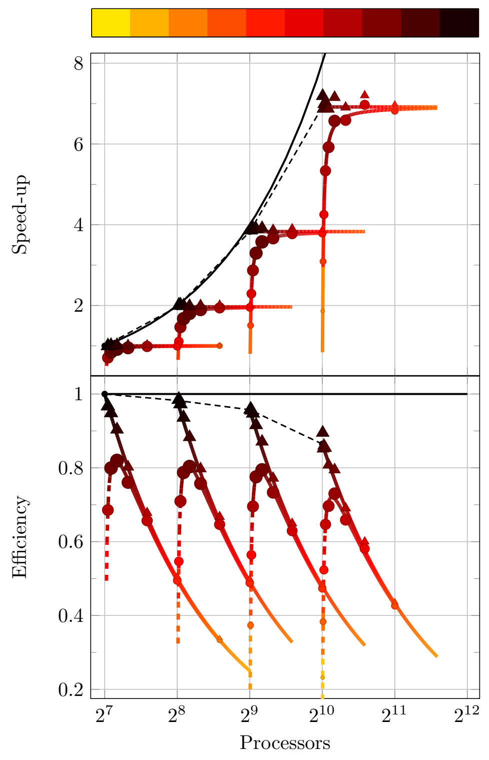


Figure 6.7: Speed up (top) and efficiency (bottom) as function of the distribution of processors for staggered scheme. The total number of processors $p = p^F + p^S$ ranges from 128 to 2048. Values p^F corresponding to the fluid partition (triangles) are taken as reference, while the processors p^S of the solid partition range from 4 to 1024. The speed up and the efficiency of the overall coupling (circles) are compared with their ideal values corresponding to the fluid partition (continuum black lines) $E = 1$ and $S = p$, respectively, as well as, with the expression given by Equation 6.3.4 (dashed curves). The optimum distribution of processors (vertical dashed lines) has been calculated by using the wall time measurements and Equation 6.2.20, while the corresponding values of efficiency $E = 0.84$, are given by Equation 6.3.2.

As in last section, each numerical experiment consists of modifying the number of processors of the solid partition p^S, while the processors of the fluid partition p_0^F are used as reference. Four numerical experiments are shown, $p^F = \{128, 256, 512, 1024\}$. They are chosen so that the efficiency of the fluid partition remains above 80%, taking as reference the execution time when $p^F = 128$. In the other hand, for each experiment, the number of processors of the solid partition p^S ranges from 4 to p^S, while the distribution of processors p ranges from $p^F + 2$ to $2p^F$. Note that, in general, the observations given in section 6.3.3 are fulfilled in each experiment. Initially, the overall efficiency increases due to the increase of p. This takes place until the ratios of running times, and allocated processors follow Equation 6.3.4, point where maximum efficiency is present. After that, as the total number of processors continues to rise, the efficiency decreases following the values of efficiency of the fluid partition. Due to the fact that in each experiment the fluid partition is taken as reference to calculate the efficiency, its speed up is maintained constant while p increases (T_F is constant). However, its efficiency decreases because of the increase of p.

6.4 Summary and remarks

This chapter presents a load balance strategy that takes into account an optimal distribution of processors for multi-physics applications solved by a partitioned coupling approach. It describes the influence of the assignment of the processors by means of expressions relating the performance metrics of each partition. The assignation is based on a suitable distribution of the available processors, which considers the performance metrics of each partition under the influence of the constrains established by the coupling approach. The results show that it is possible to find an optimal distribution of processors to achieve a load balance that equilibrates the parallel performance of the coupling system and its partitions. Firstly, it is shown that the load balance of a coupling scheme establishes a relation between its total execution time, the execution time of each partition, and how the total processors are distributed between partitions. This relation allows to assign an optimum number of processors to each partition so that an ideal load balance can be achieved in the parallel scheme. On the other hand, it is shown how to overcome the constrains imposed by the staggered scheme demonstrating that a maximum value of efficiency can be achieved.

Finally, in order to also evaluate the proposed load balance methodology, the confined turbulent jet flame presented in Section 5.4.3 is analysed. The scaling tests carried out indicate that the expected parallel performance for the different distribution of resources as well as the optimal performance of the coupled simulation is well represented by the expressions derived in this study. Therefore this strategy, can be used to assign the most appropriate distribution of processors to guarantee optimal performance. The proposed strategy holds under the following assumptions: (1) the coupling time is negligible, and (2) that the parallel solver used by each partition must maintain the efficiency constant. These results encourage the application of this methodology to other large-scale multi-physics applications including contact (Chapter 4), particles-fluid flow (Section 3.3.2), fluid-solid interaction problems, multi-fields couplings, and others, as well as large-scale architectures, and other coupling approaches.

7 Conclusions

In this work, the development, validation and use of a high-performance computing coupling tool for the solution of partitioned multi-physical simulations has been described. The emphasis has been placed on the efficient use of large-scale computers, but always considering the robustness and accuracy of the solutions.

Two of the main challenges related to the modelling of multi-physics applications by combining separate physical systems has been addressed in this thesis: the parallel coupling problem, and the efficient use of large-scale computers. In this sense, the thesis has been divided into three main parts:

Development A mathematical background related to coupling approaches used in the solution of partitioned multi-physical systems was presented in Chapter 2, along with details related to the computational development of the coupling tool PLE++ (Parallel Location and Exchange++) were given in Chapter 3.

Validation and use Two multi-physics applications were addressed in Chapters 4 and 5: contact between deformable bodies, and conjugate heat transfer, respectively.

Performance A parallel performance analysis for multi-physics applications was introduced in Chapter 6.

Main contributions of this study along with future challenges are presented in the following sections.

7.1 Summary of the thesis

Chapter 2 discussed the mathematical nature of the *coupling approaches* used in this thesis. The details related to the *iterative Dirichlet-Neumann approach* used to solve contact and conjugate problems were shown. Firstly, it was established that a suitable use of boundary conditions through the interface defined between pairs of partitions is necessary to find an accurate solution of a coupled system. In order to do that, *transmission conditions* arising from *heterogeneous domain decomposition methods* for non-overlapping partitions are necessary. It was shown that these conditions stablish that a given variable and its corresponding flux must be continuous through the coupling interface. As indicated, for each coupling simulation, these coupling variables can be determined using the weak form of the equations modelling each partition. For instance, for the advection-diffusion system with convection dominated flow in a partition, transmission conditions are the normal component of a convective vector, and a Robin-type condition. As shown, these variables are iteratively imposed as boundary conditions to each partition. Firstly, the projection of the convective vector is enforced as Dirichlet boundary condition in the inlet region of the coupling interface of a given partition. After that, in the other partition, Robin-type conditions are enforced in outflow boundaries. Using the procedure introduced in this chapter, the corresponding transmission conditions for contact between deformable bodies and for conjugate heat transfer problems were deduced in Chapters 4 and 5.

Chapter 3 presented an overview of the main challenges related to the implementation of the *coupling tool* developed in this thesis. The *Parallel Location and Exchange++* (PLE++) library is a coupling tool developed to deal with large-scale partitioned multi-physical simulations. Here, it was shown that the workflow of PLE++ can be divided in three main stages: (1) Defining the set of MPI communicators to be used by each partition. (2) Establishing peer-to-peer communication layouts between pairs of sub-domains allocated in different partitions. (3) Exchanging data between sub-domains identified as associated to the coupling. The first stage is necessary because for a coupled system using the *MPMD execution mode*, the physical solution for each partition is found independently (i.e., each solver is executed as a SPMD), so that a *internal* MPI-communicator must exist for each partition. During the second stage, the *parallel localization* allows identifying the pairs of processors related to the coupling. It is performed by detecting sub-domains allocated in different partitions which share common regions. These regions are identified using a hierarchical algorithm based on geometrical properties of the partitions. Broadly, a *global search* allows associating overlapped sub-domains allocated in different partitions. At the same time, a *local search* allows associating vertices of a local partition which are contained inside of elements allocated in a remote partition. At the end, these geometrical associations allow defining the *external* communicators to be used during exchange. It includes the interpolation of properties related to transmission conditions as well as the exchange of them between pairs of partitions Finally, two application cases were shown as examples. The first case addressed the parallel implementation of a coupling scheme for a confined premixed jet flame. In particular, part of the analysis presented here was limited to show the possible limitations of the coupling performance. The staggered approach used here requires that both partitions wait during the calculation stage of the other partition. It introduces a limitation in the maximum performance that can be achieved when the number of MPI processes are not selected appropriately. In the second case, the capability to perform heterogeneous simulations on GPUs and CPUs was considered. In this case, the GPU and Alya transfer the data throughout interconnection network and PCI-Express, while PLE++ performs the localization and mapping.

Chapter 4 described the development of a *novel* parallel algorithm to deal with *contact of deformable bodies*. Regions where the contact constrains take place are identified using the parallel localization implemented in the PLE++ library. At the same time, the algorithm makes use of an iterative Dirichlet-Neumann coupling approach for the contact resolution of the frictionless interaction between two deformable bodies. Broadly, Dirichlet boundary conditions are used in a partition to enforce the *non-penetration constrain*. After that, the *action-reaction principle* between two bodies is considered using Neumann boundary conditions in the other partition. These *transmission conditions* are enforced through the set of remote vertices on the surface of the deformable body, which are localized inside of any local element on the surface of the other partition. Firstly, *Dirichlet conditions* are enforced so that vertices can only move in parallel to each boundary face corresponding to the element in which each vertex is localized. The magnitude of this Dirichlet-type condition is calculated as the distance from each vertex to the boundary face of the element containing it. As a result of enforcing this slip condition, the stress field is updated through the contact interface, then the stress is interpolated from local elements to the remote vertices. Finally, once these values have been received in the other partition, they are enforced as a *Neumann condition*. The entire process can be repeated until the desired convergence is achieved. The validity of the approach presented here is shown through two cases. In the first case, *an elastic ball contacting with a rigid plane* is considered. The results showed a good agreement with the analytical solution for the contact traction distribution given by the Hertzian elastic contact theory. In the second case, *the impact of a rigid hemisphere against a plate* is studied. For this case, forces as function of time and velocities of the bodies are compared. Broadly, the plate is modelled as a transversally isotropic material, while the impactor is assumed as an isotropic linear elastic material with an impact energy level of 1.6J. The results obtained here agree well with those achieved by the commercial code Abaqus. Finally, a parallel performance analysis focused on the availability of the localization algorithm has been provided. The results suggests that a suitable selection of the processors distribution is necessary to achieve optimal location times. A deeper analysis of this key issue is left for future work.

Chapter 5 was focused on describing a method to address *conjugate heat transfer* (CHT) problems using a partitioned approach. Thus, the CHT problem was solved using domain decomposition methods, i.e., solving iteratively two independent problems, where the fluid partition is defined as the Dirichlet partition, while the solid partition is considered as the Neumann one. By means of this algorithm, the continuity of the *transmission conditions* (*temperature* and *heat flux density*) through the coupling interface is ensured at every time step. On the other hand, one of the main drawbacks in CHT problems is the *time-disparity*. Due to the physical nature of a CHT problem, the temporal scales between fluids and solids are usually very different. In this thesis, a method to deal with this problem has been proposed. It consists on synchronizing the physical time in fluid and solid by increasing the thermal activity ratio while the *Biot number* is maintained. The result achieved for *a constant air stream cooling a steel flat plate* shows that the entire cooling process can be successfully simulated. Finally, two practical applications are solved here. Firstly, the effect of the heat transfer condition at the solid wall on a *premixed impinging jet flame* was presented. The main results regarding heat transfer effects were summarized here. Different thermal conditions were imposed at the impinging plate and its effects on the flame dynamics, heat transfer, shear stress and wall-jet development were discussed in the context of large-eddy simulation. Secondly, the *heat loss of a confined turbulent jet flame in a labscale combustor* was investigated. In this case, the dual heat transfer approach was used in order to obtain the steady temperature distributions in the solid. For this case, one of the main results shows that a significant spatial variation of the temperature at the interface is observed.

Chapter 6 introduced a *load balance strategy* to run partitioned multi-physics applications on extreme scale architectures. The goal was to achieve an optimal balance in the data distribution so that the load is balanced not only for each component, but also for the whole coupled system. Here, it has been shown that, in a partitioned system using the MPMD execution mode, *concurrency* is related to the coupling scheme chosen, while *load balance* is associated to the amount of processors assigned to each partition. Thus, if an appropriate selection of the set of processors is done in the *parallel scheme*, then it is possible to remove completely the idle time arising from the difference between execution times of the solvers. As consequence, an *ideal load balance* can be achieved for the whole coupled system. In the case of the *staggered scheme*, it has been demonstrated that the idle time can only be reduced since at any time there exist a waiting solver. One of the main results has been to find analytical expression which can be used to measure the efficiency of a partitioned multi-physical application. This was done by introducing a *geometrical interpretation* for the *load balance*, and for the *performance metrics* (efficiency, scalability, and and speed-up). The interpretation introduced is based on the idea that the amount of time that a parallel solver computes a sequence of tasks using a set of processors can be associated to an *area* covering a trace (if it is seen as a rectangular coordinate system). The results show that for the *confined turbulent jet flame* presented in Section 5.4.3, the optimal performance of this coupled simulation is well represented by the expressions derived in this study and therefore, under some assumptions, it can be used to assign the most appropriate distribution of processors to guarantee optimal performance.

7.2 Future work

Based on the contributions summarized above, some future improvements are listed below.

Firstly, regarding the computational development of the PLE++ tool, numerical methods to deal with interpolations and accelerations methods should be considered. Interpolations methods which avoid the use of the computation mesh seems to be attractive in multi-physics simulations as fluid containing particles [110]. In particular, the use of radial basis meshless methods [111, 112] could be considered in future versions of PLE++. Regarding accelerations methods, the quasi-Newton least-squares (QNLS) [19], or similar methods widely used in fluid-solid interaction problems, could be implement in order to improve the convergence in contact and conjugate problems.

With reference to the multi-physical systems addressed here. For the contact of deformable bodies, a deeper parallel performance analysis must be considered. In particular the localization algorithm used for the contact search must be included in the load balance strategy proposed in this thesis. Regarding conjugate heat transfer problems, the method proposed to deal with the time-disparity must be validated against numerical cases which consider transient states, as well as cases considering different properties for the fluid and solid partitions.

Finally, the load balance strategy introduced in this thesis must be validated in different computer architectures, as well as for more general multi-physical applications. In particular, one interesting challenge is related to the application of this strategy to simulations involving more that two partitions (as in the case of climate modelling). Another important case is the validity of the strategy when the coupling involve overlapping partitions. In this case, the volume of data can be important since exchange and interpolations can represent an import source of time. To conclude, extensions to hybrid architectures could be also considered.

Acknowledgements

The research leading to these results has received funding through the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7, 2007–2013) under the Grant agreement No. FP7-290042 for the project COPA-GT

A

Solid partition parameters

In this section, the effect of the variation of the Biot number and the thermal diffusivity inside a solid is analysed. The solid under study is the one considered in the thermal coupling described in Section 5.4.1.

For a conjugate heat transfer problem, the coefficient of convective heat transfer h is a quantity distributed on the coupling interface. At the same time, the thermal activity ratio K, defined as $\kappa_f/\kappa_s(\alpha_s/\alpha_f)^{1/2}$, relates the thermal diffusivity of the fluid and the solid. This ratio characterises the thermal behaviour of a solid interface, and leads to two opposite situations. When $K \to \infty$, the maximum level of fluctuations are expected at the interface. On the other hand, when $K \to 0$, the interface behaves like an isothermal wall with small fluctuations to be expected [113].

In the particular case analysed in Section 5.4.1. the conductivity and thermal diffusivity of fluid and solid have been taken as constant and equal, so that the thermal activity ratio K is fixed to one and then low level of fluctuations would be expected. This agree well with the fact that the time scale ratio (Equation 5.3.6) is around seven, i.e., $t_s \sim 7t_f$. In general, this indicates that, the solid must be subject to the influence of the fluid during a considerable amount of time to have changes. In order to change this behaviour without modifying the properties of the fluid, the thermal diffusivity of the solid can be altered.

Biot Bi and Fourier Fo numbers are dimensionless parameters controlling the physical behaviour of the solid. A relation between these numbers is found in Equation 5.1.6. The product of both of them is equivalent to the ratio given by the physical time t and the thermal time constant τ. From this relation, it is possible to conclude that τ is controlled by the Biot number Bi and the diffusivity α , since $\tau = L^2/(Bi\ \alpha)$. Considering that the geometry of the problem, and the fluid properties are fixed, the Biot number can only change as consequence of varying the conductivity of the solid. However, this modification results on a change in the physical behaviour of the solid, since this number controls how the temperature is distributed in space. Thus, when the main interest is to maintain the physical

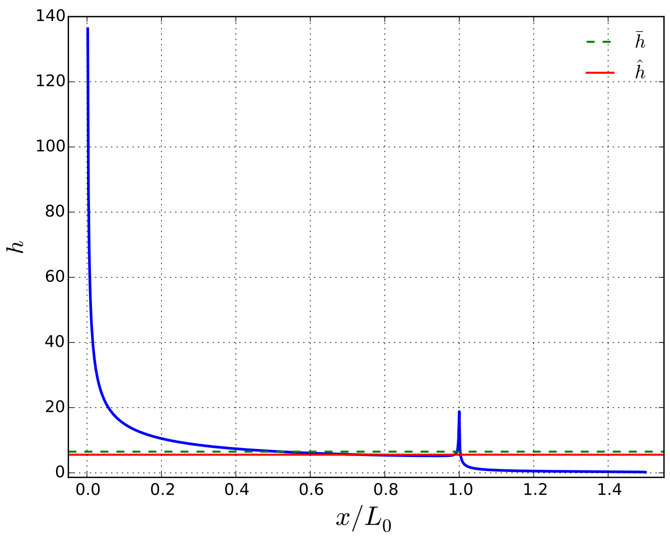


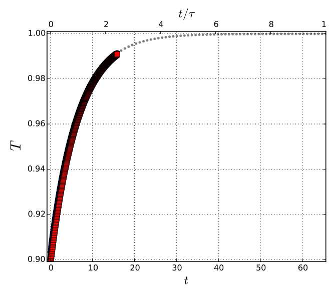
Figure A.1: Heat transfer coefficient interface distribution. $\hat{h} = 5.559$, $\bar{h} = 6.495$, $\kappa_f/\kappa_s = 1$, $\alpha_f/\alpha_s = 1$, $Bi = \hat{h}\Delta/\kappa_f \approx 1.118$, $\alpha = 1.408 \times 10^{-3}$, $\tau_0 = \rho_f c_{pf}\Delta/\hat{h} \approx 6.346$.

behaviour of the fluid and the solid, a variation of the diffusivity in the solid can be used.

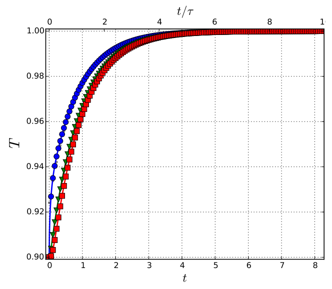
The effect of modifying the Biot number and the thermal diffusivity inside a solid is analysed here using five cases, see Table A.1. The first case corresponds to the same properties as the fluid studied in Section 5.4.1, so that $\alpha_s/\alpha_f = 1$, and $\kappa_s/\kappa_f = 1$. In the next two cases, the density ρ_s is modified. As a result, the thermal diffusivity α_s and the thermal time constant τ changes, while the Biot number is kept constant. In the last two cases, the thermal time τ is maintained constant by modifying the conductivity κ_s. Bi and α_s change for these two cases. Finally, for the five cases, the convective heat transfer is approximated as the mean of its distributed value h obtained from the coupled simulation, see Figure A.1.

	ρ_s $[kg\ m^{-3}]$	κ_s $[W\ m^{-1}\ C^{-1}]$	α_s $\kappa_s/(\rho_s c_{ps})$	τ $\rho_s c_{ps} L_s/h$	Bi hL_s/κ_s	K^2 $(\kappa\rho c_p)_f/(\kappa\rho c_p)_s$
1	3.528×10^2	4.969×10^{-1}	1.408×10^{-3}	6.347×10^0	1.118×10^0	1×10^0
2	3.528×10^1	4.969×10^{-1}	1.408×10^{-2}	6.347×10^{-1}	1.118×10^0	1×10^1
3	3.528×10^3	4.969×10^{-1}	1.408×10^{-4}	6.347×10^1	1.118×10^0	1×10^{-1}
4	3.528×10^2	4.969×10^{-2}	1.408×10^{-4}	6.347×10^0	1.118×10^1	1×10^1
5	3.528×10^2	4.969×10^0	1.408×10^{-2}	6.347×10^0	1.118×10^{-1}	1×10^{-1}

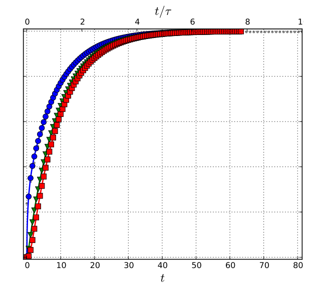
Table A.1: $\hat{h} = 5.559\ [W\ K^{-1}\ m^{-2}]$



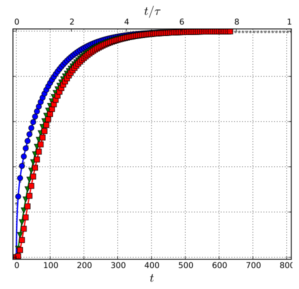
(5) $\rho_s, 10\kappa_s$



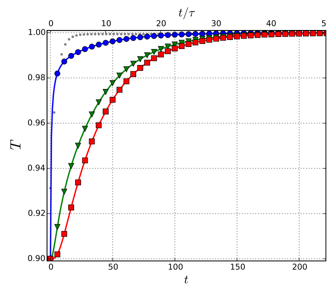
(2) $\rho_s/10,\ \kappa_s$



(1) $\rho_s,\ \kappa_s$



(3) $10\rho_s,\ \kappa_s$



(4) $\rho_s, \kappa_s/10$

Figure A.2: Three point inside the solid $\{-0.10, -0.05, 0.0\}$, squares, triangles, and circles respectively.

Each plot in Figure A.2. shows the temporal evolution for three points inside the solid, $y/\Delta = \{-0.10, -0.05, 0.0\}$. From left to right, the density increases, while from bottom to top the conductivity increases. As ρ_s increases, α_s decreases, Bi is constant, and the shape of the three temperature curves is qualitatively similar. Furthermore, the effect of modifying ρ_s can be seen in the simulation time t. The rise of ρ_s increases the thermal time τ, resulting in the increase of the simulation time necessary to achieve the steady state. The behaviour of the curves in cases 4 and 5 is completely different. In case 4, the reduction of conductivity leads to the increase of Bi. It results in a change in the shape of the curves, along with the increase of the simulation time. This occurs despite of the fact that τ is maintained fixed in both cases. In case 5, the lumped analysis, as described in Section 5.1.2, is completely suitable since $Bi \sim 0.1$. The temperature remains *uniform* within the

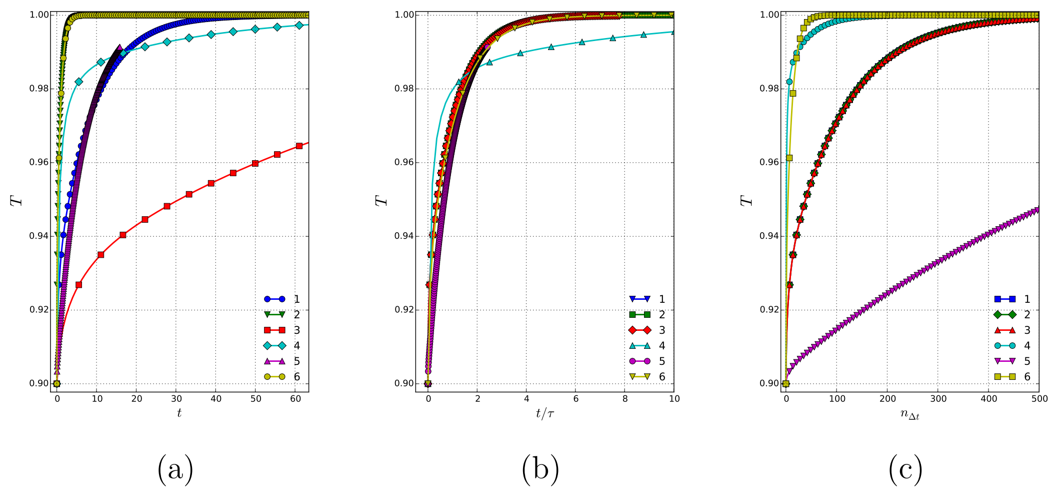


Figure A.3: Temperature T at $x/L_0 = 0.5$ and $y/\Delta = 0.0$ as function of (a) time t, (b) normalised time t/τ, and (c) number of time steps $n_{\Delta t}$.

body at all times and changes only with time. It is worth noting that, for the cases with the same Biot number, the steady state seems to be achieved around the same value of the ratio t/τ, despite of the differences in the time scale t. For cases 2,1 and 3, the steady state is reached approximately when the simulation times are 5, 50 and 500, respectively. These times correspond to $t/\tau \sim 6$ for all cases, see Figure A.3. Regarding the thermal activity ratio K for these three cases, a *reduction* of τ is associated to an *increase* of K.

A comparison of the curves corresponding to $y/\Delta = 0$ for the five cases discussed above is shown in Figure A.3(a). An additional case is also included (curve 6). It corresponds to increase the time step size of the case 2. It is done to perform a faster simulation. The equivalence between the curves with equal Biot number is shown in Figure A.3(b). Here, the thermal time τ corresponding to each case is used to plot each temperature curve as function of its own ratio t/τ. Cases 1 to 3 are completely overlapped. The difference with curves 4 and 5 can be clearly observed. Figure A.3(c) shows the temperature curves as function of the number of time steps $n_{\Delta t}$ used to perform each simulation. Once again, cases 1 to 3 overlap. It means that, given a Biot number, the same temperature can be achieved by using the same number of time steps, even though the simulation times are different. On the other hand, for the same number of time steps, as the Biot number increases the temperature achieved decreases. From this analysis, it is possible to conclude that, given a solid with prescribed thermal properties, numerical simulations with the same Biot number can be considered physically equivalent. Finally, these results are applied to accelerate the thermal coupling in Section 5.4.1, and can be considered as a numerical strategy to deal with the scale disparity in conjugate heat transfer problems.

Bibliography

[1] Jack Dongarra, Jeffrey Hittinger, John Bell, Luis Chacon, Robert Falgout, Michael Heroux, Paul Hovland, Esmond Ng, Clayton Webster, and Stefan Wild. Applied mathematics research for exascale computing. Technical report, Lawrence Livermore National Laboratory (LLNL), Livermore, CA, 2014.

[2] Anthony P Craig, Mariana Vertenstein, and Robert Jacob. A new flexible coupler for earth system modeling developed for ccsm4 and cesm1. *The International Journal of High Performance Computing Applications*, 26(1):31–42, 2012.

[3] DAVID S Blom, VERENA Krupp, ALEXANDER H Van Zuijlen, HARALD Klimach, SABINE Roller, and HESTER Bijl. On parallel scalability aspects of strongly coupled partitioned fluid-structure-acoustics interaction. In *Coupled Problems 2015: Proceedings of the 6th International Conference on Computational Methods for Coupled Problems in Science and Engineering, Venice, Italy, 18-20 May 2015*. CIMNE, 2015.

[4] S Gövert, D Mira, M Zavala-Ake, JBW Kok, M Vázquez, and G Houzeaux. Heat loss prediction of a confined premixed jet flame using a conjugate heat transfer approach. *International Journal of Heat and Mass Transfer*, 2016.

[5] S Jaure, F Duchaine, G Staffelbach, and LYM Gicquel. Massively parallel conjugate heat transfer methods relying on large eddy simulation applied to an aeronautical combustor. *Computational Science & Discovery*, 6(1):015008, 2013.

[6] David E Keyes, Lois C McInnes, Carol Woodward, William Gropp, Eric Myra, Michael Pernice, John Bell, Jed Brown, Alain Clo, Jeffrey Connors, et al. Multiphysics simulations: Challenges and opportunities. *The International Journal of High Performance Computing Applications*, 27(1):4–83, 2013.

[7] Jay Larson, Robert Jacob, and Everest Ong. The model coupling toolkit: a new fortran90 toolkit for building multiphysics parallel coupled models. *The International Journal of High Performance Computing Applications*, 19(3):277–292, 2005.

[8] Juan Alonso, Seonghyeon Hahn, Frank Ham, Marcus Herrmann, Gianluca Iaccarino, Georgi Kalitzin, Patrick LeGresley, Ken Mattsson, Gorazd Medic, Parviz Moin, et al. Chimps: A high-performance scalable module for multi-physics simulations. In *42nd AIAA/ASME/SAE/ASEE Joint Propulsion Conference & Exhibit*, page 5274, 2006.

[9] Wolfgang Joppich and M Kürschner. Mpcci—a tool for the simulation of coupled applications. *Concurrency and computation: Practice and Experience*, 18(2):183–192, 2006.

[10] Samuel Buis, Andrea Piacentini, and Damien Déclat. Palm: a computational framework for assembling high-performance computing applications. *Concurrency and Computation: Practice and experience*, 18(2):231–245, 2006.

[11] Andrea Piacentini, THIERRY Morel, ANTHONY Thévenin, and FLORENT Duchaine. O-palm: An open source dynamic parallel coupler. In *Proceedings of the IV International Conference on Computational Methods for Coupled Problems in Science and Engineering–Coupled Problems*, 2011.

[12] Roger Pawlowski, Roscoe Bartlett, Noel Belcourt, Russell Hooper, and Rod Schmidt. A theory manual for multiphysics code coupling in lime version 1.0. *Sandia National Laboratories, SAND2011-2195*, 2011.

[13] SR Slattery, PPH Wilson, and RP Pawlowski. The data transfer kit: a geometric rendezvous-based tool for multiphysics data transfer. In *International conference on mathematics & computational methods applied to nuclear science & engineering (M&C 2013)*, pages 5–9, 2013.

[14] Adnan Ibrahimbegovic, Rainer Niekamp, Christophe Kassiotis, Damijan Markovic, and Hermann G Matthies. Code-coupling strategy for efficient development of computer software in multiscale and multiphysics nonlinear evolution problems in computational mechanics. *Advances in Engineering Software*, 72:8–17, 2014.

[15] A Refloch, B Courbet, A Murrone, P Villedieu, C Laurent, P Gilbank, J Troyes, L Tessé, G Chaineray, JB Dargaud, et al. Cedre software. *Aerospace-Lab*, (2):p–1, 2011.

[16] Florent Duchaine, Stéphan Jauré, Damien Poitou, Eric Quémerais, Gabriel Staffelbach, Thierry Morel, and Laurent Gicquel. Analysis of high performance conjugate heat transfer with the openpalm coupler. *Computational Science & Discovery*, 8(1):015003, 2015.

[17] Hans-Joachim Bungartz, Florian Lindner, Bernhard Gatzhammer, Miriam Mehl, Klaudius Scheufele, Alexander Shukaev, and Benjamin Uekermann.

precice–a fully parallel library for multi-physics surface coupling. *Computers & Fluids*, 141:250–258, 2016.

[18] Yu-Hang Tang, Shuhei Kudo, Xin Bian, Zhen Li, and George Em Karniadakis. Multiscale universal interface: a concurrent framework for coupling heterogeneous solvers. *Journal of Computational Physics*, 297:13–31, 2015.

[19] Miriam Mehl, Benjamin Uekermann, Hester Bijl, David Blom, Bernhard Gatzhammer, and Alexander Van Zuijlen. Parallel coupling numerics for partitioned fluid–structure interaction simulations. *Computers & Mathematics with Applications*, 71(4):869–891, 2016.

[20] SM Longshaw, A Skillen, C Moulinec, and DR Emerson. Code coupling at scale: Towards the digital product.

[21] Prace-2ip project. `http://www.prace-ri.eu/prace-2ip/`. Accessed: 2018-April-5.

[22] Copa-gt project. `http://copagt.cerfacs.fr/description.html`. Accessed: 2018-April-5.

[23] Sherloc project. `http://sherloc-project.com/`. Accessed: 2018-April-5.

[24] Mariano Vázquez, Guillaume Houzeaux, Seid Koric, Antoni Artigues, Jazmin Aguado-Sierra, Ruth Arís, Daniel Mira, Hadrien Calmet, Fernando Cucchietti, Herbert Owen, et al. Alya: Multiphysics engineering simulation toward exascale. *Journal of Computational Science*, 14:15–27, 2016.

[25] Frédéric Archambeau, Namane Méchitoua, and Marc Sakiz. Code saturne: A finite volume code for the computation of turbulent incompressible flows-industrial applications. *International Journal on Finite Volumes*, 1(1):http–www, 2004.

[26] I Rupp and Ch Peniguel. Coupling heat conduction, radiation and convection in complex geometries. *International Journal of Numerical Methods for Heat & Fluid Flow*, 9(3):240–264, 1999.

[27] D. Mira M. Vázquez G. Houzeaux J. M. Zavala-Aké, M. Rivero. Towards an hpc-based coupling tool for eulerian-lagrangian simulations. 12th International Smoothed Particle Hydrodynamics European Research Interest Community (SPHERIC) Workshop, 2017.

[28] Daniel Mira, M Zavala-Ake, Matías Avila, Herbert Owen, Juan C Cajas, Mariano Vazquez, and Guillaume Houzeaux. Heat transfer effects on a fully premixed methane impinging flame. *Flow, Turbulence and Combustion*, 97(1):339–361, 2016.

[29] S Gövert, Daniel Mira, M Zavala-Ake, JBW Kok, Mariano Vázquez, and Guillaume Houzeaux. Heat loss prediction of a confined premixed jet flame using a conjugate heat transfer approach. *International Journal of Heat and Mass Transfer*, 107:882–894, 2017.

[30] Miguel Zavala-Aké, Daniel Mira, Mariano Vázquez, and Guillaume Houzeaux. A partitioned methodology for conjugate heat transfer on dynamic structures. In *Jülich Aachen Research Alliance (JARA) High-Performance Computing Symposium*, pages 37–47. Springer, 2016.

[31] Matias Ignacio Rivero. *A parallel algorithm for deformable contact problems.* PhD thesis, Universitat Politécnica de Catalunya, 2018.

[32] Andrea Toselli and Olof B Widlund. *Domain decomposition methods: algorithms and theory*, volume 34. Springer, 2005.

[33] Barry Smith, Petter Bjorstad, and William Gropp. *Domain decomposition: parallel multilevel methods for elliptic partial differential equations.* Cambridge university press, 2004.

[34] Hermann Amandus Schwarz. *Ueber einen Grenzübergang durch alternirendes Verfahren.* Zürcher u. Furrer, 1870.

[35] Dalibor Jajcevic, Eva Siegmann, Charles Radeke, and Johannes G Khinast. Large-scale cfd–dem simulations of fluidized granular systems. *Chemical Engineering Science*, 98:298–310, 2013.

[36] Hans-Joachim Bungartz, Florian Lindner, Bernhard Gatzhammer, Miriam Mehl, Klaudius Scheufele, Alexander Shukaev, and Benjamin Uekermann. Precice–a fully parallel library for multi-physics surface coupling. *Computers & Fluids*, 141:250–258, 2016.

[37] Tarek Mathew. *Domain decomposition methods for the numerical solution of partial differential equations*, volume 61. Springer Science & Business Media, 2008.

[38] F Gastaldi, A Quarteroni, and G Sacchi Landriani. Coupling of two-dimensional hyperbolic and elliptic equations. *Computer Methods in Applied Mechanics and Engineering*, 80(1-3):347–354, 1990.

[39] Alfio Quarteroni. *Numerical models for differential problems*, volume 2. Springer Science & Business Media, 2010.

[40] Alfio Quarteroni and Alberto Valli. *Domain decomposition methods for partial differential equations.* Number CMCS-BOOK-2009-019. Oxford University Press, 1999.

[41] David Halliday, Robert Resnick, and Jearl Walker. *Fundamentals of Physics, Chapters 33-37*. John Wiley & Sons, 2010.

[42] Behrooz Parhami. *Introduction to parallel processing: algorithms and architectures*. Springer Science & Business Media, 2006.

[43] I Rupp and C Peniguel. Syrthes 4.0 user manual, 2012.

[44] George Karypis and Vipin Kumar. A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. 1998.

[45] Torsten Langer, Alexander Belyaev, and Hans-Peter Seidel. Spherical barycentric coordinates. In *Symposium on Geometry Processing*, pages 81–88, 2006.

[46] Laksono Adhianto, Sinchan Banerjee, Mike Fagan, Mark Krentel, Gabriel Marin, John Mellor-Crummey, and Nathan R Tallent. Hpctoolkit: Tools for performance analysis of optimized parallel programs. *Concurrency and Computation: Practice and Experience*, 22(6):685–701, 2010.

[47] Christian Feichtinger, Johannes Habich, Harald Köstler, Georg Hager, Ulrich Rüde, and Gerhard Wellein. A flexible patch-based lattice boltzmann parallelization approach for heterogeneous gpu–cpu clusters. *Parallel Computing*, 37(9):536–549, 2011.

[48] Pedro Valero-Lara, Francisco D Igual, Manuel Prieto-Matías, Alfredo Pinelli, and Julien Favier. Accelerating fluid–solid simulations (lattice-boltzmann & immersed-boundary) on heterogeneous architectures. *Journal of Computational Science*, 10:249–261, 2015.

[49] Dalibor Jajcevic, Eva Siegmann, Charles Radeke, and Johannes G Khinast. Large-scale cfd–dem simulations of fluidized granular systems. *Chemical Engineering Science*, 98:298–310, 2013.

[50] Massimo Bernaschi, Massimiliano Fatica, Simone Melchionna, Sauro Succi, and Efthimios Kaxiras. A flexible high-performance lattice boltzmann gpu code for the simulations of fluid flows in complex geometries. *Concurrency and Computation: Practice and Experience*, 22(1):1–14, 2010.

[51] Scott Ellis. http://www.cirrascale.com/blog/index.php/exploring-the-pcie-bus-routes/, april 28, 2017.

[52] Davide Rossetti. https://devblogs.nvidia.com/parallelforall/benchmarking-gpudirect-rdma-on-modern-server-platforms/, april 28, 2017.

[53] Mark Harris. https://devblogs.nvidia.com/parallelforall/benchmarking-cuda-aware-mpi/, april 28, 2017.

[54] John Cheng, Max Grossman, and Ty McKercher. *Professional Cuda C Programming*. John Wiley & Sons, 2014.

[55] Florent Duchaine, Sandrine Berger, Gabriel Staffelbach, and Laurent Gicquel. Partitioned high performance code coupling applied to cfd, first jara high-performance computing symposium. 10164, 2017.

[56] Hans-Georg Matuttis and Jian Chen. *Understanding the discrete element method: simulation of non-spherical particles for granular and multi-body systems*. John Wiley & Sons, 2014.

[57] G Lu, JR Third, and CR Müller. Discrete element models for non-spherical particle systems: From theoretical developments to applications. *Chemical Engineering Science*, 127:425–465, 2015.

[58] Alexander Konyukhov and Karl Schweizerhof. *Computational Contact Mechanics: geometrically exact theory for arbitrary shaped bodies*, volume 67. Springer Science & Business Media, 2012.

[59] Zdeněk Dostál, Tomáš Kozubek, Marie Sadowská, and Vít Vonk. *Scalable Algorithms for Contact Problems*. Springer, 2017.

[60] David Kinderlehrer and Guido Stampacchia. *An introduction to variational inequalities and their applications*. SIAM, 2000.

[61] Peter Wriggers. *Computational contact mechanics*. Springer Science & Business Media, 2006.

[62] Kenneth Langstreth Johnson and Kenneth Langstreth Johnson. *Contact mechanics*. Cambridge university press, 1987.

[63] Vladislav A Yastrebov. *Numerical methods in contact mechanics*. John Wiley & Sons, 2013.

[64] Anthony C Fischer-Cripps. *Introduction to contact mechanics*. Springer, 2000.

[65] Olek C Zienkiewicz and Robert L Taylor. *The finite element method for solid and structural mechanics*. Butterworth-heinemann, 2005.

[66] Allan F Bower. *Applied mechanics of solids*. CRC press, 2009.

[67] Rolf H Krause and Barbara I Wohlmuth. A dirichlet–neumann type algorithm for contact problems with friction. *Computing and visualization in science*, 5(3):139–148, 2002.

[68] E Casoni, Antoine Jérusalem, Cristóbal Samaniego, Beatriz Eguzkitza, Pierre Lafortune, DD Tjahjanto, Xavier Sáez, Guillaume Houzeaux, and Mariano Vázquez. Alya: computational solid mechanics for supercomputers. *Archives of Computational Methods in Engineering*, 22(4):557–576, 2015.

[69] Valentin L Popov. *Contact mechanics and friction.* Springer, 2010.

[70] ASTM D7136/D7136M-12. Standard test method for measuring the damage resistance of a fiber-reinforced polymer matrix composite to a drop-weight impact event, 2015.

[71] Natthawat Hongkarnjanakul. *Modélisation numérique pour la tolérance aux dommages d'impact sur stratifié composite: de l'impact à la résistance résiduelle en compression.* PhD thesis, Toulouse, ISAE, 2013.

[72] MR Abir, TE Tay, M Ridha, and HP Lee. Modelling damage growth in composites subjected to impact and compression after impact. *Composite Structures*, 168:13–25, 2017.

[73] Ahmet S Yigit and Andreas P Christoforou. Limits of asymptotic solutions in low-velocity impact of composite plates. *Composite Structures*, 81(4):568–574, 2007.

[74] Michael Smith. *ABAQUS/Standard User's Manual, Version 6.9.* Simulia, 2009.

[75] Hans-Joachim Bungartz, Florian Lindner, Miriam Mehl, Klaudius Scheufele, Alexander Shukaev, and Benjamin Uekermann. Partitioned fluid–structure–acoustics interaction on distributed data: Coupling via precice. In *Software for Exascale Computing-SPPEXA 2013-2015*, pages 239–266. Springer, 2016.

[76] Florent Duchaine, Sandrine Berger, Gabriel Staffelbach, and Laurent Gicquel. Partitioned high performance code coupling applied to cfd.

[77] Florent Duchaine, Stéphan Jauré, Damien Poitou, Eric Quémerais, Gabriel Staffelbach, Thierry Morel, and Laurent Gicquel. Analysis of high performance conjugate heat transfer with the openpalm coupler. *Computational Science & Discovery*, 8(1):015003, 2015.

[78] Hans-Joachim Bungartz, Florian Lindner, Miriam Mehl, Klaudius Scheufele, Alexander Shukaev, and Benjamin Uekermann. Partitioned fluid–structure–acoustics interaction on distributed data: Coupling via precice. In *Software for Exascale Computing-SPPEXA 2013-2015*, pages 239–266. Springer, 2016.

[79] Adrian Bejan. *Convection heat transfer.* John wiley & sons, 2013.

[80] William S Janna. *Engineering heat transfer*. CRC Press, 1999.

[81] A.S. Dorfman. *Conjugate Problems in Convective Heat Transfer*. Heat Transfer. CRC Press, 2009.

[82] G Houzeaux and J Principe. A variational subgrid scale model for transient incompressible flows. *International Journal of Computational Fluid Dynamics*, 22(3):135–152, 2008.

[83] G Houzeaux, R Aubry, and M Vázquez. Extension of fractional step techniques for incompressible flows: The preconditioned orthomin (1) for the pressure schur complement. *Computers & Fluids*, 44(1):297–313, 2011.

[84] TL Perelman. On conjugated problems of heat transfer. *International Journal of Heat and Mass Transfer*, 3(4):293–303, 1961.

[85] G Gimenez, M Errera, D Baillis, Y Smith, and F Pardo. A coupling numerical methodology for weakly transient conjugate heat transfer problems. *International Journal of Heat and Mass Transfer*, 97:975–989, 2016.

[86] Tom Verstraete and Sebastian Scholl. Stability analysis of partitioned methods for predicting conjugate heat transfer. *International Journal of Heat and Mass Transfer*, 101:852–869, 2016.

[87] Sebastian Scholl, Tom Verstraete, Florent Duchaine, and Laurent Gicquel. Conjugate heat transfer of a rib-roughened internal turbine blade cooling channel using large eddy simulation. *International Journal of Heat and Fluid Flow*, 61:650–664, 2016.

[88] Cédric Flageul, Sofiane Benhamadouche, Eric Lamballais, and Dominique Laurence. Dns of turbulent channel flow with conjugate heat transfer: Effect of thermal boundary conditions on the second moments and budgets. *International Journal of Heat and Fluid Flow*, 55:34–44, 2015.

[89] AV Luikov. Conjugate convective heat transfer problems. *International Journal of Heat and Mass Transfer*, 17(2):257–265, 1974.

[90] Philipp Birken, Karsten J Quint, Stefan Hartmann, and Andreas Meister. A time-adaptive fluid-structure interaction method for thermal coupling. *Computing and visualization in science*, 13(7):331–340, 2010.

[91] David J Tritton. *Physical fluid dynamics*. Springer Science & Business Media, 2012.

[92] Hermann Schlichting, Klaus Gersten, Egon Krause, Herbert Oertel, and Katherine Mayes. *Boundary-layer theory*, volume 7. Springer, 1955.

[93] Louis Rosenhead. *Laminar boundary layers: an account of the development, structure, and stability of laminar boundary layers in incompressible fluids, together with a description of the associated experimental techniques*. Clarendon Press, 1963.

[94] Martin Busch and Bernhard Schweizer. Coupled simulation of multibody and finite element systems: an efficient and robust semi-implicit coupling approach. *Archive of Applied Mechanics*, 82(6):723–741, 2012.

[95] Kenneth H Huebner, Donald L Dewhirst, Doughlas E Smith, and Ted G Byrom. *The finite element method for engineers*. John Wiley & Sons, 2008.

[96] Antony Misdariis, Olivier Vermorel, and Thierry Poinsot. Les of knocking in engines using dual heat transfer and two-step reduced schemes. *Combustion and Flame*, 162(11):4304–4312, 2015.

[97] Ananth Grama. *Introduction to parallel computing*. Pearson Education, 2003.

[98] Devang Patel and Lawrence Rauchwerger. Principles of speculative run—time parallelization. In *International Workshop on Languages and Compilers for Parallel Computing*, pages 323–337. Springer, 1998.

[99] Maria Predari. *Load balancing for parallel coupled simulations*. PhD thesis, Bordeaux, 2016.

[100] Hans-Joachim Bungartz, Florian Lindner, Miriam Mehl, Klaudius Scheufele, Alexander Shukaev, and Benjamin Uekermann. Partitioned fluid–structure–acoustics interaction on distributed data: Coupling via precice. In *Software for Exascale Computing-SPPEXA 2013-2015*, pages 239–266. Springer, 2016.

[101] Nan Ding, Wei Xue, Zhenya Song, Haohuan Fu, Shiming Xu, and Weimin Zheng. An automatic performance model-based scheduling tool for coupled climate system models. *Journal of Parallel and Distributed Computing*, 2018.

[102] Yu-Liang Wu, Wenqi Huang, Siu-chung Lau, CK Wong, and Gilbert H Young. An effective quasi-human based heuristic for solving the rectangle packing problem. *European Journal of Operational Research*, 141(2):341–358, 2002.

[103] Yuefan Deng. *Applied parallel computing*. World Scientific, 2013.

[104] Charles Martel and Ramesh Subramonian. On the complexity of certified write-all algorithms. *Journal of Algorithms*, 16(3):361–387, 1994.

[105] F Thomson Leighton. *Introduction to parallel algorithms and architectures: Arrays· trees· hypercubes*. Elsevier, 2014.

[106] Clyde P Kruskal, Larry Rudolph, and Marc Snir. A complexity theory of efficient parallel algorithms. *Theoretical Computer Science*, 71(1):95–132, 1990.

[107] Ian Foster. Designing and building parallel programs: Concepts and tools for parallel software engineering. 1995.

[108] Laurent Colombet and Laurent Desbat. Speedup and efficiency of large size applications on heterogeneous networks. In *European Conference on Parallel Processing*, pages 651–664. Springer, 1996.

[109] Daihee Kim, J Walter Larson, and Kenneth Chiu. Automatic performance prediction for load-balancing coupled models. In *Cluster, Cloud and Grid Computing (CCGrid), 2013 13th IEEE/ACM International Symposium on*, pages 410–417. IEEE, 2013.

[110] Dalibor Jajcevic, Eva Siegmann, Charles Radeke, and Johannes G Khinast. Large-scale cfd–dem simulations of fluidized granular systems. *Chemical Engineering Science*, 98:298–310, 2013.

[111] Stuart R Slattery, Steven P Hamilton, and Thomas M Evans. A modified moving least square algorithm for solution transfer on a spacer grid surface. In *ANS MC2015—Joint International Conference on Mathematics and Computation (M&C), Supercomputing in Nuclear Applications (SNA) and the Monte Carlo (MC) Method, Nashville, TN, American Nuclear Society, LaGrange Park*, 2015.

[112] Jichun Li and YC Hon. Domain decomposition for radial basis meshless methods. *Numerical Methods for Partial Differential Equations*, 20(3):450–462, 2004.

[113] Alexandre Chatelain, Frédéric Ducros, and Olivier Métais. Large eddy simulation of conjugate heat-transfer using thermal wall-functions. In *Direct and Large-Eddy Simulation V*, pages 307–314. Springer, 2004.