Efficient Parallel Strategies For Conjugate Heat Transfer Problems

Guillaume Houzeaux
guillaume.houzeaux@bsc.es
Barcelona Supercomputing Center
Barcelona, Spain

Simon Santoso
simon.santoso@bsc.es
Barcelona Supercomputing Center
Barcelona, Spain

Marta Garcia-Gasulla
marta.garcia@bsc.es
Barcelona Supercomputing Center
Barcelona, Spain

Cristóbal Samaniego
cristobal.samaniego@bsc.es
Barcelona Supercomputing Center
Barcelona, Spain

Hadrien Calmet
hadrien.calmet@bsc.es
Barcelona Supercomputing Center
Barcelona, Spain

ABSTRACT

Historically, temperature boundary conditions in thermal fluids have conventionally been approached as Robin-type boundary conditions. However, with the emergence of supercomputing capabilities, there is now the opportunity to explore the solution of heat transfer in the surrounding domains and establish a strong coupling with the temperature equation in the fluid, giving rise to what is known as Conjugate Heat Transfer problems. This paper introduces two coupling strategies based on volume and surface algebraic couplings, solved using either a block Gauss-Seidel method or a block Jacobi method. The volume coupling implies solving the heat transfer problem in the fluid and solid monolithically and coupling it to the Navier-Stokes equations solved in the fluid. On the other hand, in the case of surface coupling, the Boussinesq system is solved within the fluid and then coupled to the solid through their shared interface. A comparative analysis of these approaches is presented, considering both algorithmic and computational performances within the framework of a multi-code coupling strategy. In the parallel execution of such problems, a crucial decision involves determining how to distribute the cores among the various coupled codes. We propose a method that involves overloading computational nodes, allowing different codes to utilize the entire available resources. To enhance efficiency, the overload approach is implemented with a barrier, utilizing the DLB library, to mitigate the busy wait induced by MPI subroutines during data exchange. The solution to a practical example demonstrates a nearly twofold speedup achieved by the proposed method compared to a classical approach when employing volume coupling.

CCS CONCEPTS

• **Computing methodologies** → **Massively parallel algorithms**; **Massively parallel and high-performance simulations**; **Distributed programming languages**; • **Applied computing** → **Physics**.

KEYWORDS

Conjugate heat transfer, coupling, high-performance computing, resources allocation, MPI, algorithmic performance, computational performance

ACM Reference Format:
Guillaume Houzeaux, Simon Santoso, Marta Garcia-Gasulla, Cristóbal Samaniego, and Hadrien Calmet. 2024. Efficient Parallel Strategies For Conjugate Heat Transfer Problems. In *Platform for Advanced Scientific Computing Conference (PASC '24), June 3–5, 2024, Zurich, Switzerland.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3659914.3659930

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
PASC '24, June 3–5, 2024, Zurich, Switzerland
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0639-4/24/06
https://doi.org/10.1145/3659914.3659930

1 INTRODUCTION

The phenomenon of Conjugate Heat Transfer plays a crucial role in numerous modern scenarios. One of the initial formulations of this problem was undertaken by Perelman in 1961 [19], where the author explored the boundary problem of heat transfer between a solid body, acting as a heat source, and a surrounding fluid using the asymptotic solution method for integral equations. Subsequently, a multitude of analytical and industrial studies have been conducted to understand and control this phenomenon.

In the context of analytical works, Siegel in 1963 [24] delved into the analysis of laminar convection heat transfer between two parallel plates. Theoretical predictions regarding wall temperature, heated by a step or an exponential heat source, were made in subsequent works such as [25], [26], and [27]. More recent industrial studies have also been carried out. Perelman, in 1961 [19], extended the method to more complex geometries in 1970 [14], specifically in the context of compressible flows. Various articles are dedicated to studying heat transfer in the context of turbo-machinery, as seen in [23] and [4].

Moreover, this phenomenon has been the subject of numerous numerical works. For instance, Codina et al. [3] explored the numerical approximations of heat transfer for thin walls using the Finite Element Method. The finite volume method is employed in [18] to conduct a numerical study of conjugate heat transfer in micro-channels. Other works include [15], where the study focuses on centrifuge machines used in the process industry, and [5], where a predictive model that considers radiation is developed for heat exchange between gray walls in a transparent medium.

The treatment of this kind of problems has been classically done with a monolithic approach [3][18][15] but in the latter decades, the development of partitioned methods coming from the Domain Decomposition methods (DDM) [20][28] made this treatment more suitable for industrial applications. Indeed, since most common industrial applications are of multi-physics nature, it is natural to treat each physical problems separately, while taking in consideration the influence of one physics on another, and finally build a final global solution by reassembling each part of the solution. Consequently, the method consists in dividing the global domain in different subdomains which deals with one or physical problem, solves it with the use of any classical numerical scheme, and finally reassemble the global solution, by ensuring the continuities of the variables and their fluxes of across the interface between subdomains. The use of partitioned methods to deal with coupling problems has been the topic of [17] for the investigation of aeroplasticity and aerothermoelasticity phenomenons for aerospace applications. In [11], the author investigates a surface coupling method for the study of the cardiovascular system.

The partitioned methods utilized to address such problems primarily differ in their approach to coupling, whether utilizing volume or surface coupling. Volume coupling involves decomposing physical problems in terms of governing equations, with each problem solved within its respective subdomain (e.g. Incompressible Navier-Stokes and heat transfer equation). Since these subdomains may overlap, the transmission of information in this case involves primary variables (e.g., velocity and temperature). In contrast, surface coupling is grounded in subdomain decomposition [11]. The entire coupling problem is addressed within each subdomain, and information is transmitted through the subdomains interface, consisting basically in imposing continuity of variables and their fluxes. Consequently, the quantity of information to transmit is inherently lower than in volume coupling.

This coupling can be treated with a block Jacobi method or with a block Gauss-Seidel method. Although it is known that Gauss-Seidel methods have better convergence and robustness properties than Jacobi, problems may appear concerning the computational efficiency. Indeed, the Gauss-Seidel method is by nature sequential. Therefore, in the context of high-performance computing, the amount of CPU's idle time may be as much as 50% of the time, which is not acceptable. In [11], the authors performs a large study on Gauss-Seidel parallel efficiency and investigates a way to enhance Gauss-Seidel performance by the use of an overloading strategy combined with a barrier [6].

In this paper, our objective is to compare various strategies, based on both volume and surface couplings, and solved using the block Jacobi and block Gauss-Seidel methods. We aim to elucidate the advantages and drawbacks of each method, considering both algorithmic and computational performance perspectives. To capitalize on the intelligent algorithmic properties of volume coupling when solved using a block Gauss-Seidel method in a multi-code context, we introduce an overload strategy to allow different codes to utilize all available resources. However, this strategy faces inefficiency due to the busy wait imposed by the MPI library during the coupling data exchange. For effectiveness, the overload strategy requires a mechanism to reduce the consumption of CPU cycles during this busy wait. To address this, we rely on the DLB library, which provides a barrier between different MPI processes running on the same node.

The structure of this article is as follows: Section 2 provides an overview of the governing equations at the continuous level, encompassing both volume and surface couplings. The subsequent section delves into these couplings at the algebraic level, illustrating their equivalence to the monolithic problem. Section 4 focuses on presenting the block Gauss-Seidel and block Jacobi methods employed for solving the coupled problems in a multi-code context, along with their comparative analysis. Following this, Section 5 outlines the overloading strategy utilized to enhance the computational performance of the Gauss-Seidel method. Finally, in Section 6, a practical example involving the ITER Tokamak is solved to demonstrate the efficacy of the proposed methodology.

2 COUPLING METHODS AT THE CONTINUOUS LEVEL

This section starts by describing the sets of governing equations of generic conjugate heat transfer problems. Regarding the fluid, the Boussinesq approximation is considered although alternative approximations such as Low-Mach or iso-thermal flows can be envisaged as well. We then introduce two coupling strategies, at the continuous level. Next section will deal with the practical implementation carried out at the algebraic level.

2.1 Problem setting

We consider two subdomains which will be denoted as Ω_f and Ω_s where f stands for fluid and s stands for solid, as depicted in Figure 1. We denote as Γ the interface between Ω_f and Ω_s,

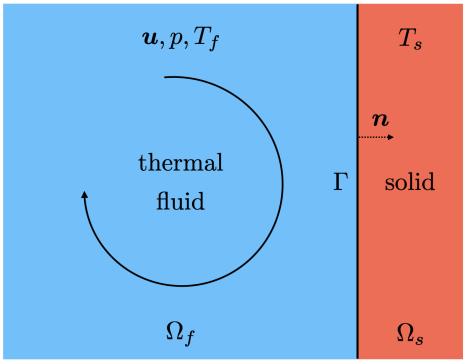

Figure 1: Conjugate heat transfer between a fluid in Ω_f and a solid in Ω_s.

with exterior normal to Ω_f denoted $\boldsymbol{n}$. In the fluid domain Ω_f we consider the incompressible Navier-Stokes equations under the Boussinesq approximation, referred to as the Boussinesq system, and solve for the velocity $\boldsymbol{u}$, pressure p and temperature T_f such that:

$$\begin{aligned} \rho_f \frac{\partial \boldsymbol{u}}{\partial t} + \rho_f \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - \mu \Delta \boldsymbol{u} &= \rho_f \boldsymbol{g}(1 - \beta(T_f - T_0)) \\ \nabla \cdot \boldsymbol{u} &= 0, \\ \rho_f C_{p,f} \frac{\partial T_f}{\partial t} + \rho_f C_{p,f} \boldsymbol{u} \cdot \nabla T_f - k_f \Delta T_f &= 0. \end{aligned} \tag{1}$$

In the solid subdomain Ω_s, we consider only the diffusion equation of the temperature T_s:

$$\rho_s C_{p,s} \frac{\partial T_s}{\partial t} - k_s \Delta T_s = Q_s, \tag{2}$$

where μ is the dynamic viscosity of the fluid, $\boldsymbol{g}$ the gravity, β the thermal expansion coefficient, ρ_* the density, T_0 the quiescent temperature, $C_{p,*}$ is the specific heat capacity, k_* is the diffusion coefficient, Q_s is the source term and the star $*$ can be s or f.

Both systems of equations (1) and (2) should be provided with boundary (Dirichlet, Neumann and Robin) and initial conditions, and not detailed here for the sake of clarity.

As mentioned earlier, the fluid and solid equations are not yet coupled. We essentially have two choices. Firstly, we can opt for a surface coupling at Γ by enforcing the transmission conditions, ensuring the continuity of temperature and heat flux across their interface. Alternatively, we can consider a volume coupling approach, where both fluid and solid temperature equations are solved within a single multi-material domain in $\Omega := \overline{\Omega_f \cup \Omega_s}$. Figure 2 illustrates both strategies.

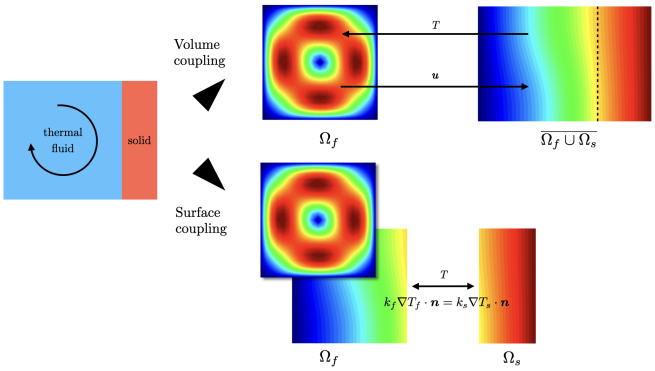

Figure 2: Volume and surface couplings for conjugate heat transfer problems.

2.1.1 Volume coupling. The volume coupling consists in solving only the Navier-Stokes equations in the fluid and a multi-material formulation of the temperature equation in both subdomains. Therefore, we solve in Ω_f

$$\begin{aligned} \rho_f \frac{\partial \boldsymbol{u}}{\partial t} + \rho_f \boldsymbol{u} \cdot \nabla \boldsymbol{u} + \nabla p - \mu \Delta \boldsymbol{u} &= \rho_f \boldsymbol{g}(1 - \beta(T_f - T_0)) \\ \nabla \cdot u &= 0 \end{aligned} \tag{3}$$

while in the complete domain Ω we consider

$$\rho C_p \frac{\partial T}{\partial t} + \rho C_p \boldsymbol{u} \cdot \nabla T - k \Delta T = Q \tag{4}$$

where

$$\begin{cases} \rho = \rho_f, & C_p = C_{p,f}, & k = k_f, & \boldsymbol{u} = \boldsymbol{u}, & Q = 0 & \text{in } \Omega_f, \\ \rho = \rho_s, & C_p = C_{p,s}, & k = k_s, & \boldsymbol{u} = \boldsymbol{0}, & Q = Q_s & \text{in } \Omega_s. \end{cases}$$

and with $T_f = T$ in Ω_f.

2.2 Surface coupling

By using Equations (1) and (2) and the following transmission conditions

$$\begin{cases} T_f &= T_s & \text{on } \Gamma, \\ k_f \nabla T_f \cdot \boldsymbol{n} &= k_s \nabla T_s \cdot \boldsymbol{n} & \text{on } \Gamma, \end{cases} \tag{5}$$

it can be shown [20] that the resulting system is strictly equivalent to system presented in Section 2.1.1. Note that combinations of Dirichlet and Neumann conditions (Robin) are also possible if they are compatible with the aforementioned continuities.

3 COUPLING METHODS AT THE ALGEBRAIC LEVEL

We now present the algebraic versions of the previous two couplings that were presented at the continuous levels. On the one hand, the volume coupling strategy follows directly from the discretization of the set of equations presented in Section 2.1.1. On the other hand,

the discretized version of the surface coupling is not obtained by discretizing directly Equations (1), (2) and (5).

The algebraic framework proposed in [9] offers several advantages. Firstly, while the current study employs a finite element method to discretize the various equations, the framework remains indifferent to the specific method being employed. Secondly, it is not reliant on the specific systems of equations in use; one can operate with algebraic vectors and residuals. Thirdly, in the context of surface coupling and when employing matching meshes at the interface, the surface coupling converges to the same solution as the monolithic coupling. This outcome would not be observed if the boundary integrals involved in the flux continuity equation 5(2) were discretized directly.

We will now present the algebraic equivalent of two couplings. The subsequent section will provide an overview of two algorithms for solving the systems, specifically the Jacobi and Gauss-Seidel methods.

3.1 Volume Coupling

As previously said, after the use of a discretization scheme such as the finite element method, one must solve a linear system to solve the discretized counterpart of equations of Sections 2.1.1. Let $\mathbf{u}$, $\mathbf{p}$, $\mathbf{t}_f$, $\mathbf{t}_\Gamma$ and $\mathbf{t}_s$ be the vectors of unknowns for the velocity, pressure, fluid temperature, interface temperature and solid temperature such that the temperature vector of unknowns is $\mathbf{t} = [\mathbf{t}_f, \mathbf{t}_\Gamma, \mathbf{t}_s]^t$. Here the interface unknowns have been specifically separated, as shown in Figure 3.

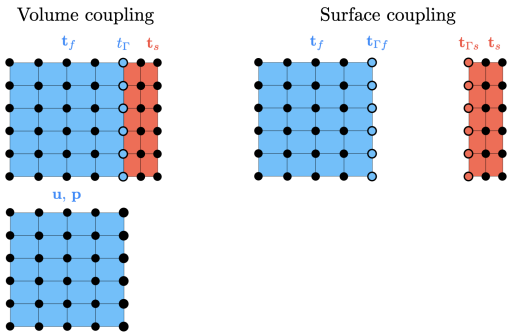

Figure 3: Unknown vectors nomenclature for volume and surface couplings.

After linearization, the discretization of the equations in Ω_f and Ω_s leads to this linear system:

$$\begin{pmatrix} \mathbf{A}_{uu} & \mathbf{A}_{up} & \mathbf{A}_{uf} & \mathbf{0} & \mathbf{A}_{u\Gamma} \\ \mathbf{A}_{pu} & \mathbf{A}_{pp} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_{fu} & \mathbf{0} & \mathbf{A}_{ff} & \mathbf{0} & \mathbf{A}_{f\Gamma} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A}_{ss} & \mathbf{A}_{s\Gamma} \\ \mathbf{A}_{\Gamma u} & \mathbf{0} & \mathbf{A}_{\Gamma f} & \mathbf{A}_{\Gamma s} & \mathbf{A}_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{p} \\ \mathbf{t}_f \\ \mathbf{t}_s \\ \mathbf{t}_\Gamma \end{pmatrix} = \begin{pmatrix} \mathbf{b}_u \\ \mathbf{b}_p \\ \mathbf{b}_f \\ \mathbf{b}_s \\ \mathbf{b}_\Gamma \end{pmatrix} \tag{6}$$

The matrices of the previous block system are obtained through the discretization of the Navier-Stokes equations using the Variational Multiscale Finite Element Method outlined in [12]. Specifically, the matrices in the first row stem from discretizing the momentum equation (3)(1), encompassing velocity, pressure, and temperature within the fluid domain. The second row involves matrices resulting from discretizing the continuity equation (3)(2), with $\mathbf{A}_{pp}$ originating solely from pressure stabilization. Moving on, the third row represents the heat transfer equation (4) within the fluid, involving velocity and temperature variables. The fourth row corresponds to the solid counterpart of the heat transfer equation, involving solely the solid domain temperature. Finally, the last row deals with the heat transfer equation at the interface.

From now on, we will assume that the temperature equation is decoupled from the Navier-Stokes equations. Such a linearization enables in practice to have two separate solvers for each system. To do so, we consider that

$$\mathbf{A}_{fu} = \mathbf{0}, \ \mathbf{A}_{\Gamma u} = \mathbf{0}, \ \text{and} \ \mathbf{A}_{ff} = \mathbf{A}_{ff}(\mathbf{u}).$$

Then, by separating the unknowns of the Navier-Stokes equations with those of the heat transfer equations, we obtain the following equivalent systems:

$$\begin{aligned} &\text{NS solver:} \quad \begin{pmatrix} \mathbf{A}_{uu} & \mathbf{A}_{up} \\ \mathbf{A}_{pu} & \mathbf{A}_{pp} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_u \\ \mathbf{b}_p \end{pmatrix} - \begin{pmatrix} \mathbf{A}_{uf}\mathbf{t}_f + \mathbf{A}_{u\Gamma}\mathbf{t}_\Gamma \\ \mathbf{0} \end{pmatrix}, \\ &\text{HT solver:} \quad \begin{pmatrix} \mathbf{A}_{ff} & \mathbf{0} & \mathbf{A}_{f\Gamma} \\ \mathbf{0} & \mathbf{A}_{ss} & \mathbf{A}_{s\Gamma} \\ \mathbf{A}_{\Gamma f} & \mathbf{A}_{\Gamma s} & \mathbf{A}_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} \mathbf{t}_f \\ \mathbf{t}_s \\ \mathbf{t}_\Gamma \end{pmatrix} = \begin{pmatrix} \mathbf{b}_f \\ \mathbf{b}_s \\ \mathbf{b}_\Gamma \end{pmatrix}. \end{aligned} \tag{7}$$

3.2 Surface coupling

In Equation (6), the submatrix $\mathbf{A}_{\Gamma_f\Gamma_f}$ of the interface Γ_f equation comes from the assembly involving the elements in both Ω_s and Ω_f. With obvious meaning, we denote as $\mathbf{A}_{\Gamma\Gamma}^{(i)}$ and $\mathbf{b}^{(i)}$ the submatrix and sub right-hand-side coming from the integration over subdomain $i = f, s$. Let us consider no advection for the sake of clarity, by letting $\mathbf{A}_{\Gamma_f u} = \mathbf{0}$. We can thus rewrite the temperature block of system (6) as:

$$\begin{pmatrix} \mathbf{A}_{ff} & \mathbf{0} & \mathbf{A}_{f\Gamma} \\ \mathbf{0} & \mathbf{A}_{ss} & \mathbf{A}_{s\Gamma} \\ \mathbf{A}_{\Gamma f} & \mathbf{A}_{\Gamma s} & \mathbf{A}_{\Gamma\Gamma}^{(f)} + \mathbf{A}_{\Gamma\Gamma}^{(s)} \end{pmatrix} \begin{pmatrix} \mathbf{t}_f \\ \mathbf{t}_s \\ \mathbf{t}_\Gamma \end{pmatrix} = \begin{pmatrix} \mathbf{b}_f \\ \mathbf{b}_s \\ \mathbf{b}_\Gamma^{(f)} + \mathbf{b}_\Gamma^{(s)} \end{pmatrix}. \tag{8}$$

Now let us duplicate the temperature unknowns on the interface, as show in Figure 3. We denote them as $\mathbf{t}_{\Gamma_f}$ and $\mathbf{t}_{\Gamma_s}$, whether they belong to Ω_f or Ω_s, respectively. Therefore, last algebraic equation is equivalent to the following system:

$$\begin{pmatrix} \mathbf{A}_{ff} & \mathbf{A}_{f\Gamma} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{ss} & \mathbf{A}_{s\Gamma} \\ \mathbf{A}_{\Gamma f} & \mathbf{A}_{\Gamma\Gamma}^{(f)} & \mathbf{A}_{\Gamma s} & \mathbf{A}_{\Gamma\Gamma}^{(s)} \end{pmatrix} \begin{pmatrix} \mathbf{t}_f \\ \mathbf{t}_{\Gamma_f} \\ \mathbf{t}_s \\ \mathbf{t}_{\Gamma_s} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_f \\ \mathbf{0} \\ \mathbf{b}_s \\ \mathbf{b}_\Gamma^{(f)} + \mathbf{b}_\Gamma^{(s)} \end{pmatrix}. \tag{9}$$

By eliminating $\mathbf{t}_{\Gamma_f}$ using the second equation and setting $\mathbf{t}_{\Gamma_s} = \mathbf{t}_\Gamma$, we can easily check that this system is equivalent to System (8). By

rearranging the terms of the last equations in (9), introducing the residual $\mathbf{r}_\Gamma^{(f)}$, and reintroducing the Navier-Stokes equations, we obtain the following two equivalent systems:

$$\begin{aligned}
\Omega_f \text{ solver: } & \begin{pmatrix} \mathbf{A}_{uu} & \mathbf{A}_{up} \\ \mathbf{A}_{pu} & \mathbf{A}_{pp} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{p} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_u \\ \mathbf{b}_p \end{pmatrix} - \begin{pmatrix} \mathbf{A}_{uf}\mathbf{t}_f + \mathbf{A}_{u\Gamma}\mathbf{t}_{\Gamma_f} \\ \mathbf{0} \end{pmatrix}, \\
& \begin{pmatrix} \mathbf{A}_{ff} & \mathbf{A}_{f\Gamma} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{t}_f \\ \mathbf{t}_{\Gamma_f} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_f \\ \mathbf{t}_{\Gamma_s} \end{pmatrix}, \\
\Omega_s \text{ solver: } & \begin{pmatrix} \mathbf{A}_{ss} & \mathbf{A}_{s\Gamma} \\ \mathbf{A}_{\Gamma s} & \mathbf{A}_{\Gamma\Gamma}^{(s)} \end{pmatrix} \begin{pmatrix} \mathbf{t}_s \\ \mathbf{t}_{\Gamma_s} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_s \\ \mathbf{b}_\Gamma^{(s)} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{r}_\Gamma^{(f)} \end{pmatrix}.
\end{aligned} \tag{10}$$

where we have defined the residual $\mathbf{r}_\Gamma^{(f)}$ computed in Ω_f on Γ as

$$\mathbf{r}_\Gamma^{(f)} = \mathbf{b}_\Gamma^{(f)} - \mathbf{A}_{\Gamma f}\mathbf{t}_f - \mathbf{A}_{\Gamma\Gamma}^{(f)}\mathbf{u}_{\Gamma_f}.$$

It should be emphasized that the residual and the flux continuity expressed in Equation (5)(2) are not fully equal, in the sense of

$$\int_\Gamma k\nabla T_f \cdot \boldsymbol{n}\, d\Gamma \approx \mathbf{r}_\Gamma^{(f)}.$$

Although the continuous flux is the right Neumann condition at the continuous level, only the algebraic formulation gives the same solution as the monolithic one (see Section 4.5.1 in [7]).

We can now clearly identify this method as a Dirichlet/Neumann method, where a Dirichlet transmission condition is applied on the solid (fourth equation in (10)) and a Neumann condition on the gas (last equation in (10)). Note that the Dirichlet and Neumann conditions have been arbitrarily applied to the fluid and solid, respectively, but can be interchanged. For this monolithic formulation, they are strictly equivalent. Lastly, it's worth noting that in this study, we assume matching meshes on the interface. The extension to non-matching meshes can be achieved following the approach outlined in [9].

4 GAUSS-SEIDEL AND JACOBI MULTI-CODE COUPLING

4.1 Multi-code coupling

In conjugate heat transfer problems, the unknowns lie in different computational domains as the velocity and pressure are only solve inside the fluid domain, while the temperature lies in the whole domain.

Following the volume coupling approach given by Equations (7), we have performed a physics partitioning (Navier-Stokes and heat transfer), while in the surface coupling approach, given by Equations (10), we have considered a domain partitioning (fluid and solid). Multi-code coupling is a practical approach for implementing multi-physics couplings of such partitioned approaches. This strategy minimizes the implementation within the code and leverages the existing parallelization strategy in the code with minimal interference (requiring only redefinitions of MPI communicators). Utilizing MPI [16] ensures a reliable exchange of information to facilitate the completion of the coupling.

In the case of volume coupling, one code (`code1.x`) handles the solution of Navier-Stokes equations in the fluid, while another code (`code2.x`) addresses the temperature solution in the fluid-solid domain. For surface coupling, the responsibility of solving the Boussinesq system in the fluid lies with one code (`code1.x`), and the other code (`code2.x`) tackles the temperature equation in the solid. Figure (4) provides a summary of these strategies, illustrating how the codes are executed using the `mpirun` function, along with the variables exchanged between them. Note that `code1.x` and `code2.x` can be two different instances coming from the same source code.

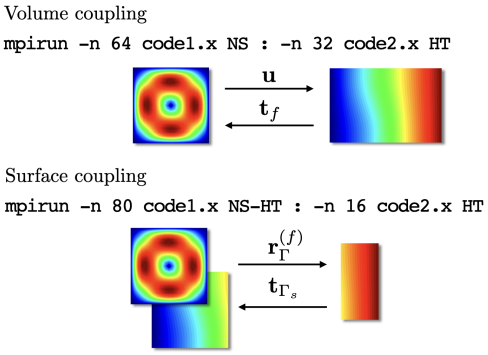

Figure 4: Multi-code coupling for volume and surface coupling.

4.2 Volume coupling

Following the multi-code approach, we introduce and iterative strategy to decouple the systems in Equation (7) so that two different codes can handle one of these. Introducing an iteration index k and a parameter $m = 0, 1$, we introduce the following iterative algorithm:

$$\begin{aligned}
\text{NS solver: } & \begin{pmatrix} \mathbf{A}_{uu} & \mathbf{A}_{up} \\ \mathbf{A}_{pu} & \mathbf{A}_{pp} \end{pmatrix} \begin{pmatrix} \mathbf{u}^{k+1} \\ \mathbf{p}^{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_u \\ \mathbf{b}_p \end{pmatrix} - \begin{pmatrix} \mathbf{A}_{uf}\mathbf{t}_f^k + \mathbf{A}_{u\Gamma}\mathbf{t}_\Gamma^k \\ \mathbf{0} \end{pmatrix}, \\
\text{HT solver: } & \begin{pmatrix} \mathbf{A}_{ff} & \mathbf{0} & \mathbf{A}_{f\Gamma} \\ \mathbf{0} & \mathbf{A}_{ss} & \mathbf{A}_{s\Gamma} \\ \mathbf{A}_{\Gamma f} & \mathbf{A}_{\Gamma s} & \mathbf{A}_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} \mathbf{t}_f^{k+1} \\ \mathbf{t}_s^{k+1} \\ \mathbf{t}_\Gamma^{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_f \\ \mathbf{b}_s \\ \mathbf{b}_\Gamma \end{pmatrix},
\end{aligned} \tag{11}$$

with $\mathbf{A}_{ff} = \mathbf{A}_{ff}(\mathbf{u}^{k+m})$. The index m can be chosen equal to 0 or 1, leading to block Jacobi or block Gauss-Seidel methods, respectively. Both will be analyzed at the end of this section. It is easy to check that upon convergence, this method converges to the same solution as Equation 6. Regarding the coupling terms in (11), the fluid equations require the temperature vector in the fluid $\mathbf{t}_f^k$ and $\mathbf{t}_\Gamma^k$ to assemble the complete system, while the temperature equation requires the vector $\mathbf{u}^{k+m}$ for the same reasons.

4.3 Surface coupling

An analogous approach can be employed for Equations (10) to decouple the solutions on Ω_s and Ω_f:

$$\Omega_f \text{ solver: } \begin{pmatrix} \mathbf{A}_{uu} & \mathbf{A}_{up} \\ \mathbf{A}_{pu} & \mathbf{A}_{pp} \end{pmatrix} \begin{pmatrix} \mathbf{u}^{k+1} \\ \mathbf{p}^{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_u \\ \mathbf{b}_p \end{pmatrix} - \begin{pmatrix} \mathbf{A}_{uf}\mathbf{t}_f^k + \mathbf{A}_{u\Gamma}\mathbf{t}_{\Gamma_f}^k \\ \mathbf{0} \end{pmatrix},$$

$$\begin{pmatrix} \mathbf{A}_{ff} & \mathbf{A}_{f\Gamma} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{t}_f^{k+1} \\ \mathbf{t}_{\Gamma_f}^{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_f \\ \mathbf{t}_{\Gamma_s}^k \end{pmatrix}.$$

$$\Omega_s \text{ solver: } \begin{pmatrix} \mathbf{A}_{ss} & \mathbf{A}_{s\Gamma} \\ \mathbf{A}_{\Gamma s} & \mathbf{A}_{\Gamma\Gamma}^{(s)} \end{pmatrix} \begin{pmatrix} \mathbf{t}_s^{k+1} \\ \mathbf{t}_{\Gamma_s}^{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_s \\ \mathbf{b}_\Gamma^{(s)} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{r}_\Gamma^{(f),k+m} \end{pmatrix},$$

with $\mathbf{A}_{ff} = \mathbf{A}_{ff}(\mathbf{u}^{k+1})$ and

$$\mathbf{r}_\Gamma^{(f),k+m} = \mathbf{b}_\Gamma^{(f)} - \mathbf{A}_{\Gamma f}\mathbf{t}_f^{k+m} - \mathbf{A}_{\Gamma\Gamma}^{(f)}\mathbf{u}_{\Gamma_f}^{k+m}. \tag{12}$$

In general, such surface coupling does not reach convergence and, most of the time a relaxation is required. Let us introduce a relaxation to the Neumann condition. For this, we introduce a relaxation parameter w and we substitute the residual update (12) by

$$\begin{aligned} \mathbf{r}_\Gamma^* &= \mathbf{b}_\Gamma^{(f)} - \mathbf{A}_{\Gamma f}\mathbf{t}_f^{k+m} - \mathbf{A}_{\Gamma\Gamma}^{(f)}\mathbf{u}_{\Gamma_f}^{k+m}, \\ \mathbf{r}_\Gamma^{(f),k+m} &= w\mathbf{r}_\Gamma^* + (1-w)\mathbf{r}^{(f),k+m-1}. \end{aligned}$$

The choice for w is case dependent and more sophisticated relaxations may be required like the Aitken method [30] or Compact Interface Quasi-Newton method [22].

4.4 Jacobi *vs* Gauss-Seidel

The algorithmic and computational performances of the block Jacobi (BJ) and block Gauss-Seidel (BGS) methods have been extensively studied in [11] in the context of Fluid-structure interactions. We summarize here the main conclusions, which apply here as well, for the sake of completeness.

Regarding algorithmic performance, BGS has a faster convergence rate and more robust convergence properties (twice faster in synthetic cases). Regarding computational performance, the block Jacobi method is parallel in the sense that both systems can be solved at the same time, while the second one serializes the execution. In terms of parallel performance, BJ should ideally be the preferred choice. However, in [11], it was demonstrated that overloading the computational nodes with an over-subscription strategy together with a mechanism to slow down MPI (referred to as a barrier) can make the BSG method as efficient as the BJ one.

In the current study, where we also examine volume and surface couplings, there is an additional significant aspect to consider in relation to Fluid-Structure Interaction (FSI) problems. Upon examining the surface coupling in Equation (10), it becomes evident that we have introduced an extra restriction to achieve convergence in the surface coupling. This additional coupling is absent in the case of volume coupling, where we adopt a monolithic approach to address the heat transfer problem throughout Ω. In addition, the convergence of the surface coupling depends on many factors such as the differences in properties, the relative size of the subdomains, relaxation parameters, etc. Simple examples serve as valuable tools for quickly grasping insights into the behavior of an algorithm. We will now introduce three examples to illustrate the convergence behavior of the block Gauss-Seidel and Jacobi methods for various diffusivity ratios, emphasizing the significance of the relaxation parameter. In two of the examples, we consider a fluid domain $[0,1]\times[0,1]$ and a solid domain $[0.5,1]\times[0,1]$ (the solid domain is typically thinner in practical scenarios), coupled at the interface on the rightmost side of the fluid. In this scenario, we only solve for the temperature equation, or equivalently, we set the fluid velocity to zero. A Neumann transmission condition is imposed on the fluid interface, and a Dirichlet condition is applied to the solid interface. All properties are set to unity, and the only variable parameter is the ratio between the diffusivity of the solid and that of the fluid, defined as follows:

$$\alpha_s = \frac{k_s}{\rho_s C_{p,s}}, \quad \alpha_f = \frac{k_f}{\rho_f C_{p,f}}, \quad \text{ratio} = \frac{\alpha_s}{\alpha_f}.$$

In the following examples, the convergence criterion is based on the L^2-norm of the residual of the Neumann transmission condition. Referring to equation (12), the Neumann residual is defined as

$$\text{Neumann residual} = \frac{\|\mathbf{r}_\Gamma^{(f),k+m} - \mathbf{r}^{(f),k+m-1}\|}{\|\mathbf{r}^{(f),k+m}\|}.$$

BGS vs *BJ for different ratios.* Figure 5 shows the results and the convergence history of the different cases considering diffusivity ratios of 1, 10 and 100, respectively. A relaxation factor $w = 0.5$ has been selected. Two primary conclusions can be drawn from the figure. Firstly, the diffusivity significantly impacts the convergence of both coupling methods, with lower ratios resulting in poorer convergence compared to higher ones. Secondly, we note the superior convergence of the Gauss-Seidel method compared to the Jacobi method, with the latter failing to converge for a unit ratio.

Relaxation for different ratios. We address the identical problem as previously, but this time exclusively employing the BGS method, known for its superior convergence. We consider this time looking at the impact of the relaxation parameter, the results being reported in Figure 6. We affirm the substantial impact of the relaxation parameter, which can hinder, and event prevent convergence. A sufficiently small relaxation factors enable to reach convergence in all cases, while it can slower convergence if is too small, as evidenced by the results of the high diffusivity ratios.

We've just outlined and illustrated the algorithmic challenges associated with adopting the surface coupling iterative approach. On the other side, from a computational standpoint, we anticipate that the volume of data transferred through volumetric interpolation is much greater in the case of volume coupling. We have therefore the following aspects to take into consideration when carrying out a conjugate heat transfer problem:

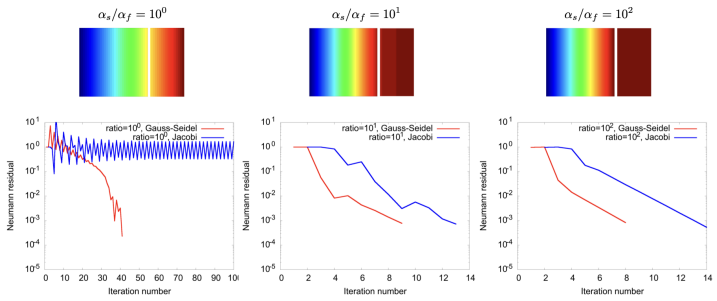

Figure 5: Convergences of BGS and BJ for different diffusivity ratios.

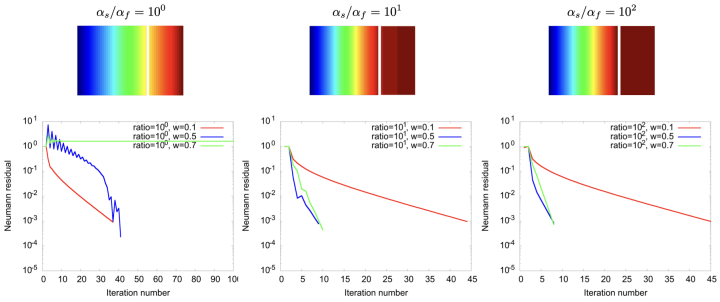

Figure 6: Convergences of the BGS with different relaxation parameters w for different diffusivity ratios.

- Coupling strategy: volume *vs* surface;
- Iterative scheme: BJ *vs* BGS;
- Resources: classical *vs* overloading *vs* overloading/barrier.

According to what has just been pointed out, the best option seems to be volume + BGS + overloading/barrier. The question is: does these qualities compensate the amount of data to exchange through MPI?

Before presenting some experimental results to answer this question running a practical example, we will now introduce the overloading strategy, concentrating on the Gauss-Seidel method.

5 PARALLEL ACCELERATION OF GAUSS-SEIDEL

In the case of multi-physics coupling, a classical strategy consists in running each code in parallel on different cores, thus using exclusive resources. Overloading a node consists in letting the two codes occupy all the resources at the same time. As the Gauss-Seidel method is a serial iterative method, while `code1.x` is running, `code2.x` is waiting for the data to be exchanged. Let's consider the case where the same sequential time is required for the two codes to run one iteration. Then, having n_p cores available, it is logical to choose $n_p/2$ cores for each code. If we overload, and the codes scale perfectly, then the time-to-solution is expected to be divided by two by overloading. This expected behavior is illustrated by synthetic traces in Figure 7 for exclusive and overloaded resources.

However, as pointed out in [11], when waiting for a message, the code enters an MPI busy wait that consumes CPU cycles at the expense of the other code being executed on the same cores. The

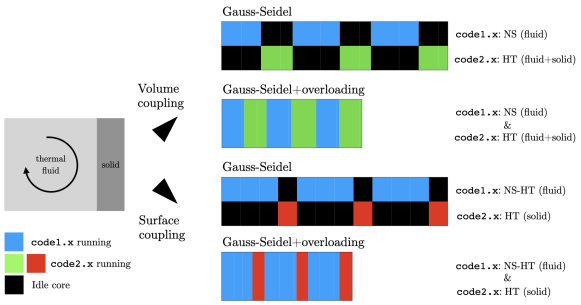

Figure 7: Different coupling and resources strategies for the Gauss-Seidel coupling.

desirable situation would be that the code waiting for the other does not use CPU cycles, leaving the whole computational capacity to the code doing useful work. To achieve this, we use the DLB library [6], a library that improves the performance of hybrid applications by addressing the load imbalance at runtime. In addition to its load balancing capabilities, DLB provides other functionalities aimed at enhancing application performance during execution. The feature employed in this work, as explained in [11], is a barrier between different MPI processes that are running in the same node: this barrier, named `DLB_node_barrier` uses the shared memory of the node to synchronize processes without MPI communication. The main difference between the `DLB_node_barrier` and an `MPI_Barrier` is that processes waiting in the `DLB_node_barrier` do not do a busy

wait (i.e. do not consume cycles) and that the `DLB_node_barrier` works only between processes inside a computational node (i.e. cannot be used between processes running in different nodes).

The workflow is explained in Figure 8, where the tasks carried out by `code1.x` are indicated in blue on the top and the ones of `code2.x` in red on the bottom. In step 1), `code1.x` is running while `code2.x` is in a `DLB_node_barrier`. When its iterations are finished, `code1.x` gets into the `DLB_node_barrier` in step 2) and then releases `code2.x`. Then the data exchange can take place in step 3). `code1.x` gets into the barrier for the next iteration in step 4) while `code2.x` starts executing its iterations. `code2.x` then finishes its iterations and gets into the barrier in step 5) so that data exchange can take place in step 6). The two synchronization points depicted in the figure and provided by `DLB_node_barrier` are identified by rectangles with dotted and continuous lines, respectively. It should be noted that with a careful implementation, the call to the barrier consists of just one line of code.

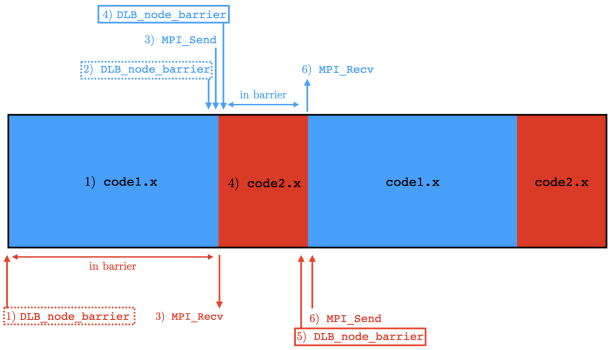

Figure 8: Workflow when overloading resources with `DLB_node_barrier`.

6 CONJUGATE HEAT TRANSFER IN AN ITER TOKAMAK

6.1 Context

This application considers the solution of a sector of a Tokamak geometry. In this scenario, we will evaluate the effectiveness of the BGS method in both surface and volume couplings. The goal is to assess the performance of the overload strategy and examine the efficiency of the proposed overloading/barrier strategy.

The Tokamak is a fusion device composed of 9 annular sectors that together form a toroidal vacuum chamber. Within the central chamber, plasma undergoes fusion reactions. Neutrons resulting from these reactions impinge upon the interior walls of the sectors and generate heat. Each sector incorporates a circular water circuit, facilitating the flow of water between the interior and exterior panels. This circuit is an integral part of the primary heat transfer system, serving the crucial role of capturing the accumulated heat. The captured heat is subsequently transformed into energy and re-injected to cool the walls of the vacuum chamber, ensuring the efficient operation of the Tokamak. The deposition of neutrons are modeled through a volumetric heat source Q_s. We focus here on one of the sector (green one in Figure 9) with inflow water temperature at 100 ^{o}C and velocity of 0.2 m/s in lower section and evacuated in upper section. Figure 9 shows the specific sector solved (in green), as well as details of the geometries and meshes used to compare the volume and surface couplings.

6.2 Solution strategy

The simulations have been carried with Alya high-performance computational mechanics code [29] on Marenostrum4 supercomputer [2]. Regarding the supercomputer, the general-purpose block of Marenostrum4 was selected. It has 48 racks with 3456 nodes. Each node has two Intel Xeon Platinum 8160 24C at 2.1 GHz, each with 24 processors, amounting to a grand total of 165,888 processors and a main memory of 390 Terabytes. Regarding the modules used, we have selected Intel compiler `intel/2018.4`, Intel MPI library `impi/2018.4` and DLB library `dlb/3.2`. Alya software is a licensed software [1]. The specific version used in this work can be retrieved using the commit hash `a0fac71de`.

The algebraic system stemming from the discretization of the Navier-Stokes equations is solved implicitly, using the approach described in [8]. The continuity equation is solved with the Deflated Conjugate Gradient (DCG) method [13] and the momentum equations with the GMRES method [21]. The temperature equation is also solved implicitly, using a GMRES method when using the volume coupling (the problem is not symmetric due to the presence of advection). Regarding the surface coupling, and as the resulting algebraic system is Symmetric Positive Definite in the solid domain, the Deflated Conjugate Gradient is used while the GMRES method is employed in the fluid. In all cases, the temperature equation is solved in a staggered way, after solving the Navier-Stokes equations.

6.3 Performance results

Figure 10 shows some results of the simulation obtained, revealing the intricate nature of the flow field. Additionally, noticeable temperature hotspots are observed on the left, coinciding with the application of the neutronic source.

To ensure equitable comparisons, we have kept the resources consistent across all tests, utilizing a total of 144 cores, that is 3 computing nodes. Furthermore, the number of iterations has been set as a constant. The distribution of the number of cores for cases employing exclusive resources has been carried out using a rule-of-thumb, as illustrated in Table 1, which also indicates the number of elements utilized for each case.

	Volume coupling		Surface coupling	
	Fluid (8.5M)	All (12.1M)	Fluid (8.5M)	Solid (3.5M)
Exclusive resources	72	72	120	24
Barrier	144	144	144	144
overloaded/Barrier	144	144	144	144

Table 1: Number of cores used in the different cases tested. The number of elements is also specified within parentheses.

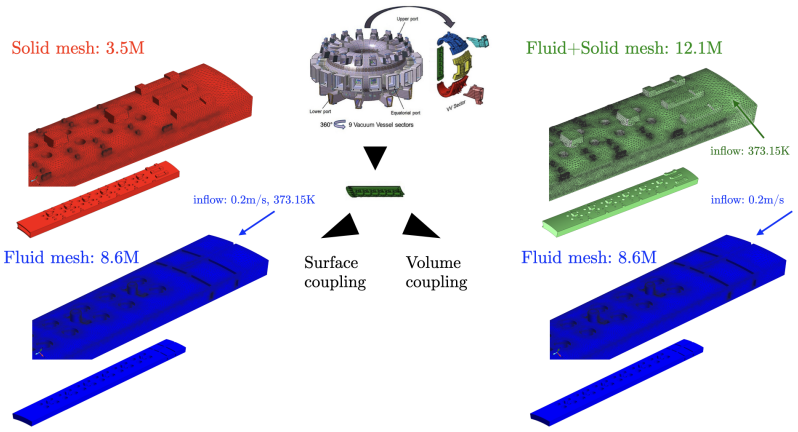

Figure 9: Surface and volume couplings.

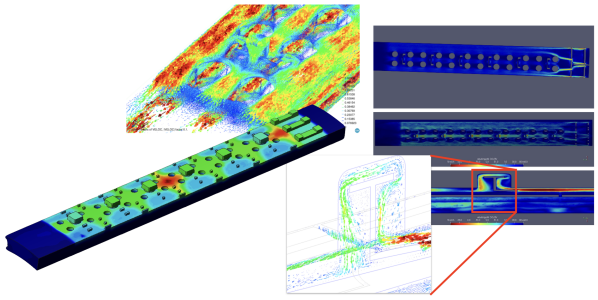

Figure 10: Snapshots of the solution. Temperature and velocity fields.

Figure 11 shows the traces obtained on the 6 cases executed. They all have the same time scale to enable direct comparison. From the results of the traces, we can draw the following conclusions:

- The duration of surface coupling with exclusive resources is slightly shorter than that of volume coupling, the time for the volume coupling being 1.4 times higher. This discrepancy can be attributed to the higher total number of elements to be simulated in the volume coupling approach (8.5M+12.1M) compared to the surface coupling (8.5M+3.5M).
- The ratio of the total number of elements for volume coupling to that of surface coupling is 1.7. This higher ratio, as compared to 1.4, is attributed to the solution cost of the Navier-Stokes equations being greater than that of the temperature equation.
- The overloaded resources approach without barrier always gives a higher duration than exclusive resources.
- The overloaded resources approach is effective only if the barrier is present.
- The overloaded resources approach with barrier gives the following speedups with respect to the exclusive resources approach:

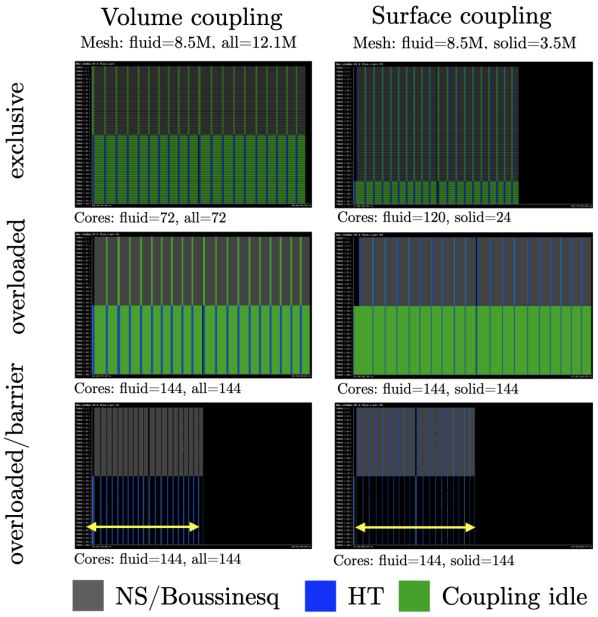

Figure 11: Traces of the run on a total of 144 cores (3 computing nodes). The x-axis is the time and the y-axis the number of cores. Colors indicate the system being solved, while green color indicates processes waiting in the coupling data transfer.

  - Volume coupling: 1.91
  - Surface coupling: 1.35
- Surface and volume couplings give the same computing time when overloading the nodes with the barrier.

Given all these remarks, it can be deduced that volume coupling demonstrates superior performance, indicating minimal impact from coupling communications on computation time. Additionally, volume coupling eliminates the need for an iterative algorithm to converge the surface coupling on the interface, as mentioned

earlier. Moreover, the overloaded resources approach alleviates the necessity to allocate cores separately for the two codes, allowing all codes to utilize the entirety of available resources. This aspect holds significant importance for the end user of the code. We thus conclude that volume + BGS + overloading/barrier is in the present case the best strategy to carry out the conjugate heat transfer. It's worth noting that for the proposed strategy with the barrier to effectively utilize all resources for both codes, each code must be strong scalable across the entire range of available cores. The high speedup obtained confirms for this example confirms this fact.

6.4 Coupling scalability

We will now assess the scalability of the couplings by varying the number of cores and adjusting the mesh size. It is anticipated that the computational load of the volume coupling will increase significantly faster with larger mesh sizes compared to the surface coupling. To investigate this, we've employed a mesh multiplication strategy that recursively increases the number of elements in parallel by a factor of 8 [10]. The original mesh is denoted as MM0 (0-mesh multiplication level), while the multiplied mesh is denoted as MM1 (1-mesh multiplication level). In this scenario, the volume coupling involves 68M and 97M elements for the fluid and complete domains, respectively, while the surface coupling involves 68M and 28M elements for the fluid and solid domains, respectively.

Table 2 shows the relative times for volume and surface couplings using the overloading with barrier strategy, for different number of cores and meshes. "Coupl." refers to the coupling time and "Total" to the total time of the execution. The percentage indicates the time taken by the coupling with respect to the total.

		Volume coupling			Surface coupling		
# cores	Mesh	Coupl.	Total	(%)	Coupl.	Total	(%)
72	MM0	0.6	100.0	0.6	0.6	99.3	0.6
144	MM0	0.7	56.1	1.3	0.6	53.2	1.1
576	MM1	2.2	229.6	1.0	0.9	227.9	0.4

Table 2: Relative times for different coupling strategies, number of cores and meshes. "Coup." refers to the coupling time and "Total" the total time of the execution. The percentage indicates the time taken by the coupling with respect to the total.

Analyzing the first two rows reveals the strong scalability of the coupling. Despite the percentage being multiplied by a factor of 2, we observe that the coupling time remains constant (resulting in a total scaling efficiency of 90%). Notably, the volume coupling demonstrates a slightly higher percentage of 1.3 compared to the surface coupling which is 1.1, although still considered low. Figures 12 shows the matrix of data communications from the solid/complete domain to the fluid subdomain for the six cases under study. The color indicates the size of the data to be transfer from the y-axis ranks to the x-axis ranks. We can draw some expected conclusions. Firstly, the volume coupling matrix is much denser than the surface coupling one. Secondly, we also observe that as the number of cores increases, the number of neighbors increases but the size of the data transferred decreases.

Comparing the first and third rows of Table 2 allows us to gauge the weak scalability of the coupling, considering that we have multiplied both the number of elements and cores by a factor of 8. As anticipated, we observe that with the increase in mesh size, the execution time for the volume coupling increases more rapidly (from 0.6 to 2.2) compared to that of the surface coupling (from 0.6 to 0.9). Nevertheless, upon examining the percentages, we observe that they remain below 1%.

We thus conclude that even for the large meshes using a first level of mesh multiplication, the volume coupling is still competitive with the surface coupling.

7 CONCLUSION

We have introduced parallel coupling strategies for addressing conjugate heat transfer problems. On the algorithmic front, our approach involves comparing volume and surface couplings. The coupling methods have been outlined at the algebraic level, offering the advantage of yielding the same solutions as the monolithic problem. Subsequently, we presented a multi-code strategy to tackle these thermally coupled problems, exploring three distinct approaches: utilizing exclusive resources, employing overload resources, and combining overloaded resources with a barrier. The inclusion of a barrier serves to reduce the activity of busy waits during data exchange between the codes.

In concluding, by solving a practical example, we demonstrated that employing volume coupling in conjunction with the overloaded resources strategy can lead to a nearly twofold reduction in time-to-solution. This strategy also eliminates the need to specify the number of cores dedicated to each physics, providing an additional advantage. In addition, scalability tests show that the coupling time remains low (around 1%) compared to the total execution time, even for large meshes. The success of this proposed strategy rests on the assumption that the different codes to be coupled continue to scale when utilizing the total available resources.

ACKNOWLEDGEMENTS

This work was partly funded by the FusionCAT project (001-P-001722) which has been 50% co-financed by the European Fund for Regional Development of the European Union within the framework of the 2014-2020 ERDF Operational Program of Catalonia, and with the support of the Generalitat of Catalonia and by the Generalitat de Catalunya SGR-Cat 2021 programme.

REFERENCES

[1] 2024. Alya, high-performance computational mechanics code. https://alya.gitlab.bsc.es/alya/alya/-/wikis/home. Accessed: 2024-02-21.
[2] 2024. Marenostrum4 system architecture. https://www.bsc.es/marenostrum/marenostrum/technical-information. Accessed: 2024-02-21.
[3] Ramon Codina and Guillaume Houzeaux. 2006. Numerical approximation of the heat transfer between domains separated by thin walls. *International Journal for Numerical Methods in Fluids* 52, 9 (2006), 963–986. https://doi.org/10.1002/fld.1215 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/fld.1215
[4] Florent Duchaine, Alban Corpron, Lorenzo Pons, Vincent Moureau, Franck Nicoud, and Thierry Poinsot. 2009. Development and assessment of a coupled strategy for conjugate heat transfer with Large Eddy Simulation: Application to a cooled turbine blade. *International Journal of Heat and Fluid Flow* 30, 6 (2009), 1129–1141. https://doi.org/10.1016/j.ijheatfluidflow.2009.07.004
[5] Marc-Paul Errera, Rocco Moretti, Julien Mayeur, Matteo Gelain, Lionel Tessé, Jean-Michel Lamet, and Emmanuel Laroche. 2020. A numerical predictive model for conjugate heat transfer with radiation. *International Journal of Heat and Mass*

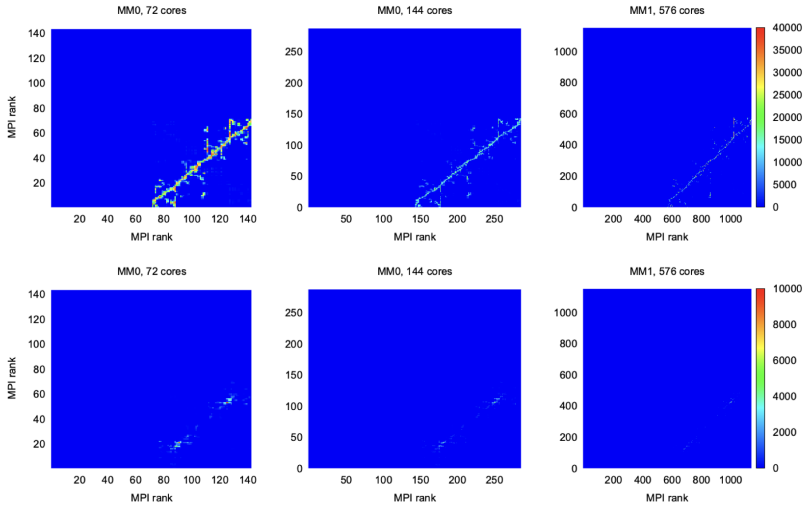

Figure 12: Communication matrices for volume (top) and surface (bottom) couplings.

Transfer 160 (2020), 120155. https://doi.org/10.1016/j.ijheatmasstransfer.2020.120155

[6] Marta Garcia-Gasulla, Jesus Labarta, and Julita Corbalan. 2014. Hints to improve automatic load balancing with LeWI for hybrid applications. *J. Parallel and Distrib. Comput.* 74, 9 (2014), 2781–2794. https://doi.org/10.1016/j.jpdc.2014.05.004

[7] Guillaume Houzeaux. 2002. *A Geometrical Domain Decomposition Method in Computational Fluid Dynamics.* Ph. D. Dissertation. Universitat Politècnica de Catalunya. http://hdl.handle.net/10803/6858 Supervisor: Ramon. Codina.

[8] G. Houzeaux, R. Aubry, and M. Vázquez. 2011. Extension of fractional step techniques for incompressible flows: The preconditioned Orthomin(1) for the pressure Schur complement. *Computers Fluids* 44, 1 (2011), 297–313. https://doi.org/10.1016/j.compfluid.2011.01.017

[9] Guillaume Houzeaux, Juan Carlos Cajas, Marco Discacciati, Beatriz Eguzkitza, Abel Gargallo-Peiró, Matias Rivero, and Mariano Vázquez. 2017. Domain decomposition methods for domain composition purpose: Chimera, overset, gluing and sliding mesh methods. *Archives of Computational Methods in Engineering* 24, 4 (2017), 1033–1070. https://doi.org/10.1007/s11831-016-9198-8

[10] Guillaume Houzeaux, Raúl de la Cruz, Herbert Owen, and Mariano Vázquez. 2013. Parallel uniform mesh multiplication applied to a Navier–Stokes solver. *Computers Fluids* 80 (2013), 142–151. https://doi.org/10.1016/j.compfluid.2012.04.017 Selected contributions of the 23rd International Conference on Parallel Fluid Dynamics ParCFD2011.

[11] Guillaume Houzeaux, Marta Garcia-Gasulla, Juan Carlos Cajas, Alfonso Santiago, Charles Moulinec, and Mariano Vázquez. 2020. Parallel Multiphysics Coupling: Algorithmic and Computational Performances. *International Journal of Computational Fluid Dynamics* 34, 7-8 (2020), 486–507. https://doi.org/10.1080/10618562.2020.1783440

[12] G. Houzeaux and J. Pr´. 2008. A Variational Subgrid Scale Model for Transient Incompressible Flows. 22, 3 (2008), 135–152. https://doi.org/10.1080/10618560701816387

[13] Rainald Löhner, Fernando Mut, Juan Raul Cebral, Romain Aubry, and Guillaume Houzeaux. 2011. Deflated preconditioned conjugate gradient solvers for the pressure-Poisson equation: Extensions and improvements. *Internat. J. Numer. Methods Engrg.* 87, 1-5 (2011), 2–14. https://doi.org/10.1002/nme.2932

[14] Aleksey Luikov and Theodore Perelman et al. 1970. Heat transfer from a plate in a compressible gas flow. *International Journal of Heat and Mass Transfer* 13, 8 (1970), 1261–1270. https://doi.org/10.1016/0017-9310(70)90067-0

[15] Asif Hussain Malik, M. S. I. Alvi, Shahab Khushnood, F. M. Mahfouz, M. K. K. Ghauri, and Ajmal Shah. 2012. Numerical study of conjugate heat transfer within a bottom heated cylindrical enclosure. In *Proceedings of 2012 9th International Bhurban Conference on Applied Sciences and Technology (IBCAST).* IEEE, 213–220. https://doi.org/10.1109/IBCAST.2012.6177556

[16] Message Passing Interface Forum. 2021. *MPI: A Message-Passing Interface Standard Version 4.0.* https://www.mpi-forum.org/docs/mpi-4.0/mpi40-report.pdf

[17] John G Michopoulos, Charbel Farhat, and Jacob Fish. 2005. Modeling and Simulation of Multiphysics Systems. *Journal of Computing and Information Science in Engineering* 5, 3 (09 2005), 198–213. https://doi.org/10.1115/1.2031269

[18] Michael Kevin Patterson, Xiaojin Wei, Yogendra Joshi, and Ravi Prasher. 2004. Numerical study of conjugate heat transfer in stacked microchannels. In *The Ninth Intersociety Conference on Thermal and Thermomechanical Phenomena In Electronic Systems (IEEE Cat. No.04CH37543)*, Vol. 1. 372–380 Vol.1. https://doi.org/10.1109/ITHERM.2004.1319199

[19] Theordore Perelman. 1961. On conjugated problems of heat transfer. *International Journal of Heat and Mass Transfer* 3, 4 (1961), 293–303. https://doi.org/10.1016/0017-9310(61)90044-8

[20] Alfio Quarteroni and Alberto Valli. 1999. *Domain Decomposition Methods for Partial Differential Equations.* Clarendon Press. https://books.google.es/books?id=CswwZ9Ca9IIC

[21] Yousef Saad. 2003. *Iterative Methods for Sparse Linear Systems* (second ed.). Society for Industrial and Applied Mathematics. https://doi.org/10.1137/1.9780898718003 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9780898718003

[22] Alfonso Santiago, Miguel Zavala-Aké, Ricard Borrell, Guillaume Houzeaux, and Mariano Vázquez. 2020. HPC compact quasi-Newton algorithm for interface problems. *Journal of Fluids and Structures* 96 (2020), 103009. https://doi.org/10.1016/j.jfluidstructs.2020.103009

[23] Ralf Schiele and Sigmar L. K. Wittig. 2000. Gas Turbine Heat Transfer: Past and Future Challenges. *Journal of Propulsion and Power* 16 (2000), 583–589.

[24] Robert Siegel and Morris Perlmutter. 1963. Laminar Heat Transfer in a Channel With Unsteady Flow and Wall Heating Varying With Position and Time. *Journal of Heat Transfer* 85, 4 (11 1963), 358–365. https://doi.org/10.1115/1.3686125

[25] Moustafa Soliman and Harold A. Johnson. 1967. Transient Heat Transfer for Turbulent Flow Over a Flat Plate of Appreciable Thermal Capacity and Containing Time-Dependent Heat Source. *Journal of Heat Transfer* 89, 4 (11 1967), 362–370. https://doi.org/10.1115/1.3614398

[26] Moustafa Soliman and Harold A. Johnson. 1968. Transient heat transfer for forced convection flow over a flat plate of appreciable thermal capacity and containing an exponential time-dependent heat source. *International Journal of Heat and Mass Transfer* 11, 1 (1968), 27–38. https://doi.org/10.1016/0017-9310(68)90061-6

[27] Ephraim M. Sparrow and Frederico Farias. 1968. Unsteady heat transfer in ducts with time-varying inlet temperature and participating walls. *International Journal of Heat and Mass Transfer* 11, 5 (1968), 837–853. https://doi.org/10.1016/0017-9310(68)90128-2

[28] Hansong Tang, Ronald Haynes, and Guillaume Houzeaux. 2020. A Review of Domain Decomposition Methods for Simulation of Fluid Flows: Concepts, Algorithms, and Applications. *Archives of Computational Methods in Engineering* 28 (01 2020). https://doi.org/10.1007/s11831-019-09394-0

[29] Mariano Vazquez, Guillaume Houzeaux, Seid Koric, Antonio Artigues, Jazmin Aguado-Sierra, Ruth Arís, Daniel Mira, Hadrien Calmet, Fernando Cucchietti, Herbert Owen, Ahmed Taha, Evan Dering Burness, José Marí Cela, and Mateo Valero. 2016. Alya: Multiphysics engineering simulation towards exascale. *J. Comput. Sci.* 14 (2016), 15–27. https://doi.org/10.1016/j.jocs.2015.12.007

[30] Wolfgang A. Wall, Steffen Genkinger, and Ekkehard Ramm. 2007. A strong coupling partitioned approach for fluid–structure interaction with free surfaces. *Computers Fluids* 36, 1 (2007), 169–183. https://doi.org/10.1016/j.compfluid.2005.08.007 Challenges and Advances in Flow Simulation and Modeling.