Abstract

Domain composition methods (DCM) consist in obtaining a solution to a problem on a domain, from the formulations of the same problem expressed on various subdomains. These methods have therefore the opposite objective of domain decomposition methods (DDM). However, they are sometimes based on the same methodology as after decomposing, DDM have to recompose. As a consequence, in the literature, the term DDM has many times substituted DCM.

In contrast with DCM, DDM are usually applied to matching meshes as their purpose consists mainly in distributing the work in parallel environments. DCM are powerful techniques that can be used for different purposes: to simplify the meshing of a complex geometry by dividing it into different meshable pieces; to perform local refinement to adapt to local mesh requirements; to treat subdomains in relative motion (Chimera, sliding mesh), etc.

The term DCM is generic and does not give any clue about how the fragmented solutions on the different subdomains are composed into a global one. In the literature, many methodologies have been proposed: they are mesh-based, formulation-based, or algebraic-based. In mesh-based formulations, the coupling is achieved at the mesh level, before the governing equations are assembled into an algebraic system (mesh conforming, Shear-Slip Mesh Update, HERMESH). The formulation-based counterpart recomposes the solution from the strong or weak formulation itself, and are implemented during the assembly of the algebraic system on the subdomain meshes. The different coupling techniques can be formulated at the strong and continuous level, at the weak and continuous/discrete level (iteration-by-subdomains, mortar element, mesh free interpolation). Although the different methods usually lead to the same solutions at the continuous level, which usually coincide with the solution of the problem on the original domain, they have very different behaviors at the discrete level and accept very different implementations. Eventually, algebraic-based formulations treat the composition of the solutions directly on the matrix and right-hand side of the
individual subdomain algebraic systems.

The present work concentrates on the parallel implementations of some algebraic-based domain composition methods. The advantages of algebraic-based formulation over the mesh or formulation-based techniques, is that the method is relatively problem independent, its implementation can be hidden in the iterative solver operations, and the method can be made implicit and explicit relatively easily.

**Keywords:** parallel (MPI), domain composition method, domain decomposition method, finite element, subdomain coupling, computational mechanics.
1 Introduction

Domain composition methods (DCM) consist in obtaining a solution to a problem on a domain, from the formulations of the same problem expressed on various subdomains. These methods have therefore the opposite objective of domain decomposition methods (DDM). However, they are sometimes based on the same methodology as after decomposing, DDM have to recompose. As a consequence, in the literature, the term DDM has many times substituted DCM.

There are different reasons for the interest in the coupling of meshes. Among others: to assemble components or meshes obtained from different sources, a typical case in industry where components are designed in different departments of the company; to simplify the meshing of complex geometries by dividing it into different meshable pieces; to perform local refinement to adapt to local mesh requirements; to couple multi-physics problems and/or subdomains in relative motion; to optimize the relative positions of some components without having to remesh the whole computational domain. All of these situations could appear simultaneously and are in fact quite frequently found in actual problems. The objective of this work is to be able to deal with the majority of them in a practical and robust way.

Several DCM strategies have been proposed in the literature, divided mainly into two different families. These families are referred to herein as mesh based and formulation based.

On the one hand, mesh-based methods aim at recomposing a conforming mesh at the intersection of the subdomain meshes. Let us mention: the so-called DRAGON (Direct Replacement of Arbitrary Grid Overlapping by Non-structured grid) meshes [1]; we also find a mesh merging technique to create a conforming mesh coming from two independent meshes and consisting in merging the meshes from their intersection [2, 3]; the Shear-Slip Mesh Update Method reconnects sliding meshes with a conforming mesh in the gap [4]; the HERMESH method joins meshes by creating an overlapping layer of elements (extensions) from one subdomain to the other [5].

On the other hand, formulation-based techniques consider the coupling at the equation level. Let us mention iteration-by-subdomain methods based on transmission conditions [6], mesh free interpolations [7, 8], and method based on constraint impo-sitions like the FETI or mortar methods [9, 10, 11, 12].

This present work studies a domain composition method for disjoint subdomains, belonging to the formulation-based family. The method is implemented at the algebraic level, making the method problem independent and implicit. This last point is very important as, unlike classical iteration-by-subdomain methods, no additional iterative loop is required to converge the coupling. In Section 2, we introduce the equivalence between domain composition method at the continuous, discrete and algebraic level. Section 3 then studies a particular method, an algebraic Dirichlet/Neumann coupling. Both explicit and implicit versions are detailed. The end of the section is devoted to the case of non-matching meshes, some possible data transmission techniques, and conservation issues. Section 4 presents some implementation aspects,
with special emphasis on parallelization. Eventually, last section presents an example of application for the Navier-Stokes equations and an example to assess the parallel performance of the proposed coupling strategy.
2 Domain Composition Methods

Let us consider the following second order partial differential equation

\[ \mathcal{L}(u) = f \quad \text{in } \Omega, \]
\[ u = 0 \quad \text{on } \partial \Omega, \]

with \( \mathcal{L}(u) = -\varepsilon \Delta u \). This equation was selected for the sake of clarity but the exposition of this section holds for a general advection-diffusion-reaction equation. In the following, we will try to keep the reading as light as possible by avoiding heavy notations. Nevertheless, the formal definitions and demonstrations can be found in the cited references.

A subdomain coupling strategy consists in obtaining a unique and global solution from local solutions obtained on separate subdomains. These subdomains can be disjoint or overlapping, being the domain \( \Omega \) the union of the subdomains. Ideally, one require to obtain the same global solution as the one-domain solution. In general, this is achieved by imposing both the continuity of the variable and its flux at the interface of the subdomains, as illustrated by Figure 1. Many methods have been devised in the literature to impose these two continuities. On the one hand, transmission conditions of Dirichlet/Neumann, Dirichlet/Robin, Robin/Robin types impose the continuities of combinations of the variable and its flux at the interfaces, and can be applied to disjoint and overlapping subdomains; Dirichlet/Dirichlet (Schwarz) methods require overlapping subdomains. On the other hand, Lagrange multipliers can be used to relax the Dirichlet condition by imposing it in a weaker sense.

In the following we will explain the classical road to devise domain composition methods for disjoint subdomains. We will start with the variational form at the continuous level, and then go to its discretized form. We will briefly explain why the equivalence between the one-domain and the multi-domain formulations at the discrete level cannot be established. Then we will devise a proper method, at the algebraic level,
which leads to the same solution as the one-domain problem under some conditions. However, we will not consider the case of overlapping subdomains, and therefore will not introduce the Schwarz method. Table 1 summarizes the methods and aspects we are going to cover.

<table>
<thead>
<tr>
<th></th>
<th>D/N</th>
<th>D/R</th>
<th>R/R</th>
<th>D/D</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Disjoint subdomains</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Continuous</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>×</td>
</tr>
<tr>
<td>Discrete</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>×</td>
</tr>
<tr>
<td>Algebraic</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>×</td>
</tr>
<tr>
<td><strong>Overlapping subdomains</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Continuous</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Discrete</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>Algebraic</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

Table 1: Some DDM/DCM methods: Dirichlet/Neumann (D/N), Dirichlet/Robin (D/R), Robin/Robin (R/R) and Dirichlet/Dirichlet (D/D) or Schwarz. yes: considered in this work. no: not considered in this work. ×: does not exist.

2.1 Continuous level

We now consider the variational formulation corresponding to equation (1). Let $a(u, v)$ be the bilinear form associated to $L(u)$ defined as:

$$a(u, v) := \varepsilon(\nabla u, \nabla v).$$

The weak problem consists in finding $u \in U^0$ such that

$$a(u, v) = \langle f, v \rangle \quad \forall v \in V^0 \quad (2)$$

together with boundary conditions, and where $U^0$ and $V^0$ are appropriate spaces for the solution and test functions, which cancel on the boundary. $\langle \cdot, \cdot \rangle$ denotes the duality paring.

Let us now consider a two-subdomain formulation. For the sake of clarity, we will only consider disjoint subdomains; the case of overlapping subdomains can be treated similarly [13]. We first need to introduce some notations, as illustrated in Figure 2. The subscript indicates the subdomain supports of the spaces; superscript 0 indicates that the functions vanish on the respective subdomain boundaries; $\Lambda$ is the trace operator on the interface $\Gamma$; finally, $E_i$ are some extension operators from the trace space to the respective subdomain spaces $V_i$, $i = 1, 2$. The concept of extension
Figure 2: Two disjoint subdomains. Notations of geometrical entities, functional spaces and extension operator.

consists in extending a function living on the interface onto a subdomain. At the discrete level for example, a possible extension operator would be to extend linear functions on the interface nodes linearly to zero on the first interior nodes. Let us define $a_i(u,v)$ the bilinear form restricted to $\Omega_i$ and $\langle f,v \rangle_{\Omega_i}$ the duality paring in $\Omega_i$.

The two-subdomain formulation reads:

\[
\begin{align*}
  a_1(u_1, v_1) &= \langle f, v_1 \rangle_{\Omega_1} & \forall v_1 \in V_{10} \\
  u_1 &= u_2 \\
  a_2(u_2, v_2) &= \langle f, v_2 \rangle_{\Omega_2} \\
  a_2(u_2, E_2 \mu) + a_1(u_1, E_1 \mu) &= \langle f, E_1 \mu \rangle_{\Omega_1} + \langle f, E_2 \mu \rangle_{\Omega_2} & \forall \mu \in \Lambda
\end{align*}
\]

In [6], two important results are shown:

- System (3) is equivalent to system (2);
- System (3) implies that $\varepsilon \nabla u_1 \cdot n_2 = \varepsilon \nabla u_2 \cdot n_2$;

According to the last item, we can rewrite the extension-based variational formulation (3) as a Dirichlet/Neumann variational formulation as:
Dirichlet/Neumann variational formulation

\[
\begin{align*}
& a_1(u_1, v_1) = \langle f, v_1 \rangle_{\Omega_1} \quad \forall \ v_1 \in V_1^0 \\
& u_1 = u_2 \quad \text{on } \Gamma \\
& a_2(u_2, v_2) = \langle f, v_2 \rangle_{\Omega_2} \quad \forall \ v_2 \in V_2 \\
& \varepsilon \nabla u_1 \cdot n_2 = \varepsilon \nabla u_2 \cdot n_2 \quad \text{on } \Gamma
\end{align*}
\]

Note that in the third equation, the test functions in \( V_2 \) do not longer vanish on \( \Gamma \) as we have to apply on it the Neumann condition.

This justifies why System (3) is referred to as Dirichlet/Neumann method: a Dirichlet transmission condition (4) is applied to solve subdomain 1, while a Neumann transmission condition (4) is applied to solve system 2. Note that in [13], equivalent results are shown for overlapping subdomains. The equivalence between (3) and (2) is illustrated in Figure 5 (Top). We will now see what happens when we try to apply these formulations at the discrete level.

### 2.2 Discrete level

Let us discretize in space and consider finite functional spaces equivalent to the continuous spaces defined earlier and solve now for \( u_h \). Then, we can show that the discrete counterpart of System (3), referred to as the discrete two-subdomain extension-based variational form, is equivalent to the discrete counterpart of System (2), referred to as the discrete one-domain variational form. However, in practice, the formulation (3) seems not to be convenient to implement, and one would rather use the discretization of (4). It is in fact relatively easier to interpolate a derivative on a boundary than to design an extension operator... Note that discrete extension operators have been successfully devised and applied in [14, 15].

Now let us go back to the continuous formulation. In order to prove the second item of the list of important results, stating that \( \varepsilon \nabla u_1 \cdot n_2 = \varepsilon \nabla u_2 \cdot n_2 \), we have to use the fact that \( L(u) = f \), as shown in [6]. This equality is true at the continuous level, but is not satisfied at the discrete level, that is \( L(u_h) \neq f \). Therefore, if we use the discretized system (4), we will not obtain an equivalent system to (3). This implies that we will neither find the same solution \( u_h \) as if we would have solved the discrete one-domain problem (2). This lack of equivalence is illustrated in Figure 5 (Mid).

### 2.3 Algebraic level

Let us discretize on a finite element mesh the one-domain formulation (2) and construct the corresponding algebraic system. According to the space discretization used, we denote \( u \) the vector of unknowns in \( \Omega \). We obtain:

\[
Au = b.
\]
We denote $\mathbf{u}_1$ and $\mathbf{u}_2$ as the vectors of unknowns of $\Omega_1$ and $\Omega_2$ respectively, excluding the interface vector of unknowns that we denote $\mathbf{u}_\Gamma$. By performing a simple node reordering, System (5) can be written as:

$$
\begin{pmatrix}
A_{11} & 0 & A_{1\Gamma} \\
0 & A_{22} & A_{2\Gamma} \\
A_{\Gamma 1} & A_{\Gamma 2} & A_{\Gamma \Gamma}
\end{pmatrix}
\begin{pmatrix}
\mathbf{u}_1 \\
\mathbf{u}_2 \\
\mathbf{u}_\Gamma
\end{pmatrix}
= 
\begin{pmatrix}
b_1 \\
b_2 \\
b_\Gamma
\end{pmatrix}.
$$

The submatrices of the interface equation come from the assembly involving the element in $\Omega_1$ and $\Omega_2$. With obvious meaning, we denote as $A_{\Gamma \Gamma}^{(1)}$ and $A_{\Gamma \Gamma}^{(2)}$ the submatrices coming from the integration over subdomains 1 and 2, respectively. We can write the latter system as:

\[ \textbf{One-domain algebraic formulation} \]

$$
\begin{pmatrix}
A_{11} & 0 & A_{1\Gamma} \\
0 & A_{22} & A_{2\Gamma} \\
A_{\Gamma 1} & A_{\Gamma 2} & A_{\Gamma \Gamma}^{(1)} + A_{\Gamma \Gamma}^{(2)}
\end{pmatrix}
\begin{pmatrix}
\mathbf{u}_1 \\
\mathbf{u}_2 \\
\mathbf{u}_\Gamma
\end{pmatrix}
= 
\begin{pmatrix}
b_1 \\
b_2 \\
b_\Gamma^{(1)} + b_\Gamma^{(2)}
\end{pmatrix} \tag{6}
$$

The different contributions to the submatrices are illustrated in Figure 3.

Now let us go back to the discrete two-domain formulation, that is the discrete counterpart of (3). To introduce the algebraic domain composition method, we consider two independent meshes for $\Omega_1$ and $\Omega_2$, conforming at the interface, and with duplicated interface nodes. Each submesh is exactly the same as the original mesh shown in Figure 3. We are going to choose carefully the discrete extension operators so that they coincide with the classical shape functions of the interface nodes. This
Figure 4: Extension operator for an interface node coinciding with its classical shape function.

choice is illustrated in Figure 4. Therefore, the algebraic equation coming from the integration and assembly of System (3) for the two-domain formulation reads:

\[
\begin{pmatrix}
A_{11} & A_{1\Gamma_1} & 0 & 0 \\
0 & I & 0 & -I \\
0 & 0 & A_{22} & A_{2\Gamma_2} \\
A_{\Gamma_1 1} & A_{\Gamma_1 \Gamma_1} & A_{\Gamma_2 2} & A_{\Gamma_2 \Gamma_2}
\end{pmatrix}
\begin{pmatrix}
u_1 \\
u_{\Gamma_1} \\
u_2 \\
u_{\Gamma_2}
\end{pmatrix} =
\begin{pmatrix}
b_1 \\
0 \\
b_2 \\
b_{\Gamma_1} + b_{\Gamma_2}
\end{pmatrix},
\]  
(7)

where the submatrices coincide exactly with that of the formulation given by (6), with
\[A_{1\Gamma_1} = A_{1\Gamma}, A_{2\Gamma_2} = A_{2\Gamma}, A_{\Gamma_1 1} = A_{\Gamma_1 \Gamma}, A_{\Gamma_2 2} = A_{\Gamma_2 \Gamma}, A_{\Gamma_1 \Gamma_1} = A^{(1)}_{\Gamma \Gamma}, A_{\Gamma_2 \Gamma_2} = A^{(2)}_{\Gamma \Gamma},
\]
\[b_{\Gamma_1} = b^{(1)}_{\Gamma} \text{ and } b_{\Gamma_2} = b^{(2)}_{\Gamma}.
\]
In the previous equation we dropped the superscript as the interface unknowns has in this case two distinct (but equal through the Dirichlet condition!) values \(u_{\Gamma_1}\) and \(u_{\Gamma_2}\). The first row of the previous system comes from the integration and assembly of Equation (3); the second row is the Dirichlet condition given Equation (3); the third row comes from the equation in subdomain \(\Omega_2\) (3); the last row coincide with the interface equation involving extension operators of Equation (3).

By substituting the second row in the other equations and defining \(u_{\Gamma_1} = u_{\Gamma_2} = u_{\Gamma}\), we recover exactly the one-domain system given by Equation (6). By casting some term to the right-hand side, we can clarify the last equation:
Two-subdomain algebraic formulation - Dirichlet/Residual coupling

\[
\begin{bmatrix}
A_{11} & A_{1}\Gamma_{1} \\
0 & I
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_{\Gamma_1}
\end{bmatrix}
= 
\begin{bmatrix}
b_1 \\
u_{\Gamma_2}
\end{bmatrix}
\]
\[
\begin{bmatrix}
A_{22} & A_{2}\Gamma_{2} \\
A_{\Gamma_22} & A_{\Gamma_2}\Gamma_{2}
\end{bmatrix}
\begin{bmatrix}
u_2 \\
u_{\Gamma_2}
\end{bmatrix}
= 
\begin{bmatrix}
b_2 \\
b_{\Gamma_2}
\end{bmatrix}
+ 
\begin{bmatrix}
0 \\
r_{\Gamma_1}
\end{bmatrix}
\]  

(8)

where we have defined the residual \( r_{\Gamma_1} \) as

\[
r_{\Gamma_1} = b_{\Gamma_1} - A_{\Gamma_11}u_1 - A_{\Gamma_1}\Gamma_1u_{\Gamma_1}.
\]

(9)

The first system solves in subdomain 1 by imposing the Dirichlet condition coming from the solution on the interface of subdomain 2, \( u_{\Gamma_2} \). The second system solves in subdomain 2 using the Neumann condition coming from subdomain 1, that is \( b_{\Gamma_1} - A_{\Gamma_11}u_1 - A_{\Gamma_1}\Gamma_1u_{\Gamma_1} \), which consists of the residual of the interface node equation of subdomain 1. We have therefore a way to partition the solution of the original problem into the solution of two problems and to obtain the same solution. But for the moment, the problems are not independent; we will now see in next section how to build partitioned strategies to solve the two problems independently, and how to couple them explicitly and implicitly.

2.4 Summary

Figure 5 summarizes the equivalences between the two-subdomain and one-domain formulations described in the previous three sections. For the continuous variational forms, both the extension-based and Dirichlet/Neumann formulations, respectively given by Systems (3) and (4), are equivalent to the one-domain formulation (2). At the discrete level, we only have equivalence for the discrete counterpart of the extension-based formulation. Finally, the two-subdomain algebraic system (8) is equivalent to the one-domain system (7). We will refer to this method as Dirichlet/Residual or Dirichlet/Neumann indifferently, as from now on, we will only consider couplings at the algebraic level. From this formulation, we can divide a family of domain composition methods, that we are now going to study.
Figure 5: Road to coupling using a Dirichlet/Neumann method. (Top): continuous variational form. (Mid): discrete variational form. (Bot): algebraic system.
3 Coupling strategies

In the last section, we concluded that Equation (8) is a good starting point to devise a coupling strategy at the algebraic level. We are now going to present different strategies which enable explicit and implicit Dirichlet/Neumann coupling at the algebraic level, also referred to as Dirichlet/Residual coupling. In a first step, we will consider matching meshes on the interface. At the end of the section, we will eventually introduce strategies to deal with non-matching meshes.

3.1 Explicit coupling

The so-called iteration-by-subdomain methods solve System (8) in a decoupled manner. Let us introduce an iteration superindex \( k \), and assume an initial condition \( u_{1,2}^0 \).

The iteration-by-subdomain method consists in solving the following two systems for \( k = 1, 2, \ldots \) until convergence:

\[
\begin{pmatrix}
A_{11} & A_{12} & 0 & 0 \\
0 & I & 0 & 0 \\
A_{21} & A_{22} & 0 & 0 \\
0 & 0 & A_{21} & A_{22}
\end{pmatrix}
\begin{pmatrix}
u_1^{k+1} \\
u_2^{k+1} \\
u_1^{k+1} \\
u_2^{k+1}
\end{pmatrix}
= 
\begin{pmatrix}
b_1 \\
b_2 \\
b_{\Gamma_1} - A_{\Gamma_1} u_1^{k+m} - A_{\Gamma_1} u_2^{k+m} \\
b_{\Gamma_2}
\end{pmatrix},
\]

(10)

Should this method converges, the solution is the same as the one-domain problem. In last equation, we have introduced index \( m \), which can take the following values:

\[
m = \begin{cases} 
0 \text{ Parallel or Jacobi coupling} \\
1 \text{ Sequential or Gauss-Seidel coupling} 
\end{cases}
\]

In the first case, \( m = 0 \), the method is said to be parallel because both problems can be solved at the same time, as the Dirichlet and residual conditions are taken from the previous iteration. In addition, we note that can rewrite Equation (7) by introducing the following block matrix:

\[
\begin{pmatrix}
A_{11} & A_{12} & 0 & 0 \\
0 & I & 0 & 0 \\
A_{21} & A_{22} & 0 & 0 \\
0 & 0 & A_{21} & A_{22}
\end{pmatrix}
\begin{pmatrix}
u_1 \\
u_2 \\
u_1 \\
u_2
\end{pmatrix}
= 
\begin{pmatrix}
b_1 \\
b_2 \\
b_{\Gamma_1} + b_{\Gamma_2}
\end{pmatrix},
\]

We can easily check that applying a block Jacobi method, we recover (10) for \( m = 0 \). Equivalently, the coupling method with \( m = 1 \) is referred to as sequential or Gauss-Seidel, as the solution of the second block requires the solution of the first block.

The block Gauss-Seidel method usually converges better than the Jacobi method. However, the Jacobi method enables the parallel solution of both problems. The best choice is therefore problem dependent. In addition, under-relaxation may be required to obtain convergence.
To illustrate the importance of the relaxation and the differences between Jacobi and Gauss-Seidel algorithms, let us solve the following manufactured problem:

\[ \mathcal{L}(u) = \mathcal{L}(u_e) \quad \text{in} \quad \Omega = [0, 1] \times [0, 1], \]
\[ u = u_e \quad \text{on} \quad \partial \Omega, \]

with

\[ \mathcal{L}(u) = a \cdot \nabla u - \varepsilon \Delta u, \]
\[ u_e = 2x \times 3y. \]

We are considering linear finite elements, so the numerical solution will coincide exactly with \( u_e \), which belong to the finite element space. We have taken \( \varepsilon = 10^{-2} \) and considered three different advections \( a = 0 \), \( a = [1, 0]^t \) and \( a = [-1, 0]^t \).

We consider a two-subdomain decomposition with \( \Omega_1 = [0, 1/2] \times [0, 1] \) and \( \Omega_2 = [1/2, 1] \times [0, 1] \). We apply a Neumann condition on \( \Omega_1 \) and a relaxed Dirichlet condition on \( \Omega_2 \), that is \( u^{k+1} = \alpha u^{k+m} + (1 - \alpha) u^k \), where \( \alpha \) is the relaxation factor.

Figure 6 compares the number of iterations required to obtain an \( L^2 \) error of \( 10^{-6} \) with

\[ L^2 \text{ error} = \frac{\|u - u_e\|}{\|u_e\|}, \]

for different methods (Jacobi/Gauss-Seidel), different relaxation parameters for the Dirichlet condition, and different advections. We can draw the following (well-known) conclusions:

- The Gauss-Seidel is much more robust than the Jacobi method;
- For the right advection, both methods converge much faster that for the left advection;
- The optimum relaxation factor is problem dependent.

With respect to the second point, the result can be easily explained. In the hyperbolic limit where \( \varepsilon \to 0 \), the matrix coefficients of any node with upstream nodes tend to zero. Therefore, apart from the force term, the algebraic Neumann condition coming from subdomain 2 is also zero. The solution on \( \Omega_1 \) is therefore obtained in one iteration, and the consequent solve in subdomain 2 will also give the exact solution.

The fact that the type of optimum condition at the interface depends on the local character of the equation (inflow, outflow) is well-known. Adaptive domain decomposition methods have been derived for disjoint subdomains to take into account the direction of the advection (flow) on the interfaces. These methods have been studied extensively in the literature \([16, 17, 18, 19, 20, 21]\). In addition, a Robin condition, which consists of a combination of Dirichlet and Neumann conditions can be useful as well to reach convergence faster.
With respect to the third point, methods such as Aitken [22] or Quasi-Newton [23] can be easily implemented as well to accelerate the convergence in a dynamic and automatic way.

### 3.2 Implicit coupling

In order to avoid iterating using Jacobi or Gauss-Seidel methods, the solution consists in implementing an implicit coupling. The idea is very simple and almost identical to substructuring parallelization methods. It is based on the fact that when using iterative solvers, the main operations are matrix-vector products $q = Ap$ and scalar products. The implicit method presented now is thus only valid when iterative solvers are considered. A matrix-vector product applied to the monolithic System (6) gives:

$$
\begin{bmatrix}
q_1 \\
q_2 \\
q_3
\end{bmatrix} =
\begin{bmatrix}
A_{11} & 0 & A_{1\Gamma} \\
0 & A_{22} & A_{2\Gamma} \\
A_{\Gamma 1} & A_{\Gamma 2} & A_{\Gamma \Gamma}^{(1)} + A_{\Gamma \Gamma}^{(2)}
\end{bmatrix}
\begin{bmatrix}
p_1 \\
p_2 \\
p_3
\end{bmatrix}.
$$

Let us define the local matrices, right-hand sides and unknowns for $i = 1, 2$:

$$
A^{(i)} = \begin{bmatrix}
A_{ii} & A_{i\Gamma} \\
A_{\Gamma i} & A_{\Gamma \Gamma}
\end{bmatrix}, \quad b^{(i)} = \begin{bmatrix}
b_i \\
b_\Gamma
\end{bmatrix}, \quad u^{(i)} = \begin{bmatrix}
u_i \\
u_\Gamma
\end{bmatrix}.
$$

Now consider the following uncoupled systems one would obtain by integrating...
independently over both subdomains:
\[ A^{(1)} u^{(1)} = b^{(1)}, \quad A^{(2)} u^{(2)} = b^{(2)}. \] (11)

For these two local systems, we would obtain for the local matrix-vector products the following:

\[ q_1 = A_{11} p_1 + A_{1\Gamma_1} p_{\Gamma_1}, \]
\[ q_{\Gamma_1} = A_{\Gamma_11} p_1 + A_{\Gamma_1\Gamma_1} p_{\Gamma_1}, \]
\[ q_2 = A_{22} p_2 + A_{2\Gamma_2} p_{\Gamma_2}, \]
\[ q_{\Gamma_2} = A_{\Gamma_22} p_2 + A_{\Gamma_2\Gamma_2} p_{\Gamma_2}. \]

After this product, by performing the following operations:

\[ q_{\Gamma_2} \leftarrow q_{\Gamma_2} + q_{\Gamma_1}, \] (12)
\[ q_{\Gamma_1} = q_{\Gamma_2}, \] (13)

we recover the same result as in Equation (11), with \( q_{\Gamma_1} = q_{\Gamma_2} = q_{\Gamma} \). In order to be able to solve the same problem \( Au = b \), one operation remains: the assembly of the right-hand side \( b \). We observe that by setting

\[ b_{\Gamma_2} \leftarrow b_{\Gamma_2} + b_{\Gamma_1}, \] (14)
\[ b_{\Gamma_1} = b_{\Gamma_2}, \] (15)

we obtain \( b_{\Gamma_1} = b_{\Gamma_2} = b_{\Gamma} \). To refer to Equations (14) and (12), we will use the term *array assembly*. The assignement will be referred to *array substitution*.

We have therefore a way to compute *almost* exactly the same iterations of an iterative solver with two subdomains as with one domain, by doing the following:

- When initializing the solver, assemble and substitute the local right-hand sides, as given by Equations (14) and (15), respectively;
- After each matrix-vector product, assemble and substitute the local products, as given by Equations (12) and (13), respectively.

We stated *almost* exactly because there exists a very slight difference. When performing a scalar product, we observe that the contribution of interface nodes are accounted for twice, while in the case of the one-domain solution, it appears only once. One could decide to take into account only one side in order to recover exactly the same results, either the Neumann interface (the one which assembles) or the Dirichlet interface (the one which substitutes).
3.3 Non-matching meshes

3.3.1 Transmission of data

The previous subsections dealt with matching grids, for which the node-to-node correspondence between the subdomains enable an easy access to the coefficients of the vectors. The treatment of non-matching grids complicates greatly the implementation of domain composition methods, in terms of data structures and parallelization issues. We refer the reader to [24] for a good review of some coupling techniques.

For the sake of clarity of the physical interpretation, we will follow the explicit formulation (6) rather than the implicit one for interpretation reasons: in fact, the explicit formulation states clearly that on subdomain 1 is imposed a Dirichlet condition, and on subdomain 2 is imposed a Neumann condition. Let us introduce some transmission matrices $T_D$ and $T_N$ which represent the Dirichlet and Neumann conditions, respectively. Equation (6) reads:

\[
\begin{pmatrix}
A_{11} & A_{1\Gamma_1} \\
0 & I
\end{pmatrix}
\begin{pmatrix}
u_1 \\
u_{\Gamma_1}
\end{pmatrix} =
\begin{pmatrix}
b_1 \\
T_D u_{\Gamma_2}
\end{pmatrix},
\]

(16)

which corresponds to a Dirichlet/Neumann coupling with the following Dirichlet and Neumann conditions:

\[
u_{\Gamma_1} = T_D u_{\Gamma_2}, \quad r_{\Gamma_2} = T_N r_{\Gamma_1},
\]

and where the residual $r_{\Gamma_1}$ is defined in Equation (9).

We therefore need to define both transmission matrices, $T_D$ and $T_N$. In the literature, two main methods have been proposed, as illustrated in Figure 7:

- Interpolation schemes;
- Projection schemes.

In the following, we are going to consider a generic interface unknown, noted $u$ at the continuous level, $u_i$ the nodal values at the discrete level, and $u$ at the algebraic level. In the previous context, $u$ can therefore be the interface unknown or the residual. We will consider a generic transmission matrix $T$ as well, which stands for either $T^D$ or $T^N$.

We will refer to the target as the set of entities where the transmission condition (Dirichlet or Neumann) is prescribed and to the source as the set of entities where this condition is transmitted from. We will use the subscript $t$ and $s$ to identify the quantities referring to the target and the source, respectively. In that sense, $\Gamma_t$ is the target surface (also referred to as wet surface in the literature) and $\Gamma_s$ is the source...
Figure 7: Scheme to transfer data from the source $\Gamma_s$ to the target $\Gamma_t$.

surface. In the previous two-subdomain context, $t = 1$ or 2, and $s = 2$ or 1. As an example, as far as the Dirichlet condition in Equation (16) is concerned, the target surface is the interface of subdomain 1 $\Gamma_1$, while the source solution is $u_{\Gamma_2}$. The transmission conditions can therefore be written:

$$u_t = Tu_s.$$  \hspace{1cm} (17)

Finally, let $n_t$ and $n_s$ be the numbers of target and source nodes, respectively.

**Interpolation schemes.** Let us consider a target node $i$ on $\Gamma_t$ with coordinates $x_i$. We want to obtain a nodal value $u_{t,i}$ on the target from the values known on the source, $u_{s,j}$. The simplest interpolation scheme considers a linear interpolation [25], by first identifying the host boundary element of the target node in the source boundary so that $u_{s,i} = \sum_{j=1}^{n_s} N_{s,j}(x_i) u_{s,j}$, and therefore

$$T_{ij} = N_{s,j}(x_i).$$

High-order interpolation schemes consider rather a cloud of nodes. Let us mention methods such as radial basis functions [26, 24] or Krigging [27].

**Projection schemes.** On the other hand, a projection method imposes continuity in a weak sense. The projection method consists in minimizing $\|u_s - u_t\|^2_{L^2(\Gamma_t)}$, that is in solving the following system:

$$\int_{\Gamma_t} u_t v_t \, d\Gamma = \int_{\Gamma_t} u_s v_t \, d\Gamma \quad \forall \, v_t \in V_t,$$  \hspace{1cm} (18)
where \( V_t \) is the space of test functions on \( \Gamma_t \). See Figure 8.

Let \( N_{t,i} \) be the shape function of node \( i \) on the target. On the one hand, the left-hand side only involves quantities on the target. On the other hand, the right-hand side involves the solution on the source as well. When computing this integral numerically, the source unknown \( u_s \) thus must be interpolated at the integration point of the target. By introducing the nodal unknowns on the target \( u_{t,j} \), we obtain:

\[
\int_{\Gamma_t} \left( \sum_{j=1}^{n_t} u_{t,j} N_{t,j} \right) N_{t,i} \, d\Gamma = \int_{\Gamma_t} u_s N_{t,i} \, d\Gamma \quad \text{for} \quad i = 1, \ldots, n_t,
\]

and therefore we end up with the following system:

\[
M_t u_t = s, \quad (19)
\]

with

\[
M_{t,ij} = \int_{\Gamma_t} N_{t,j} N_{t,i} \, d\Gamma, \quad s_i = \int_{\Gamma_t} u_s N_{t,i} \, d\Gamma.
\]

Equation (19) involves the target boundary mass matrix. By choosing a close integration rule or by lumping the mass matrix, we obtain a diagonal mass matrix \((M_{t,ij} = \delta_{ij}M_{t,i})\). As far as the right-hand side is concerned, let us start by introducing some notations (see Figure 8). Let \( b = 1, \ldots, n_b \) be the boundary elements on the target. On each boundary, we apply an integration rule with \( n_g \) integration points of coordinates \( x_{g,b} \) with weight \( w_g \) for \( g = 1, \ldots, n_g \). This integration rule is chosen to accurately compute the integral. We have

\[
s_i = \sum_{b=1}^{n_b} \sum_{g=1}^{n_g} \left( \sum_{j=1}^{n_t} N_{s,j}(x_{g,b})u_{s,j} \right) N_{t,i}w_g|J(x_{g,b})|_b,
\]
where \( |J(x_{g,b})|_b \) is the Jacobian of the boundary \( b \) computed at the integration point. Therefore, according to equation (17), we have

\[
L^2\text{-projection: } T_{ij} = \frac{1}{M_{t,i}} \sum_{b=1}^{n_b} \sum_{g=1}^{n_g} N_{s,j}(x_{g,b}) N_{t,i}(x_{g,b}) w_g |J(x_{g,b})|_b. \tag{20}
\]

Let us mention that we have considered the simple case where the target and source surfaces coincide. If this is not the case then we should consider projections of the target nodes (interpolation) or target Gauss points (projection) onto the source surface \( \Gamma_s \) (see for example [24]).

### 3.3.2 Conservation

In the context of DCM, conservation refers to the exact transmission, from one sub-domain to another, of either local or global quantities. On the one hand, the previous interpolation schemes are local operators and do not, in general, conserve any specific quantity, at least globally. On the other hand, by choosing properly the integral rule used to compute the right hand-side, we observe that taking \( v_t = 1 \) in Equation (18), we automatically conserve the integral of the unknown, and therefore the average value of it. This means that a constant is perfectly transmitted. In the case of fluid-structure interactions, if the target variable \( u \) is the fluid displacement, it means that a rigid-body translation is conserved. If the target variable is the temperature, it means that a constant increase in temperature is conserved as well.

... But what else can we conserve?

**Global flux.** Some global quantities can be automatically conserved depending on the selected transmission matrices \( T^D \) and \( T^N \). In [25, 24], it is shown that for Fluid-Structure interactions (FSI), if \( \Omega_1 \) is the fluid subdomain and \( \Omega_2 \) the solid subdomain, then, by choosing for the Neumann transmission matrix the transpose of the Dirichlet one, i.e.

\[
T^N = (T^D)^t, \tag{21}
\]

and such that \( \sum_j T^D_{ij} = 1 \text{ } \forall i \), then the method satisfies the following properties:

- Virtual work acting on the fluid is conserved;
- Rigid body translations of the solid are exactly recovered on the fluid;
- The total force on the interface is conserved, that is \( \sum_i r_{\Gamma_2,i} = \sum_i r_{\Gamma_1,i} \).

For example, linear interpolation and projection (if RHS is carefully computed) for \( T^D \) both satisfy \( \sum_j T^D_{ij} = 1 \text{ } \forall i \), and the previous conservation properties are satisfied, and in particular, the total force imposed from the fluid on the solid. Similarly, if the governing equation is the temperature equation in both subdomains, then the
conserved variable is the total heat flux. In the case of the Navier-Stokes equations, the conserved variable consists of the global traction. See also [28], where force conservation is imposed via a projection method for FSI.

**Conservation via constraint.** Another methodology to conserve any kind of global quantity [29] consists in solving a constrained minimization problem of the form

\[
\begin{align*}
\text{minimize} & \quad \|u_t - Tu_s\|^2, \\
\text{under the constraint} & \quad c^t u_t = c.
\end{align*}
\]  

(22)

The idea of solving this problem is to look for the nearest solution to Equation (17) under a scalar constraint represented by Equation (22)_2, where \(c\) is a vector and \(c\) a scalar. For example, if one wants to conserve the average of the variable, that is \(\int_{\Gamma_t} u_t \, d\Gamma = \int_{\Gamma_t} u_s \, d\Gamma\), one would choose

\[
\text{Average value conservation:} \quad c_i = \int_{\Gamma_t} N_{t,i} \, d\Gamma, \quad c = \int_{\Gamma_t} u_s \, d\Gamma.
\]

As another example, this scheme was used in the context of the Navier-Stokes equations in [29] in order to conserve a zero mass across the interface when interpolating the velocity.

The system (22) is solved by introducing the Lagrange multiplier \(\lambda\) of the constraint. The Lagrangian is given by

\[
L(u_t, \lambda) = \|u_t - Tu_s\|^2 - \lambda(c^t u_t - c).
\]

Searching for the optimal point of the Lagrangian, and defining \(\mu = \lambda/2\), leads to solving the following system:

\[
\begin{bmatrix}
I & -c \\
& & \\
c^t & 0 & & \\
& & & 0
\end{bmatrix}
\begin{bmatrix}
u_t \\
& \\
\mu
\end{bmatrix}
= \begin{bmatrix}
Tu_s \\
& \\
c
\end{bmatrix}.
\]

(23)

By performing the operation \([c^t \cdot (23)_1 - (23)_2]\), we obtain the multiplier \(\mu\). Then, by substituting this value into the first equation, we finally end up with:

\[
u_t = Tu_s + \left(\frac{c - c^t Tu_s}{c^t c}\right) c.
\]

The first term is the unconstrained transmission operation \(u_t = Tu_s\), while the second term is the correction with respect to this operation. The term between parenthesis is a scalar, the correction. Note that if the transmitted solution \(Tu_s\) already satisfies the constraint, the correction is zero, which makes the scheme consistent. Note finally that the correction is distributed in \(u_t\) according to vector \(c\), as given by the last term of the equation.
Conservative residual projection. We already observed that the residual of the governing equation \( r_t \) (Equation (9)) corresponds to (although not strictly equivalent to) the assembly of the natural condition of the variational form. We now naturally introduce the residual flux, denoted as \( \tilde{r} \). The nodal residual on the target is thus assembled from this residual flux as:

\[
\begin{align*}
\mathbf{r}_{t,i} &= \int_{\Gamma_t} \tilde{r}_s N_{t,i} \, d\Gamma,
\end{align*}
\]

(24)

as illustrated in Figure 9.

The question is how to compute the residual flux \( \tilde{r}_s \). The residual flux is the residual per unit surface. What we propose is to use the lumped source boundary mass matrix \( M_s \) (which is diagonal) to scale the residual to obtain the residual flux as:

\[
\begin{align*}
\tilde{r}_{s,i} &= \frac{r_{s,i}}{M_{s,i}},
\end{align*}
\]

(25)

where \( M_{s,i} \) is the coefficient of the diagonal mass matrix of source node \( i \). With this choice, the target residual is computed as:

\[
\begin{align*}
\mathbf{r}_{t,i} &= \int_{\Gamma_t} \left( \sum_{j=1}^{n_g} \frac{r_{s,j}}{M_{s,j}} N_{s,j} \right) N_{t,i} \, d\Gamma.
\end{align*}
\]

(26)

Taking the same nomenclature as in the description of the projection method, we therefore have that the coefficients of the transmission matrix \( \mathbf{T} \) are:

\[
\begin{align*}
T_{ij} &= \frac{1}{M_{s,j}} \sum_{b=1}^{n_b} \sum_{g=1}^{n_g} N_{s,j}(\mathbf{x}_{g,b}) N_{t,i}(\mathbf{x}_{g,b}) w_g |J(\mathbf{x}_{g,b})|_b.
\end{align*}
\]

We observe that the difference with the projection transmission matrix given by Equation (20) is that in the present case, the mass matrix involves the source mass matrix.
of node $j$ instead of the target mass matrix of node $i$.

What can we say about the scheme (24) together with Equation (25)? Let us compute the total residual (total heat power, total force, etc.):

$$\text{Total residual} = \sum_{i=1}^{n_t} r_{t,i} = \int_{\Gamma_t} \tilde{r}_s \left( \sum_{i=1}^{n_t} N_{t,i} \right) \, d\Gamma,$$

$$= \sum_{j=1}^{n_s} \tilde{r}_{s,j} \int_{\Gamma_t} N_{s,j} \, d\Gamma, \quad \left( \sum_{i=1}^{n_t} N_{t,i} = 1 \right)$$

$$= \sum_{j=1}^{n_s} \tilde{r}_{s,j} M_{s,j} \, d\Gamma, \quad \text{(lumped mass matrix)}$$

$$= \sum_{j=1}^{n_s} r_{s,j} \, d\Gamma. \quad \text{(Equation (25))}$$

Therefore, as long as the right-hand side is computed with sufficient accuracy, it conserves the total residual, that is the total heat in the case of the temperature equation and the total force in the case of Navier-Stokes and structure equations.

### 3.3.3 Resulting algorithm

The selected interpolation scheme will influence not only the accuracy of the result but also the cost of the coupling. This last point is especially true in the case of the implicit method, where the interface conditions are imposed after each matrix-vector product. This point will be raised in the next Section concerning implementation issues. To finish with the coupling strategies, we will now raise an important point concerning symmetry.

Let us take a look at the resulting algorithm defined by Equation (16). To proceed, we eliminate $u_{1_{\Gamma_1}}$ by using the second equation. After some algebraic manipulation, we end up with:

$$\begin{pmatrix}
A_{11} & 0 & A_{1\Gamma_1} T^D \\
0 & A_{22} & A_{2\Gamma_2} \\
T^N A_{\Gamma_1} & A_{\Gamma_2} & T^N A_{\Gamma_1 \Gamma_1} T^D
\end{pmatrix}
\begin{pmatrix}
u_1 \\
u_2 \\
u_{\Gamma_2}
\end{pmatrix} =
\begin{pmatrix}
b_1 \\
b_2 \\
b_{\Gamma_2} + T^N b_{\Gamma_1}
\end{pmatrix}$$

This system can be solved both explicitly or implicitly, as described in Sections 3.1 and 3.2, respectively. We will now examine the case of a symmetric problem, that is

$$A_{11} = A_{11}^t, \quad A_{22} = A_{22}^t, \quad A_{\Gamma_1 \Gamma_1} = A_{\Gamma_1 \Gamma_1}^t, \quad A_{\Gamma_2 \Gamma_2} = A_{\Gamma_2 \Gamma_2}^t, \quad A_{\Gamma_1} = A_{\Gamma_1}^t, \quad A_{\Gamma_2} = A_{\Gamma_2}^t.$$
that leads to a symmetric problem as well. We observe that, in general, the coupled system is not symmetric, even though the original independent systems are. If the coupled problems are solved explicitly, therefore independently, the lack of symmetry of the coupled system is not an issue. The problem arises for implicit coupling. These point will prevent us from using efficient solvers specially designed for symmetric matrices (CG, deflated CG, etc.). So what can we do to fix this?

A simple solution consists in selecting the transmission matrices such that $T^D = T$ and $T^N = T'$, such as in Equation (21), where this choice enables one to conserve the total force on the interface. We can easily check that we end up with a symmetric system and therefore the last choice for the transmission matrices preserves symmetry.

### 3.3.4 Going to implicit

We have explained in this section how to implement an explicit Dirichlet/Neumann method at the algebraic level. For the implicit method applied to matching meshes, the method was described in Section (3.2). Therefore, by considering the transmission matrices, Equations (12) and (13) become:

\begin{align}
q_{\Gamma_2} &\leftarrow q_{\Gamma_2} + T^N q_{\Gamma_1}, \quad (27) \\
q_{\Gamma_1} &\leftarrow T^D q_{\Gamma_2}. \quad (28)
\end{align}
This section is devoted to implementation issues, with special emphasis on the parallelization aspects, in a distributed memory context. The DCM presented in the previous sections was implemented in Alya, a high performance computational mechanics code developed at BSC-CNS, extensively described in [30]. One important point of interest here is the parallelization strategy. The mesh is partitioned using METIS [31], and the coupling of the partitions is mainly present at the iterative solvers and involves:

- Global communications to compute scalar products;
- Point-to-point communications to assemble the results of each matrix-vector products.

Additional communications may be required to assemble complex preconditioners, but are not of interest in the following.

Let us first introduce some notations, illustrated in Figure 10. The target surface is the surface where a transmission condition, either Dirichlet or Neumann, is imposed. The source surface is the surface where this transmission condition is computed from. On the target surface we identify three more entities: the target boundary element, the target nodes and the target points. The target points can be nodes or Gauss points, whether the transmission condition is carried out through interpolation or through projection. The target nodes are the nodes belonging to target boundaries, where the transmission condition is assembled after each matrix-vector product, as both Dirichlet and Neumann algebraic transmission conditions are imposed node-wise.

There are two main possible implementations. The first one consists in constructing explicitly the transmission matrix; the second one consists in assembling the transmission condition on the fly. Here we will describe this last option, although the first one has for sure some advantages over the last one. Nevertheless, what we are going to discuss can be applied to both implementations.
The DCM method can be divided into two main steps: a pre-process to determine
the connectivities between the target and the source, and an assembly process con-
sisting in exchanging some data from the source to the target and assembling it on
the target, after each matrix-vector product. We will now discuss both aspects, by
considering a two-subdomain coupling for the sake of clarity.

4.1 Pre-process

In a distributed memory context, the mesh is divided into \( n_p \) partitions. Figure 11
shows an example of a DCM to couple two subdomains, partitioned into seven parti-
tions in total. In this example, we depicted a particular target point of subdomain 1.
This target point is a node of subdomain 1 if an interpolation scheme is considered
or an integration point if a projection scheme is selected instead. The first question
to ask is: which partition can interpolate the values of a particular target point? As
we are working in a distributed memory environment, the partitions have a-priori no
direct access to the meshes of other partitions. Therefore, all queries must be carried
out through MPI. To limit the information traffic through the network, one possibi-
lity consists in using the bounding boxes of the partitions to reduce the number of queries.
We can imagine faster algorithms as well as bin or oct-trees to reduce the number of
queries further more. For the moment, let us stick to the simplest option consisting in
using only the bounding boxes. For each target point \( x_t \) of each parallel partition of
the mesh, we apply the procedure described in Algorithm 1.

After carrying out this algorithm, each partition with target points knows the list
of target points hosted by each partition \( p \). And these partitions know they host these
Algorithm 1 Find the host partition $p_{\text{host}}$ of a target point $x_t$

Input: target point coordinate $x_t$
Output: host partition $p_{\text{host}}$ of $x_t$

1. $\text{MPI}_\text{SEND}(x_t)$ to partitions $p$’s whenever $x_t$ is in their bounding box.
2. Partition $p$ computes projection of $x_t$ onto their source boundary. $d_p$ is the distance between $x_t$ and the projection point.
3. $\text{MPI}_\text{RECV}(d_p)$ from the $p$’s.
4. Identify the partition $p_{\text{host}}$ with the minimum distance.
5. $\text{MPI}_\text{SEND}(p_{\text{host}})$ to the $p$’s partitions to inform them of the result of the minimization. Partition $p = p_{\text{host}}$ will be in charge of target point $x_t$.

target points. They are therefore able to construct a scheduling to exchange data on these target points, and the assembly can be carried out.

4.2 Assembly

In order to solve the problem implicitly, we have to carry out the operations given by Equations (27) and (28), consecutively. Therefore, the Neumann condition comes first. For this, we use the conservative residual projection algorithm presented in Section (3.3.2). According to Equation (26), and substituting the residual $r$ by the result of the matrix-vector product $q$, the target has to assemble the following equation:

$$q_{t,i} = \int_{\Gamma_t} \left( \sum_{j=1}^{n_s} (q_{s,j}/M_{s,j})N_{s,j} \right) N_{t,i} \, d\Gamma.$$  \hspace{1cm} (29)

The partitions of the source that host the target points (that is the Gauss points used to compute this integral) send to the target the following interpolation at the target point $x = x_t$:

**Computed at the source:** $\tilde{q}(x) = \sum_{j=1}^{n_s} (q_{s,j}/M_{s,j})N_{s,j}(x)$.

Once all these values $\tilde{q}(x)$ have been received by the target partitions, these partitions can proceed by integrating Equation (29) on their target boundaries, that is:

$$q_{t,i} = \int_{\Gamma_t} \tilde{q}(x)N_{t,i} \, d\Gamma.$$
5 Results

5.1 2D Cavity: flow

We consider a cavity flow in the domain $[0, 1] \times [0, 1]$, with unit velocity on top. The fluid properties are chosen such that the Reynolds number is relatively high $Re = 5000$. The meshes considered are shown in Figure 12. Two meshes for the one-domain simulation and two meshes for the Dirichlet/Neumann method. The large subdomain is applied a Neumann condition and the small one a Dirichlet condition.

All the details concerning the flow solver can be found in [32]. Let us only mention one point of interest of this solver for this work. The solution procedure consists in solving for the pressure Schur complement. Therefore, two solves for the momentum equation and one solve for the pressure Schur complement are needed at each Newton-Raphson iteration. The Dirichlet/Neumann is therefore applied to both systems. As far as the Neumann condition is concerned, a sufficient number of integration points have been used for the residual computation. The Dirichlet condition is interpolated linearly.

Figure 13 shows the contours of velocity and pressure obtained on Mesh4. Figure 14 shows the vertical velocity on a horizontal cut at $y = 0.5$, and the horizontal velocity on a vertical cut at $x = 0.9$, for the four simulations (Mesh1, Mesh2, Mesh3 and Mesh4). The one-domain solution on Mesh2 should be seen as the target solution, and we expect that the D/N on Mesh4 give a better solution than the one-domain solution on Mesh1, as the mesh is refined on the bottom right corner. As far as the
horizontal cut is concerned, we cannot observe a substantial difference. Nevertheless, we see that the solution is quite different for the four simulations on the cut going through the recirculation zone. This recirculation is better captured by the D/N method than with the coarse mesh. We conclude that the D/N can be used as a local refinement technique.

5.2 3D Cavity: parallelization

With this second example we wish to study the parallelization aspects of the method. In this case, the interface of subdomain 1 with subdomain 2 comes from a hole cutting procedure, specific to the Chimera method, as described in [5]. The geometry is shown

Figure 14: Cavity flow: results. (Top) Vertical velocity along horizontal cut at $y = 0.5$. (Bot.) Horizontal velocity at vertical cut at $x = 0.9$. 
in Figure 15. In the Chimera jargon, subdomain 1 is referred to as the background, and imposed a Neumann condition; subdomain 2 is referred to as the patch, and is imposed a Dirichlet condition. The right part of the Figure 15 shows the partitioning into 63 partitions of the mesh (1.6M elements), carried out with METIS. The patch is the finest subdomain divided into 49 partitions while subdomain 1 is divided into 14 partitions. Note that no partition has more than one subdomain.

Figure 16 shows, for both subdomains, the following results: on the left part of the figure, the number of target points tested in Algorithm 1 using the bounding box criterion; on the right part, the number of points really hosted by each partition. We observe that a lot of communication is necessary to arrive at the final list of hosted points. For the Neumann coupling, 40% of the points tested are eventually hosted. In the case of the Dirichlet condition, only 26% are hosted. In the case of moving subdomains, this algorithm should be applied at each time step to recouple them. It is therefore important that it is sufficiently fast to limit its impact on the overall simulation. To give an order of magnitude, we have computed the complete time to prepare the couplings, where most of it is spent in Algorithm 1. This time is 2.54 s. In comparison, we have solved a PDE using the BICGSTAB [33] iterative solver, with 240 iterations. The time for this solver is 1.41 s. We therefore conclude that the algorithm should be optimized. For example, this can be done by limiting the number of candidate partitions to which the target points are sent (step 1 of the algorithm).

Finally, Figure 17 shows the trace of the BICGSTAB, obtained with Paraver, a trace visualizer developed at BSC-CNS, Spain [34]. In these graphs, the $x$-axis represents CPU time and the $y$-axis the partition number. The colors show the different status of each CPU. Blue: CPU working. Red: idle CPU. Orange: global communication. Yellow arrows: point-to-point communication. On the left part of the figure, we can
Figure 16: 3D Cavity. Number of tested points and hosted points for the Dirichlet and Neumann couplings.

Figure 17: 3D Cavity. Trace of the BICGSTAB algorithm. Blue: CPU working. Red: idle CPU. Orange: global communication. Yellow arrows: point-to-point communication. (Left) Ten iterations. (Right) One single matrix-vector-product.
observe the initialization phase and 10 iterations of the algebraic solver. At each iteration of the solver, two matrix-vector products are carried out. The intra-subdomain communications (parallelization of the product) and inter-subdomain communications (subdomain coupling) are depicted by the yellow arrows. On the right part of the figure, we have the trace of one single matrix-vector product. This product is flanked by two scalar-products depicted in orange. We can observe two important things. First of all, and as mentioned earlier, METIS has partitioned the two subdomains independently. Although not shown in this figure, the partitions of subdomain 1, meshed with Tetrahedra (4 integration points) are well-balanced with the partitions of subdomain 2, meshed with Hexahedra (8 integration points). However, a Tetrahedra mesh has in average 6 times less nodes than an hexahedra mesh. We therefore observe a strong imbalance in the matrix-vector product (in blue) between the subdomains. The second important point is the relative weight between the intra-subdomain communications and the inter-subdomain communications. On the trace we can distinguish clearly the first coupling, the Neumann condition, and the second one, the Dirichlet condition. We observe that both have more or less the same duration and their cumulated duration is of the same order the intra-subdomain communications between the partitions, making this acceptable for the simulation.
6 Conclusions

We have presented a parallel domain composition method for computational mechanics problems. The coupling is implicit and implemented at the algebraic level. It is therefore equation independent and implies modifying only the matrix-vector product. The formulation has been tested on the Navier-Stokes equations and its parallel performance illustrated through the iterations of the BICGSTAB iterative solver. We are presently working to the extension of the method to overlapping subdomains in order to devise an implicit Chimera method.
References


