Archives of Computational Methods in Engineering manuscript No.
(will be inserted by the editor)

Domain decomposition methods for domain composition purpose: Chimera, overset, gluing and sliding mesh methods

G. Houzeaux, J.C. Cajas, M. Discacciati, B. Eguzkitza, A. Gargallo-Peiró, M. Rivero and M. Vázquez

Received: date / Accepted: date

Abstract Domain composition methods (DCM) consist in obtaining a solution to a problem, from the formulations of the same problem expressed on various subdomains. These methods have therefore the opposite objective of domain decomposition methods (DDM). Indeed, in contrast to DCM, these last techniques are usually applied to matching meshes as their purpose consists mainly in distributing the work in parallel environments. 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.

DCM are powerful techniques that can be used for different purposes: to simplify the meshing of a complex geometry by decomposing it into different meshable pieces; to perform local refinement to adapt to local mesh requirements; to treat subdomains in relative motion (Chimera, sliding mesh); to solve multiphysics or multiscale problems, 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, equation-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 equation-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

G. Houzeaux, J.C. Cajas, B. Eguzkitza, A. Gargalló, M. Rivero and M. Vázquez
Barcelona Supercomputing Center (BSC-CNS), Edificio NEXUS I, Campus Nord UPC, Gran Capitán 2–4, 08034, Barcelona, Spain
E-mail: guillaume.houzeaux@bsc.es
M. Discacciati
Department of Mathematical Sciences, Loughborough University, Epinal Way, Loughborough, Leicestershire, LE11 3TU, UK
E-mail: m.discacciati@lboro.ac.uk

techniques can be formulated for the strong formulation at the continuous level, for the weak formulation either at the continuous or at the 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 can be implemented in many different ways. 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 introduces mesh-based, equation-based and algebraic-based DCM. It however focusses on algebraic-based domain composition methods, which have many advantages with respect to the others: they are relatively problem independent; their implicit implementation can be hidden in the iterative solver operations, which enables one to avoid intensive code rewriting; they can be implemented in a multi-code environment.

Contents

1 Introduction

Domain composition methods

Domain Composition Methods (DCM) consist in obtaining a solution to an equation from the formulations of the same equation expressed on various subdomains $i = 1, 2, \ldots$. These methods have therefore the opposite objective of Domain Decomposition Methods (DDM), which usually aim at parallelizing the solution process by decomposing the computational domain, see Figure 1. The mathematical tools employed in DCM and DDM are nevertheless similar, as after partitioning the computational domain, DDM have to recompose the solution, just like DCM. However, their implementations mainly differ for four reasons: (1) the sizes of the local problems in DDM are much smaller than in DCM. Thus, while in DDM one can afford to invert exactly the local or subdomain matrices (denoted by $\mathbf{A}_{ii}$), DCM can usually not; (2) DDM are usually

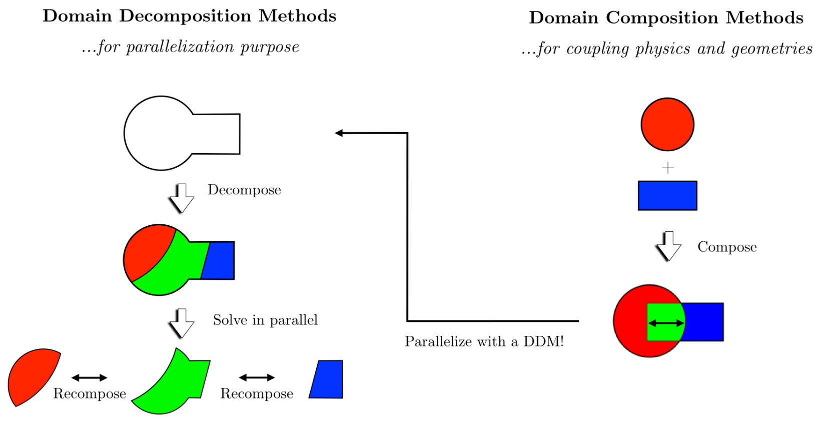

Domain Decomposition Methods
...for parallelization purpose
Domain Composition Methods
...for coupling physics and geometries
Decompose
Solve in parallel
Recompose
Recompose
+
Compose
Parallelize with a DDM!

Fig. 1: Domain decomposition mehtods (DDM) *vs* Domain Composition Methods (DCM).

used for parallelization purposes and are therefore applied to matching meshes (the case of non-matching meshes is also considered in classical DD books, such as the one by Quarteroni and Valli [72], and the corresponding DDM are called *non-conforming*) (3) in a general context (multi-code coupling), the equation for the interface unknowns may hardly be explicitly solved, unlike the case of DDM; (4) In DCM few subdomains with respect to DDM are coupled and thus no coarse solver is required.

In the following, we will use the term DDM or DCM depending on the objective of the method, parallelization or coupling, respectively. However, we will employ the term DDM when referring to the mathematical techniques per se. DCM embrace the different appellations used in the literature like Chimera, overset, gluing mesh, sliding mesh, etc. It should be noted that multi-physics coupling can be achieved using very similar techniques to that of DCM. Although we will not explicitly treat it in this paper, we will mention multi-physics examples throughout the paper.

We refer the reader to the books [77,72,80] for an extensive review on domain decomposition methods.

DCM have many practical applications: 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 easily meshable pieces; to perform local refinement to adapt to local solution requirements; to couple 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. It is nevertheless remarkable that these applications could all be solved by a remeshing of the computational domain, maybe at each time step in the case of moving subdomains. However, remeshing is a very complex methodology in a parallel context, and may require frequent load rebalancing. This explains why DCM are still very popular and useful. DCM can also be applied to solve multiphysics problems [72,30], like fluid-structure interactions, incompressible-compressible flow couplings, conjugate heat transfer, etc.

The main objective of this work is to be able to deal with the majority of the aforementioned applications in a practical and robust way. Several DCM strategies have been proposed in the literature, divided mainly into three different families. These families are referred to herein as *mesh-based*, *equation-based* and *algebraic-based*. The three of them will be treated, although more emphasis will be put on the algebraic-based techniques, as it is probably the most general and practical way of coupling subdomains on parallel supercomputers. In addition we will consider both explicit and implicit schemes. On the one hand, explicit schemes or substructuring approaches consider the solution of the physical problem in each subdomain alternatively. On the other hand, implicit schemes directly include the coupling in the algebraic system (monolithic approach), directly in the matrix or via modification of matrix-vector products in iterative solvers (current work). Eventually we will consider disjoint and overlapping subdomains.

To summarize, we will treat the mesh-based, equation-based and algebraic-based formulations.Both explicit and implicit couplings will be considered, on disjoint as well as on overlapping subdomains. Although we will try to be as exhaustive as possible in the description of the different DCM present in the literature, we will nevertheless concentrate on methods that are easily implementable and efficient in a distributed memory context. This will be achieved by proposing techniques to couple subdomains in a multi-code environment, using existing codes, and by minimizing the implementation impact on these codes. Figure 2 illustrates the computational context. Different or identical physics are solved in parallel, using different or identical codes, and are coupled explicitly or implicitly using MPI. One application example is the simulation of a stirring reactor, showed in Figure 3. Implicit coupling is used to couple the fixed and rotating parts of the thermo-fluid domain. Temperature coupling is achieved explicitly by coupling the thermo-fluid code to another code specifically dedicated to heat transfer.

Mesh-, equation- and algebraic-based methods

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 [62]; a mesh merging technique to create a conforming mesh coming from two

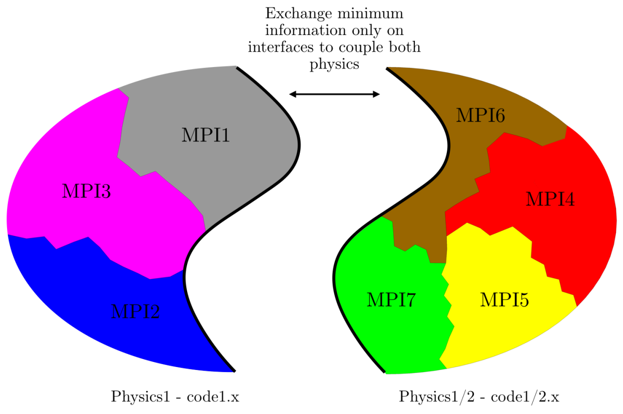

Exchange minimum information only on interfaces to couple both physics
MPI1
MPI3
MPI2
MPI6
MPI4
MPI7
MPI5
Physics1 - code1.x
Physics1/2 - code1/2.x

Fig. 2: Computational context.

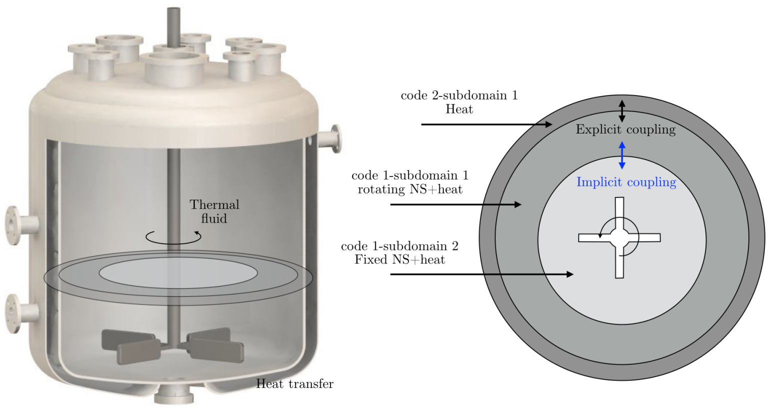


Fig. 3: Example of application if implicit and explicit coupling. Left figure from Ashe, R. (2007): https://en.wikipedia.org/wiki/Chemical_reactor.

independent meshes by merging the meshes from their intersection [64, 21]; the Shear-Slip Mesh Update Method that reconnects sliding meshes with a conforming mesh in the gap [6]; the HERMESH method that joins meshes by creating an overlapping layer of elements (extensions) from one subdomain to the other [52].

Equation-based DCM consider the coupling at the equation level. Let us mention iteration-by-subdomain methods based on transmission conditions

[72], mesh free interpolations [8,53], and methods based on constraint impositions like the mortar element method [9].

Algebraic-based methods work directly on the algebraic system, by imposing the continuity of the system unknown and the residual of the equation [48]. They have a close relation with equation-based techniques.

We will present in this paper the three families of DCM. We will put more emphasis on algebraic-based methods, which have the advantage to be problem independent, and which can be relatively easily implemented in parallel, in a multi-code environment, and which can be made implicit as well.

Explicit vs *implicit*

On the one hand, the coupling of subdomains can quite easily be treated explicitly. In this case, the solutions in the different subdomains are obtained independently and the coupling is achieved through transmission conditions. These conditions can be implemented as boundary conditions or at the algebraic level. These method are referred to as iteration-by-subdomain in the literature. The main drawback is that, by introducing a coupling loop, their convergence may be very poor, and acceleration techniques like under-relaxation, Aitken or Quasi-Newton are usually mandatory to obtain a sufficiently robust scheme. When treating the algebraic-based methods, we will express the coupling as an interface problem. The classical Dirichlet/Neumann will be interpreted as a preconditioned Richardson iteration for this interface problem. We will then propose simple ways of accelerating this problem in a multi-code context. Let us note than *iterative coupling* is more appropriate than *explicit coupling* as, literally, the interface problem is solved with iterative solvers.

On the other hand, implicit methods require a bit more of intervention in the code. However, they enable to completely avoid the problem of convergence as the coupling will be included directly in the algebraic systems, as an add-on to the matrix-vector products involved in classical iterative solvers used for the subdomains algebraic systems. In the case of iterative solvers, implicit coupling may thus be referred to as *direct coupling* or *monolithic coupling* approach.

Disjoint vs *overlapping*

DCM methods can be applied to disjoint and overlapping subdomains. Mesh-based strategies can even accept a gap between the subdomains (e.g. HERMESH method), or explicitly create this gap and then build a conforming gluing mesh between the subdomains (e.g. Shear-Slip Mesh Update Method). The main applications for disjoint subdomains are the coupling *a la lego* of submeshes, or the coupling of rotating and fixed parts like chemical reactors, gas turbines and wind turbines. Disjoint subdomains are also encountered in multi-physics contexts (fluid-structure interaction, filtration, etc.).

The main applications for overlapping methods are the treatments of arbitrary motions of objects, the optimization of the placements of objects, the gluing of overlapping geometries. The main technique to treat these cases is the *Chimera method*, also referred to as *overset meshes/grids method*. The Chimera method has been historically presented as a separate technique for coupling overset meshes. This technique usually considers a background mesh, which covers the complete computational domain, and then patches meshes attached to the moving objects. The main objectives of the Chimera method is to avoid remeshing, by substituting this task by some DCM techniques. The first step of a Chimera method is to create a hole in the background mesh in order to define interfaces between the background and the patch meshes. Then, the solutions on the background and the patches are coupled using a DCM method. Therefore: Chimera = hole cutting + DCM for overlapping meshes.

Content of the paper

The paper will treat successively the mesh-based, equation-based, and algebraic-based techniques in Sections 2, 3 and 4, respectively. The case of algebraic-based techniques will be treated thoroughly, including some implementation aspects.

Throughout the paper we will use many subscripts and superscripts to denote subdomains, nodes, blocks and iterations. Table 1 shows the nomenclature we have adopted.

Symbol	Description
u_i	Continuous variable in subdomain i
$u_{i,h}$	Discrete variable in subdomain i
$\mathbf{u}_i$	Block (i) of a block vector $\mathbf{u}$
$u_{i,n}$	Coefficient n of vector $\mathbf{u}_i$
$\mathbf{u}^k$	Array $\mathbf{u}$ evaluated at iteration k of a solver
$\mathbf{u}^{(i)}$	Local vector of subdomain i (different from $\mathbf{u}_i$)
A_{ij}	Coefficient (i,j) of matrix $\mathbf{A}$
$\mathbf{A}_{ij}$	Block (i,j) of a block matrix $\mathbf{A}$
$\mathbf{A}^{(i)}$	Local matrix of subdomain i
$1,2,\Gamma,\Gamma_1,\Gamma_2$	Block descriptors of subdomains and interfaces
s,t	Subdomain subscript: source, target
c,f	Subdomain subscript: coarse grid, fine grid

Table 1: Nomenclature.

2 Mesh-based DCM

Mesh-based DCM only take geometrical aspects into consideration to connect independent meshes. In general, the relative position between the meshes to be coupled is not a restriction. The coupled meshes could be disjoint, partial or totally overlapped (like Chimera-type problems) or even could present a gap between them.

Mesh-based DCM are generally referred to as meshing assembly models and their main drawback remains their parallel implementation. They are divided into two main groups:

- Conforming approaches:
  - creating a gap and meshing it with a conforming mesh;
  - mesh merging;
  - imprinting and merging.
- Non-conforming approach: HERMESH method.

Let us discuss both approaches.

2.1 Conforming approaches

The first group of mesh-based DCM is composed of methods that obtain a final conformal mesh from two independent meshes.

The Shear-Slip Mesh Update Method [6] is probably the simplest one, and it serves as a sliding-mesh technique to couple a fixed subdomain with a moving one in a conformal way. In this method, the two adjacent subdomains present a gap between them and have the same number of faces on each side of the interface. They are thus connected by a layer of elements to obtain a conforming mesh. These elements get distorted if (at least) one of the subdomains moves and, when the distortion is too severe, the gap is remeshed.

The second strategy, named DRAGON [62] (Direct Replacement of Arbitrary Grid Overlapping by Non-structured grid), generates a final conformal mesh from two structured meshes coupled together by one non-structured mesh. The non-structured mesh is formed a posteriori from a hole-cutting process. The hole is filled in with the non-structured mesh which connects both structured meshes in a conforming manner.

The third strategy, named Mesh Matching [78], deals with the issue of handling non-conforming hexahedral-to-hexahedral interfaces with the target of recovering a final all-hexahedral conformal mesh. This method locally modifies the topology of the hexahedral elements on one or on both sides of the interface surfaces, allowing the meshes to be merged into a conforming mesh across the interface.

In the literature, we also find mesh merging techniques to create a conforming mesh from two independent meshes by rebuilding them in the neighborhood of their intersection. These algorithms perform Boolean operations between groups of surface meshes to extract a common interface between the non-conforming meshes. Next, the elements coming from the non-conforming meshes that intersect these interface are removed, and the area is re-meshed conforming the interface and obtaining a final conformal mesh. In [64], the authors present an algorithm based on tracing the neighbors of intersecting triangles (TNOIT) to determine the intersection lines. Once this intersection is done, the nodes on the intersection lines are repositioned, elements cut by the intersection line are removed and the gap is meshed by the proper connection of nodes between the intersection line and the surface boundary. A similar technique is found in [21] to join a carotid artery with stenosis and to merge surface models representing the internal and external carotid arteries of a normal subject.

In the framework of finite volumes, we highlight the code developed by EDF R&D, *Code_Saturne* [75], which uses non-conforming meshes. Adjacent boundaries of non-conforming meshes are split into their intersecting subsets, obtaining a conforming mesh of polyhedra with an arbitrary number of faces per cell. The idea is to build new faces, which correspond to the intersections of the initial faces of the non-conforming meshes. The terminology used by the authors for this process is *conforming joining*.

We also mention the geometric technique of imprinting although this method differs from all the previous ones since it works at the geometric level rather than at the mesh level. The idea is to compute the intersection graph between two objects to create a coincident topology where both objects intersect. After the imprinting operation, a mesh mirror technique or a merging strategy is applied in order to obtain a conforming mesh between the two objects. Further details on this technique can be found in [83] or [25], where non-manifold interfaces between volumes are created from an assembly model for conformal meshing. This technique, together with other approaches, is used in the CUBIT project [14]. It is a full-featured software toolkit for the robust generation of two- and three-dimensional finite element meshes and geometry preparation. Its main goal is to reduce the time for generating meshes, and it is particularly oriented to generate large hex meshes of complicated and interlocking assemblies. One of the main functionalities of this code is to import two independent and non-conforming meshes and modifying them locally to obtain a final conformal one. This resulting mesh has been modified only locally in the interface of the original meshes.

Finally, the authors of this paper have also developed a strategy to couple hexahedral meshes in the framework of wind farm design and management. The proposed approach can be used both as a Chimera method (coupling intersecting meshes) or as a mesh merging technique (coupling disjoint meshes). Given a background hexahedral mesh and several hexahedral meshes of different objects to insert in the background mesh, the proposed method generates a final hybrid mesh conformal with the inserted objects. Either in the joint or disjoint case, the proposed method removes all the hexahedra necessary to insert the interior meshes and to guarantee a smooth transition between the sizes of the object and background meshes. If necessary, the method adds transition layers of hexahedra to the object meshes to smoothly recover the size of the background elements. Next, the proposed method uses several templates to divide the last layer of hexahedra of the background and object meshes into pyramids and tetrahedra, see [71]. Finally, it uses the Delaunay-based mesh generator TetGen [76] to fill with tetrahedra the gap between the different meshes.

Example

The problem at hand is the experimental *Sexbierum test* [26] which consists of a single wake case of a wind farm simulation where the Reynolds averaged Navier-Stokes equations with a specific $k-\varepsilon$ turbulence model are solved. The real topography was considered, using experimental wind measures as boundary conditions and the actuator disk theory to model the effect of the wind turbines. The actuator disk model [3] consists in introducing a sink in the momentum equations to extract volumetric momentum from the Navier-Stokes equations. We will now use this problem to test the hybrid conforming method. (The same test case will also be used to compare the HERMESH method presented in the next section, and the implicit algebraic Dirichlet/Dirichlet method introduced in Section 4.2.3).

This example requires applying the Chimera method, so that first a hole has to be created in the background mesh. The latter is a structured mesh with a boundary layer while the patch mesh containing the disk is formed either by an unstructured tetrahedral mesh (for the HERMESH and Dirichlet/Dirichlet) or by another hexahedral mesh for the hybrid conforming approach. The mesh is composed of a total of half a million elements in both cases, and the ratio between both sizes of the elements forming the independent meshes is 1:5. Figure 4 shows the mesh generation procedure for the *Sexbierum test* using the hybrid conforming method presented previously. Since this method is specially developed for wind farm simulations, the procedure itself generates the transition in the upwind and downstream directions so that it is smooth and has the desired resolution to capture the wake effects, with a growth factor between meshes that is always in the interval $(0.8, 1.2)$. In particular, in Figure 4 (Left) we illustrate the background mesh, where we can observe the region of hexahedra that will be removed and the hexahedral mesh that will be inserted. Next, Figure 4 (Right) shows the final generated hybrid mesh, where we can observe that the inner mesh grows smoothly to conform the background mesh size, and how the flexibility of tetrahedra is used to close the gap between both meshes.

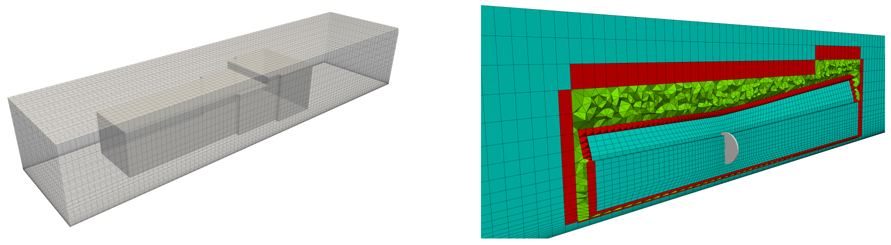


Fig. 4: Mesh generation process in the Sexbierum case using the conformal hybrid approach. (Left) Background wind mesh, where we also illustrate the hexahedral region to be removed and interior hexahedral mesh to be inserted. (Right) Final conformal hybrid mesh, where hexahedra are colored in blue, tetrahedra in green, and pyramids in red.

Finally, Figure 5 shows an entire wind farm using the conforming strategy in the case of 80 patches.

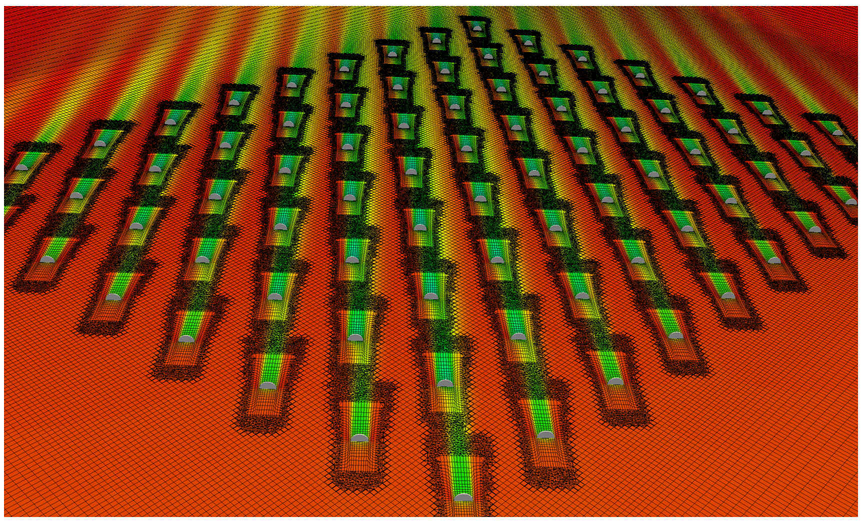


Fig. 5: Conformal hybrid approach applied to an entire wind farm. Mesh and velocity module.

2.2 Non-conforming approach

In the other class of strategies based on mesh manipulation we find the HERMESH method developed in [34,33]. It creates new connectivities between independent meshes with the existing nodes, such that the total number of nodes or degrees of freedoms is not modified. The continuity of the solution is achieved by introducing new elements which are treated slightly differently from the original elements of the meshes. These elements are referred to as *extension elements* since, in fact, they extend the mesh from one subdomain to the other by connecting the existing nodes of both meshes. The idea is illustrated in a simple one-dimensional case in Figure 6. At the top of the figure, we have shown a one-dimensional domain Ω and the shape functions associated to each node of the mesh. Next, in the middle, we have depicted two disjoint subdomains with a gap between them. Finally, at the bottom, we have drawn (with discontinuous lines) the two extension elements associated to each interface node, called fringe nodes. We can observe that in order to assemble the two independent meshes, two extension elements have been created while, in the one-domain case, the two fringe nodes are connected with one element. Each extension element connect the nodes of the adjacent subdomains in order to create a shape function with compact support. So, as the

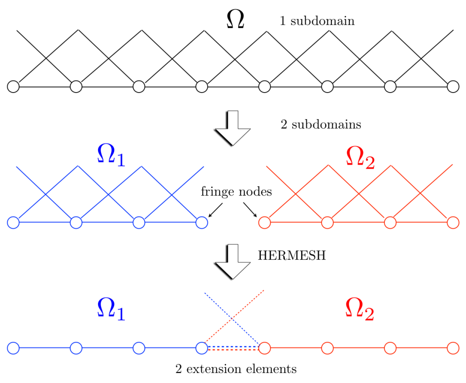


Fig. 6: HERMESH method. Joining disjoint subdomains in the one-dimensional case.

extension elements only contain the shape function associated to the fringe node, the contribution to the global matrix is different from that of normal finite elements.

The resulting method takes into account the transmission conditions implicitly and does not depend on the equation to be solved. The method was applied to the Navier-Stokes equations as a gluing mesh strategy in [35], as a Chimera method in [52], and to solid mechanics problems in [20]. Discretization error is nodally zero if the exact solution belongs to the finite element space and the L^2 norm of the solution error is consistent with that of the original scheme. If the coupled meshes plus the extension elements coincide with the one-domain mesh, then the solution is exactly the same as the one-domain solution. These properties and different applications of the method are presented in [33]. The main drawback of HERMESH method is the difficulty to implement it in parallel due to the huge amount of communications that it would imply. This issue is also discussed in the mentioned references.

Example

Let us consider the same example of Section 2.1. The hole cutting process creates 1932 hole elements in the background mesh. The geometry and extension elements are shown in Figure 7. At the bottom, the figure shows the extension elements of the patch and background. The first ones are tetrahedra as they connect the triangular faces of the patch interface to the nodes of the background. The second ones are pyramids, as they connect the quadrilateral faces of the background structured mesh to the nodes of the patch. Figure 8 shows some cuts in the vicinity of the wind turbine. We observe good continuity of the solution across the interfaces.

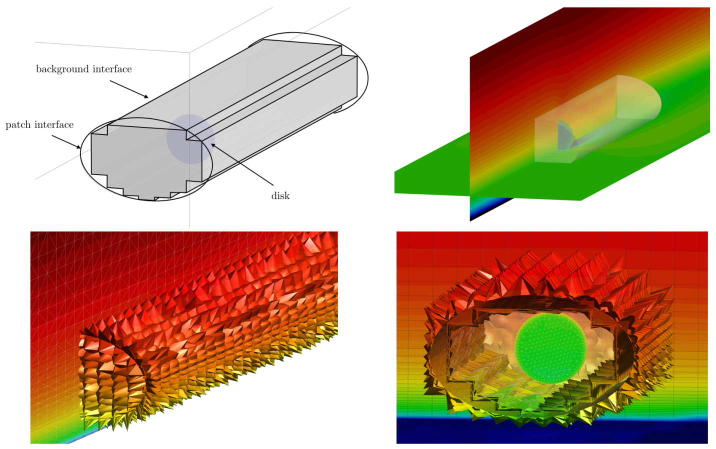


Fig. 7: HERMESH method. (Top left) Geometries of background and patch, and hole. (Top right) Velocity contours in the vicinity of the wind turbine. (Bottom left) Tetrahedra extension element from patch to background. (Bottom right) Pyramid elements from background to patch.

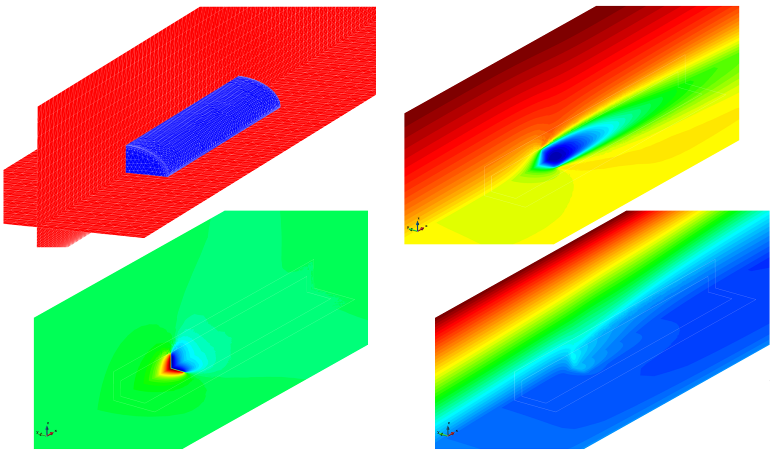


Fig. 8: HERMESH method. (Top left) Mesh of background and patch. (Top right) Velocity module. (Bottom left) Pressure. (Bottom right) Turbulent viscosity.

Finally, Figure 9 compares the velocity deficit obtained behind the wind turbine with the one-domain solution obtained by the conforming approach described in Section 2.1. We observe good agreement between the solutions.

(Note that the figure shows also the results of the algebraic implicit Dirichlet/Dirichlet method to be presented in Section 4.2.3.)

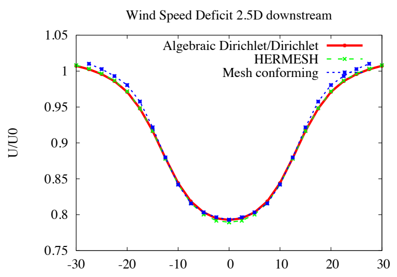


Fig. 9: Velocity deficit: HERMESH method compared with one-domain and algebraic Dirichlet/Dirichlet.

3 Equation-based DCM

The goal of a subdomain coupling strategy is to obtain a unique and global solution from local solutions obtained on separate subdomains. These subdomains can be disjoint or overlapping. Let Ω be the domain obtained as the union of subdomains and, for the sake of simplicity, let us consider two subdomains Ω_1 and Ω_2. If the subdomains are overlapping, we denote the interface of Ω_1 in Ω_2 as Γ_1, and the interface of Ω_2 in Ω_1 as Γ_2. If the subdomains are disjoint then the two interfaces coincide and the common interface is denoted by Γ. The outward normal direction to subdomain Ω_i is $\boldsymbol{n}_i$. A generic normal $\boldsymbol{n}$ will be used when any of the normals can be used. At the continuous level, variables in Ω_i will be denoted by u_i, for $i = 1, 2$. At the discrete level, they will be denoted as $u_{h,i}$.

3.1 Some techniques

As we mentioned in the introduction, in this work we aim to present those DCM with minimum impact on the original code and which enable multi-code couplings at a reasonable price. Therefore, we will not spend too much time on formulations that introduce additional degrees of freedom or new discrete spaces, apart from those coming from the original subdomains to be coupled. We have divided the equation-based DCM into three main categories:

– Constrained coupling methods.
– Mesh-free methods.
– Transmission-conditions based methods.

In this introductory subsection, we will briefly comment on the first two categories, while the remaining part of this section will entirely be devoted to the last type of equation-based DCM.

Constrained coupling methods

Let us consider two disjoint subdomains, non-matching on their interfaces. Constrained coupling methods consist in imposing the continuity of some quantities across the subdomains by means of constraint equations. The most common methods in the literature are the three field method [18] and the mortar element method [10]. In the three field method, Lagrange multipliers are introduced in both subdomains to enforce the continuities of the unknown and its flux across the interfaces, in a weak sense. For example, given a test function μ in a suitable trace space, say Λ, on the interface Γ, the weak continuity of the unknown is expressed as:

$$\int_\Gamma \mu(u_1 - u_2) d\Gamma = 0 \quad \forall\, \mu \in \Lambda.$$

Obviously, the cost of such a method is quite high and implementation can be a non-trivial task as different interface grids may have to be generated to discretize each constraint space.

The mortar element method [10,11]

> *[. . .] is a domain decomposition technique that allows to take benefit of the presence of the subdomains in order to choose the discretization method the best adapted to the local behavior of the solution of the partial differential equation which must be approximated [. . .]*

and provides a remarkable theoretical framework for coupling non-matching meshes. Basically, the constraint expresses the weak continuity of the variable across the interface. The method can then be implemented as an iteration-by-subdomain algorithm (e.g. Dirichlet/Neumann) or via the introduction of a Lagrange multiplier to account for the constraint. This technique has been applied not only to couple different discretization methods, but also to contact problems, to fluid-structure interaction problems, or to treat non-conforming adaptivity [46]. However, the implementation of the mortar element method is onerous.

Let us finally mention the Finite Element Tearing and Interconnecting method (FETI), which differs from the mortar element method in the choice of the discrete space of the Lagrange multiplier [36].

Mesh-free methods

The basic idea of the coupling via mesh-free methods is to define mesh-free connections in the interface between finite element grids using new elements called *meshless gluing elements*. The main difficulty of this strategy is determining the influence domain of the elements of the interface since the shape functions do not maintain the same properties of the classical finite element shape functions. Different strategies to interpolate the information between the independent meshes are found in the literature. In [55] we find the moving-least-square (MLS) interpolation, while in [79] we find compact support radial basis function (CS-RBF), radial point interpolation method (RPIM) and moving Kriging interpolation method (MKIM). A general presentation of these methods can be found in [7] and an excellent review in [57]. A useful application of this hybrid combination is in enrichment purposes, see [63] and [53]. A general review of the existing methods for this family of hybrid formulations can be found in [73].

3.2 Transmission conditions

In this section we will describe the coupling methods based on transmission conditions of Dirichlet, Neumann and Robin type. We will start describing what is sought at the continuous level, and introducing some terminology. Then in Section 3.3.2, we will discuss the concept of extension operators at the variational level which enables one to show the equivalence of the Dirichlet/Neumann formulation and the one-domain formulation (Section 3.3.3). We will also explain why the solution of the Dirichlet/Neumann problem is not equivalent to the one-domain solution at the discrete level in Section 3.4. Conservation aspects will be treated at the algebraic level in Section 4.3, as similar techniques can be employed in the equation-based context. We will then finalize this section by introducing explicit and implicit couplings of the subdomains using transmission conditions, in Sections 3.4.4 and 3.4.5, respectively.

We would like to stress that in the domain decomposition community, for which the main purpose of DDM is to parallelize the computation, the variational formulation is mainly used to derive the mathematical properties of the DDM methods (equivalence with one-domain, convergence properties, etc.). However, these DDM methods are most of the time implemented at the algebraic level, in the way to be presented in Section 4. In fact, their algebraic implementations enable one to avoid defining extension operators (which then comes naturally), computing surface integral, etc. In the following we intend to present the discrete methods based on transmission conditions as directly implementable for domain composition purposes.

Let us consider the following advection-diffusion problem

$$\begin{aligned} \mathcal{L}(u) &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega, \\ \text{with} \quad \mathcal{L}(u) &:= -\varepsilon\Delta u + \boldsymbol{a}\nabla\cdot u, \end{aligned} \tag{1}$$

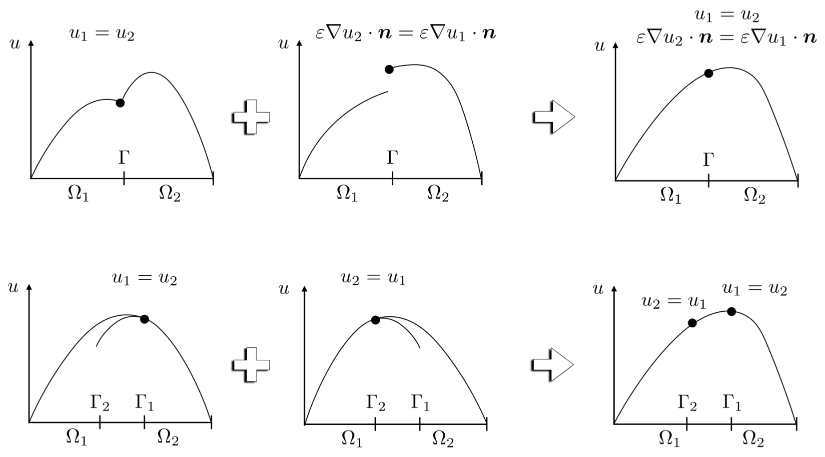

u
$u_1 = u_2$
Γ
Ω1
Ω2
$\varepsilon\nabla u_2 \cdot \boldsymbol{n} = \varepsilon\nabla u_1 \cdot \boldsymbol{n}$
$u_1 = u_2$
Γ2
Γ1
$u_2 = u_1$

Fig. 10: Domain composition methods. $\boldsymbol{n}$ is the outward normal to one of the subdomains. (Top) Dirichlet/Neumann method for disjoint subdomains (also valid for overlapping subdomains). (Bottom) Dirichlet/Dirichlet or Schwarz method only valid for overlapping subdomains.

where ε is the diffusion coefficient and $\boldsymbol{a}$ the divergence-free advection field. This equation was selected for the sake of clarity but the exposition of this section holds for a general advection-diffusion-reaction equation with more general boundary conditions. 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. We will now briefly introduce the different existing techniques to treat disjoint subdomains, and then overlapping subdomains.

3.2.1 Disjoint subdomains

Ideally, when designing a DCM method, one requires to obtain the same global solution as the one-domain solution. For disjoint subdomains [72], this is achieved by imposing both the continuity of the variable $u_1 = u_2$ and its flux $\varepsilon\nabla u_1 \cdot \boldsymbol{n} = \varepsilon\nabla u_2 \cdot \boldsymbol{n}$ at the interface $\mathit{\Gamma}$ of the subdomains, as illustrated by Figure 10 (Top). This coupling is referred to as Dirichlet/Neumann, with obvious meaning. Many alternative methods have been devised in the literature, which all allow to impose the continuity both of the variable and of its flux. By introducing some parameters α_i and β_i for $i = 1, 2$, we note that by imposing the following transmission conditions [27]:

$$\begin{cases} \beta_1\varepsilon\nabla u_1 \cdot \boldsymbol{n} + \alpha_1 u_1 = \beta_1\varepsilon\nabla u_2 \cdot \boldsymbol{n} + \alpha_1 u_2, \\ \beta_2\varepsilon\nabla u_2 \cdot \boldsymbol{n} + \alpha_2 u_2 = \beta_2\varepsilon\nabla u_1 \cdot \boldsymbol{n} + \alpha_2 u_1, \end{cases} \tag{2}$$

and as long as $|\alpha_1\beta_2 - \alpha_2\beta_1| \neq 0$, we obtain both the continuity of the variable and its flux across the interface Γ, that is:

$$\begin{cases} u_1 = u_2, \\ \varepsilon \nabla u_1 \cdot \boldsymbol{n} = \varepsilon \nabla u_2 \cdot \boldsymbol{n}. \end{cases}$$

Depending on the values of α_i and β_i, we end up with different transmission conditions:

$$\begin{cases} \text{Dirichlet:} & \alpha_i \neq 0,\ \beta_i = 0, \\ \text{Neumann:} & \alpha_i = 0,\ \beta_i \neq 0, \\ \text{Robin:} & \alpha_i \neq 0,\ \beta_i \neq 0. \end{cases}$$

Different combinations of these conditions lead to different coupling methods. Let us mention the Dirichlet/Neumann [13,67,72,37], Dirichlet/Robin [1], Robin/Robin [61,45] and Robin/Neumann [19] methods.

Adaptive methods take into account the local character of the advection on the interface [81,19,23,1,41,2]. Typically, in these methods, a Dirichlet condition is imposed at the inflow boundary, i.e., where $\boldsymbol{a} \cdot \boldsymbol{n} < 0$, and a Neumann condition at the outflow boundary, i.e. where $\boldsymbol{a} \cdot \boldsymbol{n} \geq 0$, with $\boldsymbol{n}$ being the outward normal of the subdomain under consideration.

Finally, when the elliptic operator degenerates to a first order hyperbolic one for $\varepsilon \to 0^+$ in (at least) one subregion of Ω, *heterogenous methods* allow to impose different transmission conditions across the interface, to take into account the different nature of the governing equations of each subdomain [42,38,30]. One example of application arises in the coupling of viscous and inviscid flows [17,12].

3.2.2 Overlapping subdomains

The case of overlapping subdomains has mainly been considered in the context of the Schwarz method [59,60]. See also [39] for a historical perspective of the Schwarz method. Its original version consists in imposing Dirichlet boundary conditions on both interfaces Γ_1 and Γ_2, as illustrated in Figure 10 (Bottom). However, the main applications of the Schwarz method have focused on DDM in the context of parallelization techniques. In fact, when implemented at the algebraic level, the Schwarz method can be used as a solver or as a preconditioner of a Krylov method. It should be noted that the term Schwarz has also been used in the literature to designate the Robin/Robin method applied to non-overlapping subdomains [61].

Some mixed transmission conditions described for disjoint subdomains have been also applied to overlapping subdomains [51], with applications to the Chimera method [50]. However, they have received little attention in the literature [69,70,56].

We point out that methods considering different types of transmission conditions on the interfaces of overlapping subdomains have been studied also in the context of the so-called virtual control methods [58,43], the least squares conjugate gradient method [44], and recently, the interface control domain decomposition methods [31,32,29].

3.3 Continuous level

In the following we will explain how to devise domain composition methods and to show their equivalence to the one-domain formulation.

3.3.1 One-domain formulation

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

$$a(u,v) := \varepsilon(\nabla u, \nabla v) + (1-\flat)(\boldsymbol{a}\cdot\nabla u, v) - \flat(u, \boldsymbol{a}\cdot\nabla v).$$

To derive this bilinear form, we have applied the Gauss theorem to only a $\flat$ portion of the advection term. By choosing $\flat = 0, 1/2, 1$ one can obtain different formulations with specific coercivity properties, which can be useful in the context of DCM [47]. For the moment we will consider $\flat = 0$ and will go back to other options later on. Let us denote the duality paring between H^1 and H^{-1} by $\langle\cdot,\cdot\rangle$. The weak problem consists in finding $u \in V^0$ such that

One-domain variational formulation:

$$a(u,v) = \langle f, v\rangle \quad \forall v \in V^0 \tag{3}$$

where V^0 is the subspace of $H^1(\Omega)$ whose functions are null on the boundary $\partial\Omega$.

3.3.2 Extension-based two-subdomain formulation

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 [51]. We first need to introduce some notations, as illustrated in Figure 11. The subscript indicates the subdomain support of the spaces; superscript 0 indicates that the functions of the space vanish on all the subdomain boundaries; if the superscript is omitted, then the functions do not vanish on the interface. Λ is the trace space on the interface Γ; finally, E_i are some continuous extension operators from the interface trace space Λ onto the associated subdomain space V_i. At the discrete level for example, a possible extension operator would extend linear functions on the interface nodes linearly to zero on the first interior nodes. Let us define $a_i(u,v)$ the bilinear

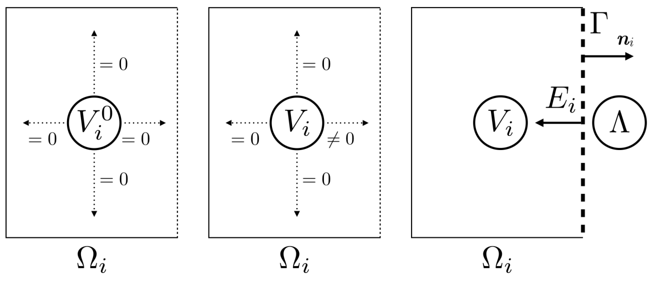


Fig. 11: Notations: geometrical entities Ω_i,, Γ and outward normal $\boldsymbol{n}_i$; functional spaces V_i, V_i^0 and Λ; extension operator E_i which extend functions in the trace space Λ into V_i. Subscript $i = 1, 2$ stands for the subdomain number.

form restricted to Ω_i and $\langle \cdot, \cdot \rangle_{\Omega_i}$ the duality paring between H^1 and H^{-1} in Ω_i.

The extension-based two-subdomain formulation, described thoroughly in [72], reads:

Extension-based variational formulation

$$\begin{cases} a_1(u_1, v_1) = \langle f, v_1 \rangle_{\Omega_1} & \forall\, v_1 \in V_1^0 \\ u_1 = u_2 & \text{on } \Gamma \\ a_2(u_2, v_2) = \langle f, v_2 \rangle_{\Omega_2} & \forall\, v_2 \in V_2^0 \\ a_1(u_1, E_1\mu) + a_2(u_2, E_2\mu) = \langle f, E_1\mu \rangle_{\Omega_1} + \langle f, E_2\mu \rangle_{\Omega_2} & \forall\, \mu \in \Lambda \end{cases} \tag{4}$$

Equation $(4)_1$ and $(4)_3$ are the weak forms of the boundary value problems in Ω_1 and Ω_2, respectively. Note that both test functions v_i are in V_i^0 and therefore vanish on the interface. Equation $(4)_2$ imposes the continuity of the local solutions u_i across the interface Γ. Finally, the last equation completes the system by imposing the continuity of the conormal derivatives across the interface in weak sense. When discretizing problem (4), we will see that by properly choosing the extensions into V_1 and V_2, we can recover the original algebraic system. In [72], two important results are shown:

– System (4) is equivalent to system (3);
– System (4) implies that $\varepsilon \nabla u_2 \cdot \boldsymbol{n}_2 = \varepsilon \nabla u_1 \cdot \boldsymbol{n}_2$ in a weak sense.

3.3.3 Dirichlet/Neumann two-subdomain formulation

Using the (weak) continuity of the conormal derivatives $(4)_4$, we can rewrite the extension-based variational formulation (4) as a Dirichlet/Neumann variational formulation as:

Dirichlet/Neumann variational formulation

$$\begin{cases} a_1(u_1, v_1) = \langle f, v_1 \rangle_{\Omega_1} & \forall\, v_1 \in V_1^0 \\ u_1 = u_2 & \text{on } \Gamma \\ a_2(u_2, v_2) = \langle f, v_2 \rangle_{\Omega_2} + \int_\Gamma v_2 \varepsilon \nabla u_1 \cdot \boldsymbol{n}_2 \, d\Gamma & \forall\, v_2 \in V_2 \end{cases} \tag{5}$$

where the Neumann transmission condition $\varepsilon\nabla u_2 \cdot \boldsymbol{n}_2 = \varepsilon\nabla u_1 \cdot \boldsymbol{n}_2$ on Γ was substituted in last equation. We note that the third equation is provided with the Neumann condition expressed by Equation $(5)_4$ and therefore the test function space V_2 does no longer vanish on Γ (in contrast to Equation $(4)_3$). The equivalence between Systems (4) and (5) justifies why the latter one is referred to as Dirichlet/Neumann method: a Dirichlet transmission condition $(5)_2$ is applied to solve subdomain 1, while a Neumann transmission condition is applied to solve subdomain 2. Note that in [51], equivalent results are shown for overlapping subdomains. The equivalence between (3), (4) and (5) is illustrated in Figure 12 (Top). We will now see what happens when we try to apply these formulations at the discrete level.

3.4 Discrete level

3.4.1 Dirichlet/Neumann

Let us discretize in space and construct finite dimensional functional spaces to find a discrete solution u_h. Then, we can show that the discrete counterpart of System (4), referred to as the discrete two-subdomain extension-based variational form, is equivalent to the discrete counterpart of System (3), referred to as the discrete one-domain variational form. However, in practice, the formulation (4) seems not to be convenient to implement in a multi-code environment, and one would rather use the discretization of (5):

Discrete Dirichlet/Neumann variational formulation

$$\begin{cases} a_1(u_{1,h}, v_{1,h}) = \langle f, v_{1,h} \rangle_{\Omega_1} & \forall\, v_{1,h} \in V_{1,h}^0 \\ u_{1,h} = u_{2,h} & \text{on } \Gamma \\ a_2(u_{2,h}, v_{2,h}) = \langle f, v_{2,h} \rangle_{\Omega_2} + \int_\Gamma v_{2,h} \varepsilon \nabla u_{1,h} \cdot \boldsymbol{n}_2 \, d\Gamma & \forall\, v_{2,h} \in V_{2,h} \end{cases} \tag{6}$$

It is in fact relatively easier to interpolate a derivative on a boundary than to design a generic extension operator. In Section 4.1.1, we will devise a particular and simple extension operator which leads to an algebraic system identical to that of the one-domain formulation.

Although system (6) seems to be convenient, we face a problem: the lack of equivalence of its solution to that of the one-domain discrete formulation. Let us go back to the continuous formulation. In order to prove that the extension-based formulation implies $\varepsilon\nabla u_1 \cdot \boldsymbol{n}_2 = \varepsilon\nabla u_2 \cdot \boldsymbol{n}_2$, we have to use the fact that $\mathcal{L}(u) = f$, as shown in [72]. Since at the discrete level $\mathcal{L}(u_h) = f$ does not

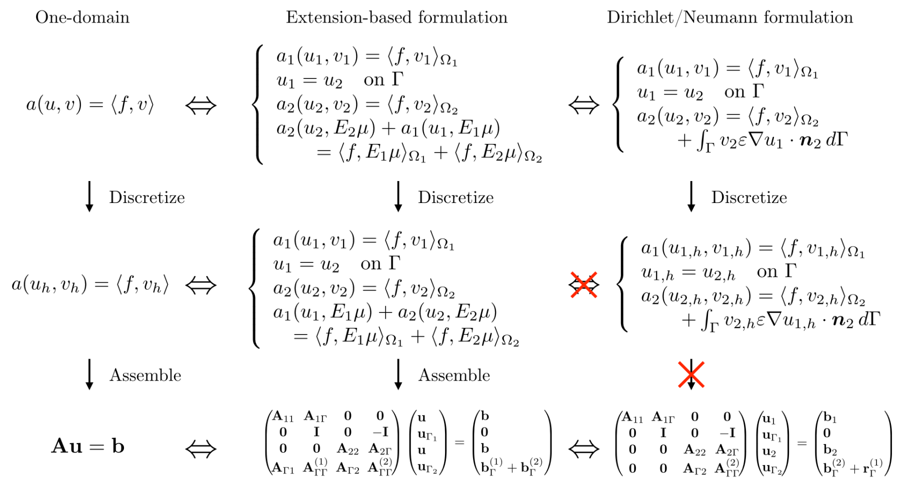

One-domain
Extension-based formulation
Dirichlet/Neumann formulation
$a(u,v) = \langle f, v\rangle$
$a_1(u_1,v_1) = \langle f, v_1\rangle_{\Omega_1}$, $u_1 = u_2$ on Γ, $a_2(u_2,v_2) = \langle f, v_2\rangle_{\Omega_2}$, $a_2(u_2,E_2\mu) + a_1(u_1,E_1\mu) = \langle f, E_1\mu\rangle_{\Omega_1} + \langle f, E_2\mu\rangle_{\Omega_2}$
$a_1(u_1,v_1) = \langle f, v_1\rangle_{\Omega_1}$, $u_1 = u_2$ on Γ, $a_2(u_2,v_2) = \langle f, v_2\rangle_{\Omega_2} + \int_\Gamma v_2 \varepsilon \nabla u_1 \cdot \boldsymbol{n}_2 \, d\Gamma$
Discretize
$a(u_h,v_h) = \langle f, v_h\rangle$
$a_1(u_1,v_1) = \langle f, v_1\rangle_{\Omega_1}$, $u_1 = u_2$ on Γ, $a_2(u_2,v_2) = \langle f, v_2\rangle_{\Omega_2}$, $a_1(u_1,E_1\mu) + a_2(u_2,E_2\mu) = \langle f, E_1\mu\rangle_{\Omega_1} + \langle f, E_2\mu\rangle_{\Omega_2}$
$a_1(u_{1,h},v_{1,h}) = \langle f, v_{1,h}\rangle_{\Omega_1}$, $u_{1,h} = u_{2,h}$ on Γ, $a_2(u_{2,h},v_{2,h}) = \langle f, v_{2,h}\rangle_{\Omega_2} + \int_\Gamma v_{2,h} \varepsilon \nabla u_{1,h} \cdot \boldsymbol{n}_2 \, d\Gamma$
Assemble
$\mathbf{A}\mathbf{u} = \mathbf{b}$
$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{1\Gamma} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{2\Gamma} \\ \mathbf{A}_{\Gamma 1} & \mathbf{A}_{\Gamma\Gamma}^{(1)} & \mathbf{A}_{\Gamma 2} & \mathbf{A}_{\Gamma\Gamma}^{(2)} \end{pmatrix} \begin{pmatrix} \mathbf{u} \\ \mathbf{u}_{\Gamma_1} \\ \mathbf{u} \\ \mathbf{u}_{\Gamma_2} \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ \mathbf{0} \\ \mathbf{b} \\ \mathbf{b}_\Gamma^{(1)} + \mathbf{b}_\Gamma^{(2)} \end{pmatrix}$
$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{1\Gamma} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{2\Gamma} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{\Gamma 2} & \mathbf{A}_{\Gamma\Gamma}^{(2)} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_{\Gamma_1} \\ \mathbf{u}_2 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{0} \\ \mathbf{b}_2 \\ \mathbf{b}_\Gamma^{(2)} + \mathbf{r}_\Gamma^{(1)} \end{pmatrix}$

Fig. 12: Relation between the two-domain extension-based, the Dirichlet/Neumann and the one-domain formulations. (Top) Continuous variational forms. (Middle) Discrete variational forms. (Bottom) Algebraic systems.

hold in general, there is no equivalence. Therefore, if we consider system (6), we will not obtain a system equivalent to (4). This implies that we will not find the same solution u_h as if we would have solved the discrete one-domain problem (3). This lack of equivalence is illustrated in Figure 12 (Mid).

To summarize, the discrete counterpart of problems and (3) and (4) are equivalent, as shown in [72]. However, a similar result does not hold between the discrete counterparts of (3) and (6) since, in general, $\mathcal{L}(u_h) \neq f$.

3.4.2 Dirichlet/Robin

Similar formulations can be obtained for the Dirichlet/Robin, Robin/Robin and Robin/Neumann couplings, for both disjoint or overlapping subdomains. For example, considering the general Robin condition given by Equation (2), and taking $\alpha_1 = 1$, $\beta_1 = 0$ and $\beta_2 = 1$, the formulation reads:

Discrete Dirichlet/Robin variational formulation

$$\begin{cases} a_1(u_{1,h}, v_{1,h}) = \langle f, v_{1,h}\rangle_{\Omega_1} & \forall\, v_{1,h} \in V_{1,h}^0 \\ u_{1,h} = u_{2,h} & \text{on } \Gamma \\ a_2(u_{2,h}, v_{2,h}) + \int_\Gamma v_{2,h}\alpha_2 u_{2,h} d\Gamma = \langle f, v_{2,h}\rangle_{\Omega_2} & \\ \qquad + \int_\Gamma v_{2,h}(\varepsilon\nabla u_{1,h}\cdot \boldsymbol{n}_2 + \alpha_2 u_{1,h}) d\Gamma & \forall\, v_{2,h} \in V_{2,h} \end{cases} \tag{7}$$

Now let us consider $\flat \neq 0$. We can show that the Dirichlet/Neumann formulation becomes naturally a Dirichlet/Robin method and the equation in subdomain Ω_2 becomes [47]:

$$a_2(u_{2,h}, v_{2,h}) = \langle f, v_{2,h} \rangle_{\Omega_2} + \int_\Gamma v_{2,h}(\varepsilon \nabla u_{1,h} \cdot \boldsymbol{n}_2 - \flat(\boldsymbol{a} \cdot \boldsymbol{n}_2) u_{1,h}) d\Gamma. \quad (8)$$

In fact, we can show that the transmission condition in Ω_2 coincides with the natural condition of the variational problem. Referring to the general Robin transmission conditions in Equation (2), the last formulation would correspond to the choice:

$$\begin{aligned} &\text{Dirichlet: } \alpha_1 = 1, && \beta_1 = 0, \\ &\text{Robin: } \quad \alpha_2 = -\flat(\boldsymbol{a} \cdot \boldsymbol{n}_2), && \beta_2 = 1. \end{aligned} \quad (9)$$

Similarly to the Dirichlet/Neumann formulation (6), the discrete Dirichlet/Robin formulation is not equivalent to the discrete one-domain formulation.

3.4.3 Computing the Neumann condition

So what can we do with these bad news? Even if we cannot establish the equivalence between the one- and two-domain formulations, one can always discretize the Neumann condition and hope to obtain a reasonable solution, whose rate of mesh convergence is consistent with that of the underlying discretization scheme. For the sake of generality, let us consider possibly overlapping subdomains and denote by Γ_2 the interface of Ω_2 in Ω_1. The Neumann condition consists in computing the following integral on the interface Γ_2 and assembling it to the algebraic system in Ω_2:

$$\int_{\Gamma_2} \varepsilon v_{2,h} \nabla u_{1,h} \cdot \boldsymbol{n}_2 \, d\Gamma.$$

To assemble this term, we therefore require the gradient of the solution at the Gauss points of the interface Γ_2, noted $(\nabla u_{1,h})(\boldsymbol{x}_g)$, where $\boldsymbol{x}_g$ is the coordinate of the Gauss point. Let us consider two possibilities, illustrated in Figure 13. Let us assume that the Gauss point $\boldsymbol{x}_g$ is hosted by element e, with *nnode* nodes in Ω_1. Let us denote by $N^{(i)}(\boldsymbol{x}_g)$ the shape function of node i in e, evaluated at $\boldsymbol{x}_g$. On the one hand, the gradients can be computed by computing the derivatives of the interpolated solution $u_{1,h}$ in element e:

$$(\nabla u_{1,h})(\boldsymbol{x}_g) = \sum_{i=1}^{nnode} \nabla N^{(i)}(\boldsymbol{x}_g) \, u_{1,h}^{(i)}.$$

In [50], the authors show that this scheme is only first order in space, therefore penalizing the mesh convergence of the overall solution. Another option

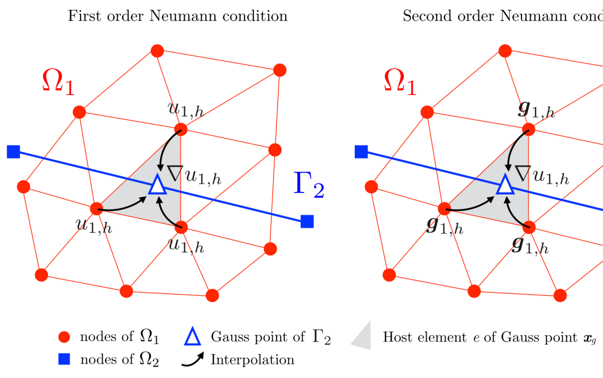

First order Neumann condition
Second order Neumann condition
Ω_1
Γ_2
$u_{1,h}$
$\nabla u_{1,h}$
$\boldsymbol{g}_{1,h}$
nodes of Ω_1
nodes of Ω_2
Gauss point of Γ_2
Interpolation
Host element e of Gauss point $\boldsymbol{x}_g$

Fig. 13: Two ways of computing the gradient, required at the interface Gauss points, involved in the Neumann condition. (Left) Gradients are interpolated at the Gauss point from the solution. (Right) Gradients are first projected $\boldsymbol{g}_h$, and then interpolated at the Gauss points.

consists in projecting the gradients in Ω_1 and then interpolate them at the Gauss points:

$$\text{Projection:} \qquad \int_{\Omega_1} \boldsymbol{g}_{1,h}\, v_{1,h}\, d\Omega = \int_{\Omega_1} \nabla u_{1,h}\, v_{1,h}\, d\Omega, \tag{10}$$

$$\text{Interpolation:} \qquad (\nabla u_{1,h})(\boldsymbol{x}_g) = \sum_{i=1}^{nnode} N^{(i)}(\boldsymbol{x}_g)\, \boldsymbol{g}_{1,h}^{(i)}.$$

Note that to correctly project the gradients, which can only be first order on the boundary nodes, one must consider an overlap of at least one layer of elements. Using this one-layer overlap, and through the solutions of numerical examples, the authors of [50,47] demonstrate that this scheme is second order and is thus consistent with the order of the finite element scheme. Finally, note that the projection step given by Equation (10) can be trivially solved using a lumped mass matrix to represent the left-hand side.

3.4.4 Explicit coupling

The formulations previously presented are monolithic in the sense that the solutions in both subdomains are implicitly coupled through the transmission conditions. We are now going to consider an explicit decoupling. The explicit solution of the Dirichlet/Neumann variational formulation consists in solving the two subproblems independently, at the same time or in a staggered way, by delaying the updates of the transmission conditions. This is formally achieved by introducing an iteration index k and solving the following problem. Given initial guesses for the local solutions, solve for $k = 0, 1, 2, \dots$:

Iteration-by-subdomain Dirichlet/Neumann

$$\begin{cases} a_1(u_{1,h}^{k+1}, v_{1,h}) = \langle f, v_{1,h}\rangle_{\Omega_1} & \forall\, v_{1,h} \in V_{1,h}^0 \\ u_{1,h}^{k+1} = u_{2,h}^k & \text{on } \Gamma \\ a_2(u_{2,h}^{k+1}, v_{2,h}) = \langle f, v_{2,h}\rangle_{\Omega_2} + \int_\Gamma v_{2,h}\varepsilon\nabla u_{1,h}^{k+m}\cdot \boldsymbol{n}_2\, d\Gamma & \forall\, v_{2,h} \in V_{2,h} \end{cases}$$

This method is referred to as iteration-by-subdomain domain decomposition method in the literature. In this system, $m = 0, 1$ according to whether the problems are solved one after the other or at the same time, respectively. The methods are referred to as Jacobi (or parallel) and Gauss-Seidel (or sequential), respectively. The same iteration-by-subdomain methods can be obtained for all mixed methods (Dirichlet/Robin, Robin/Robin) and for the Dirichlet/Dirichlet couplings.

The convergence of the iterative method is an important issue and it depends on many factors: the geometry of Ω_1 and Ω_2, the coefficients of the equation ε and $\boldsymbol{a}$, the location of the interface, etc. This issue has been studied extensively in the literature. See for example [51] for the comparison study of the convergences of the disjoint and overlapping Dirichlet/Robin and adaptive Dirichlet/Neumann methods, and Dirichlet/Dirichlet method. Note that a simple way to enhance the convergence consists in relaxing the update of the transmission conditions, for example: $u_{1,h}^{k+1} = \alpha u_{2,h}^k + (1-\alpha)u_{1,h}^k$, where α is the relaxation factor.

3.4.5 Implicit coupling

For the sake of generality, let us consider a coupling of non-conforming and overlapping meshes. The implicit coupling, represented by System (6), is not at all an easy task to carry out in parallel, because it involves the creation in each partition of new degrees of freedom (coming from the adjacent subdomains) and would require the reconstruction of the communication arrays. However, it can be relatively easily done as a preprocess, before partitioning the mesh. The methodology is explained in [33] and illustrated in Figure 14. Let us take the example of the Dirichlet condition. By using a linear interpolation, we have the following equation for the unknown $u_{2,1}$ in Ω_2:

$$u_{2,i} - \sum_{i=1}^{4} N_i u_{1,i} = 0,$$

where the shape functions N_i are evaluated in Ω_1 at the position of the node of subdomain Ω_2. The idea is then to create a virtual element, whose connectivity array contains the four nodes of subdomain Ω_1 and the Dirichlet node of Ω_2. When partitioning the mesh by elements, only one partition will therefore host the virtual element. This partition will then be in charge of assembling the complete equation for this Dirichlet node. Exactly the same strategy can be applied to impose implicitly the Neumann condition, as explained in [33].

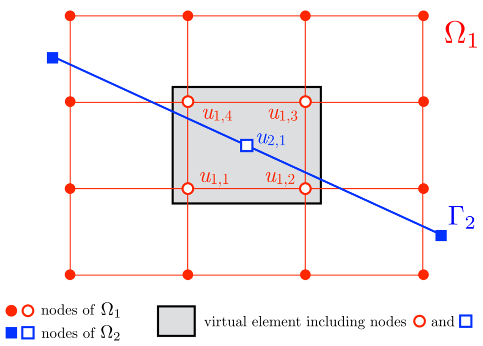


Fig. 14: Introduction of virtual elements to impose Dirichlet and Neumann conditions in an implicit way.

3.4.6 Summary

In this section we have presented some domain composition methods based on transmission conditions, for disjoint and overlapping subdomains, and based on both explicit and implicit couplings. One of the major problems of the methods involving a Neumann or Robin condition is the lack of equivalence with the one-domain formulation and the possible deterioration of the finite element mesh convergence when the derivatives are not computed in a proper way. In addition, such methods are not general enough: the Neumann condition involves the physical coefficients of the equation in play, and one should therefore code one ad-hoc Neumann condition for each physical problem to be coupled. In the next section, we will examine algebraic-based couplings, which allows to solve all these issues at once.

4 Algebraic-based DCM

For the continuous variational forms, both the extension-based and Dirichlet/Neumann formulations, respectively given by Systems (4) and (5), are equivalent to the one-domain formulation (3). At the discrete level, we only have equivalence for the discrete counterpart of the extension-based formulation. This section provides an alternative approach. We will show how to construct algebraic counterparts of the transmission-based domain composition techniques. The advantage of working at the algebraic level is twofold: one the one hand, one can show the equivalence between the different two-domain formulations and the one-domain problem; on the other hand, working on the algebraic systems enables one to hide the transmission conditions in the iterative solver in order to create an implicit method and thus avoid convergence issues. This second advantage brings into light a subtle but fundamental issue, although, frequently, a forgotten factor: at the deep core of a simulation code lays an algebraic solver, which is the chief responsible for both successful speed-ups and painful bottlenecks. Therefore, it is highly desirable to concentrate on this level as many numerical issues as possible to, afterwards, design the best and most efficient solving strategy. This represents a point of view that moves from the paper to the computer.

In Section 4.1.1 we will explicitly present the algebraic formulations of the Dirichlet/Neumann, Dirichlet/Robin, Robin/Robin and Dirichlet/Dirichlet methods. In Section 4.1.2, the explicit coupling, referred to as iteration-by-subdomain method will be introduced to decouple the solutions on the different subdomains. Implicit method will be introduced in Section 4.1.4. Then, in Section 4.2, we will explain how to write the previous formulation in the context of non-matching meshes and describe some interpolation and projection transmission matrices. Section 4.3 will deal with conservation aspects. In Section 4.4, we will comment on implementation aspects in a distributed memory context, for both explicit and implicit couplings.

4.1 Coupling strategies

We will start considering mixed methods for disjoint subdomains, namely the Dirichlet/Neumann, Dirichlet/Robin and Robin/Robin methods. For the overlapping case, we will focus on a Dirichlet/Dirichlet (Schwarz) method. Mixed methods for overlapping subdomains are also an option, but not straightforward in the context of a multiple-code coupling [47]. Then we will explain two ways of decoupling the solutions on the subdomains and how these local solutions can be recomposed to obtain a global solution. On the one hand, explicit coupling leads to the so-called iteration-by-subdomain methods. We will explain the close relation of such methods with the solution of the interface Schur complement and treat convergence issues. On the other hand, implicit coupling is achieved at the matrix-vector product level. In the case of the Dirichlet/Dirichlet method, we will introduce two possible implementations.

To start with, we will consider matching meshes on the interface. We will eventually introduce strategies to deal with non-matching meshes in Section 4.2.

4.1.1 Dirichlet, Neumann and Robin conditions

Let us discretize on a finite element mesh the one-domain formulation (3) and construct the corresponding algebraic system. We denote by $\mathbf{u}$ the vector of unknowns in Ω. The global algebraic system reads:

$$\mathbf{A}\mathbf{u} = \mathbf{b}. \tag{11}$$

We denote by $\mathbf{u}_1$ and $\mathbf{u}_2$ the vectors of interior unknowns of Ω_1 and Ω_2 respectively, excluding the interface vector of unknowns that we denote by $\mathbf{u}_\Gamma$.

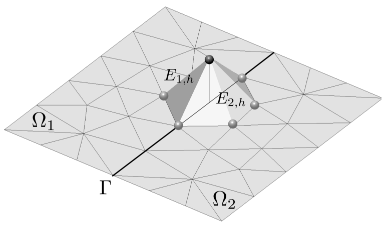


Fig. 15: Extension operator for an interface node coinciding with its classical shape function.

By performing a simple node reordering, System (11) can be written as:

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{A}_{1\Gamma} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{2\Gamma} \\ \mathbf{A}_{\Gamma 1} & \mathbf{A}_{\Gamma 2} & \mathbf{A}_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_\Gamma \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_\Gamma \end{pmatrix}.$$

The submatrices of the interface equation come from the assembly involving the elements in both Ω_1 and Ω_2. With obvious meaning, we denote as $\mathbf{A}^{(i)}_{\Gamma\Gamma}$ and $\mathbf{b}^{(i)}$ the submatrix and sub right-hand-side coming from the integration over subdomain i. We can rewrite the latter system as:

One-domain algebraic formulation

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{A}_{1\Gamma} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{2\Gamma} \\ \mathbf{A}_{\Gamma 1} & \mathbf{A}_{\Gamma 2} & \mathbf{A}^{(1)}_{\Gamma\Gamma} + \mathbf{A}^{(2)}_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_\Gamma \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}^{(1)}_\Gamma + \mathbf{b}^{(2)}_\Gamma \end{pmatrix} \tag{12}$$

Assembly of the extension-based formulation

Now let us go back to the discrete two-domain formulation, that is the discrete counterpart of (4), which involves the extension operators. The discrete extension operators are chosen such that they coincide with the classical shape functions of the interface nodes, as illustrated in Figure 15. The solution on the interface is duplicated, and noted $\mathbf{u}_{\Gamma_1}$ and $\mathbf{u}_{\Gamma_2}$, whether it belongs to Ω_1 or Ω_2, respectively. Therefore, the algebraic equation coming from the integration and assembly of the four equations of System (4) reads:

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{1\Gamma} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{2\Gamma} \\ \mathbf{A}_{\Gamma 1} & \mathbf{A}^{(1)}_{\Gamma\Gamma} & \mathbf{A}_{\Gamma 2} & \mathbf{A}^{(2)}_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_{\Gamma_1} \\ \mathbf{u}_2 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{0} \\ \mathbf{b}_2 \\ \mathbf{b}^{(1)}_\Gamma + \mathbf{b}^{(2)}_\Gamma \end{pmatrix}. \tag{13}$$

By eliminating $\mathbf{u}_{\Gamma_1}$ using the second equation and setting $\mathbf{u}_{\Gamma_2} = \mathbf{u}_\Gamma$, we can easily check that this system is equivalent to System (12). Note that another choice for the extension operators would lead to a different algebraic system.

By rearranging the terms of the last equations in (13), we obtain the following equivalent system:

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{1\Gamma} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & -\mathbf{I} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{2\Gamma} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{\Gamma 2} & \mathbf{A}_{\Gamma\Gamma}^{(2)} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_{\Gamma_1} \\ \mathbf{u}_2 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{0} \\ \mathbf{b}_2 \\ \mathbf{b}_\Gamma^{(2)} + \mathbf{r}_\Gamma^{(1)} \end{pmatrix}, \tag{14}$$

where we have defined the residual $\mathbf{r}_\Gamma^{(1)}$ computed in Ω_1 on Γ as

$$\mathbf{r}_\Gamma^{(1)} = \mathbf{b}_\Gamma^{(1)} - \mathbf{A}_{\Gamma 1}\mathbf{u}_1 - \mathbf{A}_{\Gamma\Gamma}^{(1)}\mathbf{u}_{\Gamma_1}. \tag{15}$$

We can now clearly draw the analogy between this system and the discrete Dirichlet/Neumann method given by System (6). In Equation $(6)_3$, the Neumann condition on the flux “corresponds” to the residual of the equation on the interface $\mathbf{r}_\Gamma^{(1)}$ appearing in last equation of System (14). The equivalence between the algebraic formulations are illustrated in Figure 12 (Bottom).

Dirichlet/Neumann

In order to express the Dirichlet/Neumann formulation in the context of multi-code coupling, and to treat non-matching and overlapping meshes using the same notation, we duplicate the interface obtaining two copies Γ_1 and Γ_2, respectively. We also introduce some new submatrices such that for this specific case we have: $\mathbf{A}_{1\Gamma_1} = \mathbf{A}_{1\Gamma}$, $\mathbf{A}_{2\Gamma_2} = \mathbf{A}_{2\Gamma}$, $\mathbf{A}_{\Gamma_1 1} = \mathbf{A}_{\Gamma 1}$, $\mathbf{A}_{\Gamma_2 2} = \mathbf{A}_{\Gamma 2}$, $\mathbf{A}_{\Gamma_1\Gamma_1} = \mathbf{A}_{\Gamma\Gamma}^{(1)}$, $\mathbf{A}_{\Gamma_2\Gamma_2} = \mathbf{A}_{\Gamma\Gamma}^{(2)}$, $\mathbf{b}_{\Gamma_1} = \mathbf{b}_\Gamma^{(1)}$ and $\mathbf{b}_{\Gamma_2} = \mathbf{b}_\Gamma^{(2)}$. Figure 16 shows the nomenclature used throughout this section. Using this new notation, we can easily check that System (14) can be equivalently written as:

Two-subdomain algebraic formulation - Dirichlet/Neumann coupling

$$\begin{gathered} \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{1\Gamma_1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_{\Gamma_1} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix} \\ \begin{pmatrix} \mathbf{A}_{22} & \mathbf{A}_{2\Gamma_2} \\ \mathbf{A}_{\Gamma_2 2} & \mathbf{A}_{\Gamma_2\Gamma_2} \end{pmatrix} \begin{pmatrix} \mathbf{u}_2 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_2 \\ \mathbf{b}_{\Gamma_2} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{r}_{\Gamma_1} \end{pmatrix} \end{gathered} \tag{16}$$

where the interface residual $\mathbf{r}_{\Gamma_1}$ evaluated on Γ_1 is defined as

$$\mathbf{r}_{\Gamma_1} = \mathbf{b}_{\Gamma_1} - \mathbf{A}_{\Gamma_1 1}\mathbf{u}_1 - \mathbf{A}_{\Gamma_1\Gamma_1}\mathbf{u}_{\Gamma_1}.$$

The first system is the equation for subdomain 1 together with a Dirichlet condition coming from the solution on the interface of subdomain 2, $\mathbf{u}_{\Gamma_2}$. The

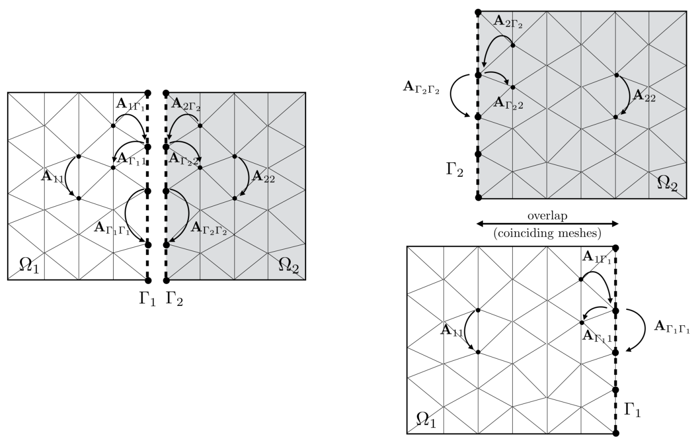


Fig. 16: Two-subdomain coupling, nomenclature. (Left) Disjoint subdomains. (Right) Overlapping subdomains.

second system is the equation for subdomain 2 together with an algebraic Neumann condition coming from subdomain 1, that is $\mathbf{b}_{\Gamma_1} - \mathbf{A}_{\Gamma_1 1}\mathbf{u}_1 - \mathbf{A}_{\Gamma_1\Gamma_1}\mathbf{u}_{\Gamma_1}$, which is the residual of the interface node equation of subdomain 1.

By duplicating the interface nodes and rearranging the terms of the equation, we have thus derived a two-domain formulation equivalent to the original problem. For the moment, the problems are strongly coupled; in next section we will now see how to build partitioned strategies to solve the two problems independently, and how to couple them both explicitly and implicitly.

Dirichlet/Robin

In Section 3.4.2, we commented that by integrating by parts the advection term, one ends up with a natural condition of Robin type, as given by Equation (9). This means that at the algebraic level, System (16) would naturally consists of a Dirichlet/Robin coupling, corresponding to Equation (8) at the discrete level.

However, the Robin condition can be selected such that it does not necessarily coincide with the natural condition. At the algebraic level, this corresponds to penalizing the Neumann condition by the continuity of the unknown. The solution remains unchanged but the convergence properties of the method, in the context of an iteration-by-subdomain method can be greatly affected. The formulation reads:

Two-subdomain algebraic formulation - Dirichlet/Robin coupling

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{1\Gamma_1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_{\Gamma_1} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{A}_{22} & \mathbf{A}_{2\Gamma_2} \\ \mathbf{A}_{\Gamma_2 2} & \mathbf{A}_{\Gamma_2\Gamma_2} \end{pmatrix} \begin{pmatrix} \mathbf{u}_2 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_2 \\ \mathbf{b}_{\Gamma_2} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{r}_{\Gamma_1} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{D}_2(\mathbf{u}_{\Gamma_1} - \mathbf{u}_{\Gamma_2}) \end{pmatrix}$$

where $\mathbf{D}_2$ is a matrix, preferably diagonal, which helps to enforce the Dirichlet condition. With reference to the weak formulation (7) using $\flat = 0$, this corresponds to the case $\mathbf{D}_2 = -\alpha_2 \mathbf{M}_{\Gamma_2\Gamma_2}$ where $\mathbf{M}_{\Gamma_2\Gamma_2}$ is the mass matrix on Γ_2. Let us finally note that by taking $\mathbf{D}_2 = \frac{1}{\epsilon}\mathbf{I}$ with $\epsilon \to 0$, a Dirichlet condition would be automatically imposed on Γ_2.

Robin/Robin

The Robin/Robin coupling is the most general one. It reads:

Two-subdomain algebraic formulation - Robin/Robin coupling

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{1\Gamma_1} \\ \mathbf{A}_{\Gamma_1 1} & \mathbf{A}_{\Gamma_1\Gamma_1} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_{\Gamma_1} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_{\Gamma_1} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{r}_{\Gamma_2} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{D}_1(\mathbf{u}_{\Gamma_2} - \mathbf{u}_{\Gamma_1}) \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{A}_{22} & \mathbf{A}_{2\Gamma_2} \\ \mathbf{A}_{\Gamma_2 2} & \mathbf{A}_{\Gamma_2\Gamma_2} \end{pmatrix} \begin{pmatrix} \mathbf{u}_2 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_2 \\ \mathbf{b}_{\Gamma_2} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{r}_{\Gamma_1} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{D}_2(\mathbf{u}_{\Gamma_1} - \mathbf{u}_{\Gamma_2}) \end{pmatrix}$$

We note that subtracting the second and fourth equation, we end up with $(\mathbf{D}_1 + \mathbf{D}_2)(\mathbf{u}_{\Gamma_1} - \mathbf{u}_{\Gamma_2}) = \mathbf{0}$, which gives $\mathbf{u}_{\Gamma_1} = \mathbf{u}_{\Gamma_2}$ whenever $(\mathbf{D}_1 + \mathbf{D}_2)$ is invertible. We can then easily show that this system is equivalent to the one-domain formulation (12).

By playing around with matrices $\mathbf{D}_1$ and $\mathbf{D}_2$, one can recover Dirichlet or Neumann conditions, even locally. This is the starting formulation to derive algebraic-based *adaptive iteration-by-subdomain methods*. See for example [19, 23, 41]. Assume $\mathbf{D}_i$ is a diagonal matrix for $i = 1, 2$, and let $D_{i,k}$, $\boldsymbol{a}_k$ and $\boldsymbol{n}_k$ be the node k diagonal coefficient of $\mathbf{D}_i$, advection nodal vector $\mathbf{a}$ and outward normal vector $\mathbf{n}$ coefficients, respectively. Letting $\epsilon \to 0$, and choosing

$$D_{i,k} = \begin{cases} \frac{1}{\epsilon} & \text{if } \boldsymbol{a}_k \cdot \boldsymbol{n}_k < 0 \quad \to \quad \text{Dirichlet}, \\ 0 & \text{if } \boldsymbol{a}_k \cdot \boldsymbol{n}_k \geq 0 \quad \to \quad \text{Neumann}, \end{cases}$$

we are therefore able to design an adaptive Dirichlet/Neumann method based on the local nodal character of the flow (inflow/outflow), in the general context of the Robin/Robin formulation.

Dirichlet/Dirichlet

Now we consider the case of overlapping subdomains. The previous methods (Dirichlet/Neumann, Dirichlet/Robin and Robin/Robin) cannot be directly applied as they are. In fact, at convergence, the residual $\mathbf{r}_{\Gamma_1}$ is null on Γ_2! The reason is that the residual used as a Neumann condition should only contain subdomain 1 contributions from the outer part of the boundary Γ_2. In fact, referring to Equation (4) applied to the overlapping case, the extension is from the boundary towards the part of subdomain 1 which does not overlap with subdomain 2 [51]. Therefore, the algebraic formulation for these aforementioned coupling methods is not straightfoward, and we will only focus on the Dirichlet/Dirichlet coupling.

The Dirichlet/Dirichlet method, commonly referred to as the Schwarz method, is probably the most extensively commented and studied DDM in the literature [59]. It is usually implemented as an explicit coupling *a la Jacobi*, and usually used as a preconditioner to accelerate the convergence of Krylov methods. The Dirichlet/Dirichlet method will be studied in the context of an implicit coupling in Section 4.1.4, as it can be used for Chimera type applications, where there exists an overlap between the background and patch meshes. The method reads:

Two-subdomain algebraic formulation - Dirichlet/Dirichlet coupling

$$\begin{aligned} \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{1\Gamma_1} \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_{\Gamma_1} \end{pmatrix} &= \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{u}_2|_{\Gamma_1} \end{pmatrix} \\ \begin{pmatrix} \mathbf{A}_{22} & \mathbf{A}_{2\Gamma_2} \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{u}_2 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix} &= \begin{pmatrix} \mathbf{b}_2 \\ \mathbf{u}_1|_{\Gamma_2} \end{pmatrix} \end{aligned} \tag{17}$$

where $\mathbf{u}_i|_{\Gamma_j}$ means the restriction of the vector $\mathbf{u}_i$ on the interface Γ_j (remember that the nodes coincide in the overlapping zone).

4.1.2 Explicit coupling

The so-called *iteration-by-subdomain* methods solve System (16) in a decoupled manner. Let us introduce an iteration superindex k, and assume initial conditions $\mathbf{u}^0_{\Gamma_1}$ and $\mathbf{u}^0_{\Gamma_2}$. The iteration-by-subdomain method consists in solving the following two systems for $k = 1, 2, \ldots$ until convergence:

Iteration-by-subdomain Algebraic Dirichlet/Neumann

$$\begin{aligned} \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{1\Gamma_1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1^{k+1} \\ \mathbf{u}_{\Gamma_1}^{k+1} \end{pmatrix} &= \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{u}_{\Gamma_2}^{k} \end{pmatrix} \\ \begin{pmatrix} \mathbf{A}_{22} & \mathbf{A}_{2\Gamma_2} \\ \mathbf{A}_{\Gamma_2 2} & \mathbf{A}_{\Gamma_2\Gamma_2} \end{pmatrix} \begin{pmatrix} \mathbf{u}_2^{k+1} \\ \mathbf{u}_{\Gamma_2}^{k+1} \end{pmatrix} &= \begin{pmatrix} \mathbf{b}_2 \\ \mathbf{b}_{\Gamma_2} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{b}_{\Gamma_1} - \mathbf{A}_{\Gamma_1 1}\mathbf{u}_1^{k+m} - \mathbf{A}_{\Gamma_1\Gamma_1}\mathbf{u}_{\Gamma_1}^{k+m} \end{pmatrix} \end{aligned} \tag{18}$$

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

$$m = \begin{cases} 0: & \text{Parallel } \textit{or} \text{ Jacobi coupling} \\ 1: & \text{Sequential } \textit{or} \text{ Gauss-Seidel coupling} \end{cases}$$

In the first case, $m = 0$, the method is said to be parallel because both problems can be solved at the same time for $k+1$, the Dirichlet and residual conditions being computed at the previous iteration k. In addition, note that we can rewrite Equation (13) by introducing the following block matrix:

$$\left(\begin{array}{cc|cc} \mathbf{A}_{11} & \mathbf{A}_{1\Gamma_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & -\mathbf{I} \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{2\Gamma_2} \\ \mathbf{A}_{\Gamma_1 1} & \mathbf{A}_{\Gamma_1\Gamma_1} & \mathbf{A}_{\Gamma_2 2} & \mathbf{A}_{\Gamma_2\Gamma_2} \end{array}\right) \left(\begin{array}{c} \mathbf{u}_1 \\ \mathbf{u}_{\Gamma_1} \\ \hline \mathbf{u}_2 \\ \mathbf{u}_{\Gamma_2} \end{array}\right) = \left(\begin{array}{c} \mathbf{b}_1 \\ \mathbf{0} \\ \hline \mathbf{b}_2 \\ \mathbf{b}_{\Gamma_1} + \mathbf{b}_{\Gamma_2} \end{array}\right).$$

We can easily check that applying a block Jacobi method, we recover (18) 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. We will treat convergence issues at the end of this subsubsection, by introducing a relaxation parameter.

Similar iteration-by-subdomain methods can be designed for the Dirichlet/Robin, Robin/Robin and Dirichlet/Dirichlet methods presented previously. The properties of such methods have been extensively studied in the literature.

Relation with Schur complement

Let us go back to the monolithic formulation given by Equation (13), and eliminate the unknowns $\mathbf{u}_1$ and $\mathbf{u}_2$ to obtain the interface Schur complement system for $\mathbf{u}_\Gamma = \mathbf{u}_{\Gamma_1} = \mathbf{u}_{\Gamma_2}$:

$$\mathbf{S}\mathbf{u}_\Gamma = \mathbf{b}_S, \tag{19}$$

with

$$\begin{aligned} \mathbf{S} &= \mathbf{S}_1 + \mathbf{S}_2, \quad \mathbf{b}_S = \mathbf{b}_{S_1} + \mathbf{b}_{S_2}, \\ \mathbf{S}_i &= \mathbf{A}_{\Gamma_i\Gamma_i} - \mathbf{A}_{\Gamma_i i}\mathbf{A}_{ii}^{-1}\mathbf{A}_{i\Gamma_i}, \\ \mathbf{b}_{S_i} &= \mathbf{b}_{\Gamma_i} - \mathbf{A}_{\Gamma_i i}\mathbf{A}_{ii}^{-1}\mathbf{b}_i. \end{aligned} \tag{21}$$

for $i = 1, 2$. Now let us see how the method (18) with $m = 1$ (Gauss-Seidel) relates with the solution of the interface Schur complement system. We observe that given an interface value $\mathbf{u}_{\Gamma_2}^k$ in System (18), we generate with the second system a new interface unknown $\mathbf{u}_{\Gamma_2}^{k+1}$. We have $\mathbf{u}_{\Gamma}^{k} = \mathbf{u}_{\Gamma_2}^{k}$ and $\mathbf{u}_{\Gamma}^{k+1} = \mathbf{u}_{\Gamma_2}^{k+1}$. From the first two equations of both systems, we obtain:

$$\begin{aligned}\mathbf{u}_1^{k+1} &= \mathbf{A}_{11}^{-1}(\mathbf{b}_1 - \mathbf{A}_{1\Gamma_1}\mathbf{u}_{\Gamma}^{k}),\\ \mathbf{u}_2^{k+1} &= \mathbf{A}_{22}^{-1}(\mathbf{b}_2 - \mathbf{A}_{2\Gamma_2}\mathbf{u}_{\Gamma}^{k+1}).\end{aligned}$$

Now substituting these two expressions into the last equation of System (18), we get:

$$\mathbf{S}_2\mathbf{u}_{\Gamma}^{k+1} = \mathbf{b}_S - \mathbf{S}_1\mathbf{u}_{\Gamma}^{k}.$$

By adding $(\mathbf{S}_2\mathbf{u}_{\Gamma}^{k} - \mathbf{S}_2\mathbf{u}_{\Gamma}^{k})$ to the right hand side, we finally obtain the following equation for the interface unknown:

Dirichlet/Neumann: interface Schur complement

$$\mathbf{u}_{\Gamma}^{k+1} = \mathbf{u}_{\Gamma}^{k} + \mathbf{S}_2^{-1}(\mathbf{b}_S - \mathbf{S}\mathbf{u}_{\Gamma}^{k}) \tag{22}$$

This equation is a Richardson (or simple) iteration preconditioned by $\mathbf{S}_2^{-1}$ to solve Equation (19). Therefore, solving subdomain 1 with a Dirichlet condition, and subdomain 2 with a Neumann condition to obtain $\mathbf{u}_{\Gamma}^{k+1}$ from $\mathbf{u}_{\Gamma}^{k}$, is equivalent to solving a preconditioned Richardson iteration for the interface unknown $\mathbf{u}_{\Gamma}^{k+1}$, using $\mathbf{S}_2^{-1}$ as a preconditioner. Similarly, we can show that swapping the Dirichlet and Neumann conditions, the Gauss-Seidel method would be equivalent to solving a Richardson iteration preconditioned by $\mathbf{S}_1^{-1}$.

Convergence issues

The convergence of iteration-by-subdomain methods, whether they are implemented in the equation-based or algebraic-based context, is an issue. We have just shown how these methods are equivalent to the solution of the interface variable Schur complement by a preconditioned Richardson iteration. In order to accelerate the convergence, we have mainly three ways:

– Rely on an ad-hoc strategy by introducing a relaxation parameter α in the Schur complement equation:

$$\mathbf{u}_{\Gamma}^{k+1} = \mathbf{u}_{\Gamma}^{k} + \alpha\mathbf{S}_2^{-1}(\mathbf{b}_S - \mathbf{S}\mathbf{u}_{\Gamma}^{k}). \tag{23}$$

This is equivalent to relaxing the interface unknown after the Neumann step as follows:

$$\begin{aligned}\mathbf{u}_{\Gamma}^{k+1,*} &= \mathbf{u}_{\Gamma}^{k} + \mathbf{S}_2^{-1}(\mathbf{b}_S - \mathbf{S}\mathbf{u}_{\Gamma}^{k}),\\ \mathbf{u}_{\Gamma}^{k+1} &= \alpha\mathbf{u}_{\Gamma}^{k+1,*} + (1-\alpha)\mathbf{u}_{\Gamma}^{k}.\end{aligned}$$

– Consider other preconditioners than $\mathbf{S}_2^{-1}$. For example, the choice $\sigma_1\mathbf{S}_1^{-1} + \sigma_2\mathbf{S}_2^{-1}$ with σ_1 and σ_2 being two positive constants, leads to the so-called Neumann/Neumann method [16] (see also the remark at the end of this section);
– Consider efficient solvers rather than the simple Richardson method to solve for the Schur complement (22): CG, GMRES, BiCGSTAB, etc.

On the one hand, the introduction of a relaxation parameter is easy to implement, but the optimum α is highly problem-dependent, and we cannot guarantee convergence. On the other hand, introducing a more complex preconditioner or solver could not be a trivial task in the context of a multi-code coupling without involving some rewriting.

4.1.3 Solving the Schur complement system in a multi-code context

Let us investigate how one could work on the solution of the Schur complement (19), without considering explicitly the solution of the interface unknowns. In iterative solvers, mainly three operations are required: scalar product $\mathbf{x}_\Gamma \cdot \mathbf{y}_\Gamma$, matrix-vector product $\mathbf{S}\mathbf{x}_\Gamma$, and preconditioning which consists in solving a system of the form $\mathbf{S}_2\mathbf{x}_\Gamma = \mathbf{y}_\Gamma$. The scalar product can be carried out without any difficulty as interface vectors are available in both subdomains. Let us examine the action of $\mathbf{S} = \mathbf{S}_1 + \mathbf{S}_2$ on an interface vector $\mathbf{x}_\Gamma$. We first note that the action of $\mathbf{S}_1$ and $\mathbf{S}_2$ can be considered independently. For generic vectors $\mathbf{x}_\Gamma$ and $\mathbf{y}_\Gamma$ we can decompose each product into two steps:

Solving the Schur complement problem with an iterative solver:
Matrix-vector product $\mathbf{y}_\Gamma = \mathbf{S}\mathbf{x}_\Gamma$
for generic vectors $\mathbf{x}_\Gamma$ *and* $\mathbf{y}_\Gamma$

Solve Dirichlet problem in Ω_i: $$\mathbf{A}_{ii}\mathbf{x}_i = -\mathbf{A}_{i\Gamma_i}\mathbf{x}_\Gamma \quad \forall i \tag{24}$$

Compute: $$\mathbf{S}_i\mathbf{x}_\Gamma = \mathbf{A}_{\Gamma_i\Gamma_i}\mathbf{x}_\Gamma + \mathbf{A}_{\Gamma_i i}\mathbf{x}_i \quad \forall i \tag{25}$$

Add: $$\mathbf{y}_\Gamma = \mathbf{S}_1\mathbf{x}_\Gamma + \mathbf{S}_2\mathbf{x}_\Gamma \tag{26}$$

Therefore, applying $\mathbf{S}_i$ to an interface vector $\mathbf{x}_\Gamma$ consists in: first, solving a system in each subdomain using $\mathbf{x}_\Gamma$ as a Dirichlet boundary condition on the interface and zero elsewhere on the boundary, as the term $\mathbf{A}_{i\Gamma_i}\mathbf{x}_\Gamma$ is the only contribution to the right-hand side of Equation (24); second, performing the matrix-vector product on the interface as given by Equation (25); third, summing up the different subdomain contributions, Equation (26). The first two operations can be carried out locally and independently in each subdomain.

Now, let us consider the the preconditioning step, which consists in computing $\mathbf{x}_\Gamma = \mathbf{S}_2^{-1}\mathbf{y}_\Gamma$, or, equivalently, in solving $\mathbf{S}_2\mathbf{x}_\Gamma = \mathbf{y}_\Gamma$. By doing some simple algebra, we end up with:

Solving the Schur complement problem with an iterative solver:
Preconditioning $\mathbf{x}_\Gamma = \mathbf{S}_2^{-1}\mathbf{y}_\Gamma$

$$\text{Solve Neumann problem in } \Omega_2\text{: } \begin{pmatrix} \mathbf{A}_{22} & \mathbf{A}_{2\Gamma_2} \\ \mathbf{A}_{\Gamma_2 2} & \mathbf{A}_{\Gamma_2\Gamma_2} \end{pmatrix} \begin{pmatrix} \mathbf{x}_2 \\ \mathbf{x}_\Gamma \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{y}_\Gamma \end{pmatrix} \quad (27)$$

for generic vectors $\mathbf{x}_\Gamma$ and $\mathbf{y}_\Gamma$. Therefore, the preconditioning step $\mathbf{x}_\Gamma = \mathbf{S}_2^{-1}\mathbf{y}_\Gamma$ consists in solving a Neumann problem in Ω_2 using $\mathbf{y}_\Gamma$ as the Neumann data.

We have shown that the Schur complement system for the interface unknown can be solved using an iterative solver in a multi-code context, where only interface data of Dirichlet and Neumann type are required. The basic operations, matrix-vector product and preconditioning, involve the solution of Dirichlet and Neumann problems in the different subdomains, as given by Equations (24) and (27), respectively. However, efficient iterative solvers like GMRES involve lots of matrix-vector products to construct the Krylov space. Therefore, such methods can be very costly as at each matrix-vector product, very large systems in each subdomains may be solved in the subdomains. This is the main difference between DCM and DDM, as in DDM local matrices $\mathbf{A}_{ii}$ can be inverted exactly, at a reasonable cost.

There exist alternative ways of solving the interface problem. For example, in [65], both Dirichlet and Neumann data are considered explicitly by introducing Lagrange multipliers representing the Neumann data $\mathbf{y}_\Gamma$. It can be shown that the following system

$$\begin{aligned} \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{1\Gamma_1} \\ \mathbf{A}_{\Gamma_1 1} & \mathbf{A}_{\Gamma_1\Gamma_1} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_{\Gamma_1} \end{pmatrix} &= \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_{\Gamma_1} + \mathbf{y}_{\Gamma_1} \end{pmatrix}, \\ \begin{pmatrix} \mathbf{A}_{22} & \mathbf{A}_{2\Gamma_2} \\ \mathbf{A}_{\Gamma_2 2} & \mathbf{A}_{\Gamma_2\Gamma_2} \end{pmatrix} \begin{pmatrix} \mathbf{u}_2 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix} &= \begin{pmatrix} \mathbf{b}_2 \\ \mathbf{b}_{\Gamma_2} + \mathbf{y}_{\Gamma_2} \end{pmatrix}, \end{aligned} \quad (28)$$

together with the following compatibility conditions

$$\begin{aligned} \mathbf{u}_{\Gamma_1} - \mathbf{u}_{\Gamma_2} &= 0, \\ \mathbf{y}_{\Gamma_1} + \mathbf{y}_{\Gamma_1} &= 0, \end{aligned}$$

is equivalent to the one-domain system. By setting $\mathbf{u}_\Gamma = \mathbf{u}_{\Gamma_1}$ and $\mathbf{y}_\Gamma = \mathbf{y}_{\Gamma_1}$, and by eliminating interior unknowns in Equation (28), we end up with the following equation for the interface unknowns:

$$\begin{pmatrix} \mathbf{S}_1 & -\mathbf{I} \\ \mathbf{S}_2 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{x}_\Gamma \\ \mathbf{y}_\Gamma \end{pmatrix} = \begin{pmatrix} \mathbf{b}_S^{(1)} \\ \mathbf{b}_S^{(2)} \end{pmatrix}.$$

We note that this system is equivalent to the original Schur complement system (by summing up the two rows). Just like this last one, last system can be solved using efficient Krylov solvers. We observe that each matrix-vector product involves the solution of Dirichlet problems in both subdomains, as the basic operations consists in computing $\mathbf{S}_1\mathbf{x}_\Gamma$ and $\mathbf{S}_2\mathbf{x}_\Gamma$.

Solving the Schur complement system with the Orthomin(1) solver

To finish with the acceleration of the Dirichlet/Neumann using the Schur complement system, and to get an insight on the importance of convergence, we will now briefly illustrate the impact of the solver on the convergence of the Schur complement system. We choose a simple solver which involves very few additional operations with respect to the Richardson method, namely the Orthomin(1) method. This method consists in determining the relaxation parameter in Equation (23) such that it minimizes the residual. By defining the Schur complement residual $\mathbf{r}^k := \mathbf{b}_S - \mathbf{S}\mathbf{u}_\Gamma^k$, using Equation (23), we have:

$$\mathbf{r}^{k+1} = \mathbf{r}^k - \alpha\mathbf{S}\mathbf{S}_2^{-1}\mathbf{r}^k.$$

Now, minimizing $\mathbf{r}^{k+1}$ with respect to α, we find that

$$\alpha = \frac{(\mathbf{r}^k, \mathbf{S}\mathbf{S}_2^{-1}\mathbf{r}^k)}{(\mathbf{S}\mathbf{S}_2^{-1}\mathbf{r}^k, \mathbf{S}\mathbf{S}_2^{-1}\mathbf{r}^k)}.$$

Let us now compare the convergence properties of the relaxed Richardson (Gauss-Seidel) and the Orthomin(1) algorithms, by considering the following manufactured problem:

$$\begin{aligned} \mathcal{L}(u) &= \mathcal{L}(u_e) && \text{in } \Omega = (0, L) \times (0, L), \\ u &= u_e && \text{on } \partial\Omega, \end{aligned}$$

with $L = 1$, $\mathcal{L}$ defined as in Equation (1), and $u_e = 2x+3y$. 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 considered different Péclet numbers $\mathrm{Pe} = \frac{|\boldsymbol{a}|L}{\varepsilon}$ by varying ε, and advection pointing either to the right or to the left: $\boldsymbol{a} = [1, 0]^t$ or $\boldsymbol{a} = [-1, 0]^t$. 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 Ω_1 and a relaxed Dirichlet condition on Ω_2, that is $\mathbf{u}_{\Gamma_2}^{k+1} = \alpha\mathbf{u}_{\Gamma_1}^{k+m} + (1-\alpha)\mathbf{u}_{\Gamma_2}^k$, where α is the relaxation factor.

Figure 17 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 relaxation parameters and advection. We can draw the following (well-known) conclusions:

- The Orthomin(1) is much more robust that the Richardson algorithm;

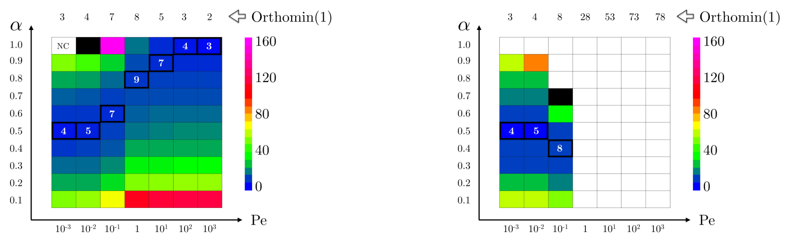


Fig. 17: Number of iterations to achieve convergence. (Left) $\boldsymbol{a} = [1, 0]^t$. (Right) $\boldsymbol{a} = [-1, 0]^t$. On the top, the number of iterations achieved by the Orthomin(1) algorithm. The figures indicate the optimum relaxation α for each Péclet number.

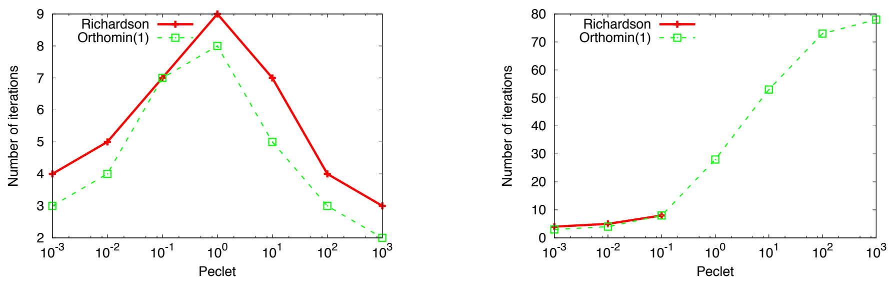


Fig. 18: Comparison of the Orthomin(1) method with the Richardson method using the optimum relaxation parameter. (Left) $\boldsymbol{a} = [1, 0]^t$. (Right) $\boldsymbol{a} = [-1, 0]^t$.

- The Orthomin(1) converges in less iterations than the Richardson method with optimum relaxation, as shown in Figure 18;
- The Orthomin(1) always converges, whatever the direction of the advection;
- The Richardson converges only for few low Péclet cases when the advection points towards the left.

Regarding this last comment, the fact that the type of optimal 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 [40, 19, 23, 41, 81, 24]. In addition, a Robin condition, which consists of a combination of Dirichlet and Neumann conditions can be useful as well to reach convergence faster.

4.1.4 Implicit coupling

A possible way to avoid iterating using Jacobi or Gauss-Seidel methods 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 $\mathbf{q} = \mathbf{A}\mathbf{p}$ and scalar products. The implicit method that we are presenting is thus only valid when iterative solvers are considered. At the end of the section we will show an example of implementation of the implicit method for the Dirichlet/Neumann and Dirichlet/Dirichlet couplings.

A matrix-vector product applied to the monolithic System (12) gives:

$$\begin{pmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \mathbf{q}_\Gamma \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{A}_{1\Gamma} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{2\Gamma} \\ \mathbf{A}_{\Gamma 1} & \mathbf{A}_{\Gamma 2} & \mathbf{A}_{\Gamma\Gamma}^{(1)} + \mathbf{A}_{\Gamma\Gamma}^{(2)} \end{pmatrix} \begin{pmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_\Gamma \end{pmatrix}. \tag{29}$$

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

$$\mathbf{A}^{(i)} = \begin{pmatrix} \mathbf{A}_{ii} & \mathbf{A}_{i\Gamma_i} \\ \mathbf{A}_{\Gamma_i i} & \mathbf{A}_{\Gamma_i\Gamma_i} \end{pmatrix}, \quad \mathbf{b}^{(i)} = \begin{pmatrix} \mathbf{b}_i \\ \mathbf{b}_{\Gamma_i} \end{pmatrix}, \quad \mathbf{u}^{(i)} = \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_{\Gamma_i} \end{pmatrix}.$$

Now consider the following uncoupled systems one would obtain by assembling them independently:

$$\mathbf{A}^{(1)}\mathbf{u}^{(1)} = \mathbf{b}^{(1)}, \quad \mathbf{A}^{(2)}\mathbf{u}^{(2)} = \mathbf{b}^{(2)}.$$

Dirichlet/Neumann(Robin)

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

$$\begin{pmatrix} \mathbf{q}_1 \\ \mathbf{q}_{\Gamma_1} \\ \mathbf{q}_2 \\ \mathbf{q}_{\Gamma_2} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{11}\mathbf{p}_1 + \mathbf{A}_{1\Gamma_1}\mathbf{p}_{\Gamma_1} \\ \mathbf{A}_{\Gamma_1 1}\mathbf{p}_1 + \mathbf{A}_{\Gamma_1\Gamma_1}\mathbf{p}_{\Gamma_1} \\ \mathbf{A}_{22}\mathbf{p}_2 + \mathbf{A}_{2\Gamma_2}\mathbf{p}_{\Gamma_2} \\ \mathbf{A}_{\Gamma_2 2}\mathbf{p}_2 + \mathbf{A}_{\Gamma_2\Gamma_2}\mathbf{p}_{\Gamma_2} \end{pmatrix}.$$

After this product, by performing the following operations:

$$\begin{cases} \text{Neumann:} & \mathbf{q}_{\Gamma_2} \Leftarrow \mathbf{q}_{\Gamma_2} + \mathbf{q}_{\Gamma_1}, \\ \text{Dirichlet:} & \mathbf{q}_{\Gamma_1} = \mathbf{q}_{\Gamma_2}, \end{cases} \tag{30}$$

we recover the same result as in Equation (29), with $\mathbf{q}_{\Gamma_1} = \mathbf{q}_{\Gamma_2} = \mathbf{q}_\Gamma$, whenever $\mathbf{p}_{\Gamma_1} = \mathbf{p}_{\Gamma_2} = \mathbf{p}_\Gamma$. In order to be able to solve the same problem $\mathbf{A}\mathbf{u} = \mathbf{b}$, one operation remains: the assembly of the right-hand side $\mathbf{b}$. We observe that by setting

$$\begin{cases} \text{Neumann:} & \mathbf{b}_{\Gamma_2} \Leftarrow \mathbf{b}_{\Gamma_2} + \mathbf{b}_{\Gamma_1}, \\ \text{Dirichlet:} & \mathbf{b}_{\Gamma_1} = \mathbf{b}_{\Gamma_2}, \end{cases} \tag{31}$$

we obtain $\mathbf{b}_{\Gamma_1} = \mathbf{b}_{\Gamma_2} = \mathbf{b}_{\Gamma}$. To refer to Equations (31)$_1$ and (30)$_1$, we will use the term *array assembly*, or *algebraic Neumann transmission condition*. The assignment, given by Equations (31)$_2$ and (30)$_2$ will be referred to *array substitution*, or *algebraic Dirichlet transmission condition*.

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 (31)$_1$ and (31)$_2$, respectively;
– After each matrix-vector product, assemble and substitute the local products, as given by Equations (30)$_1$ and (30)$_2$, respectively.

We stated *almost* exactly because there exists a very slight difference. When performing a scalar product, the contributions of interface nodes are accounted for twice, while in the case of the one-domain solution, they are accounted for 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).

Dirichlet/Dirichlet

Now let us consider the overlapping case. Using the same formalism as in the case of the implicit Dirichlet/Neumann, the Dirichlet/Dirichlet consists in performing the following operations after the matrix-vector product:

$$\begin{cases} \text{Dirichlet:} & \mathbf{q}_{\Gamma_1} = \mathbf{q}_2|_{\Gamma_1}, \\ \text{Dirichlet:} & \mathbf{q}_{\Gamma_2} = \mathbf{q}_1|_{\Gamma_2}. \end{cases} \tag{32}$$

By imposing these two conditions, we end-up with

$$\begin{pmatrix} \mathbf{q}_1 \\ \mathbf{q}_{\Gamma_1} \\ \mathbf{q}_2 \\ \mathbf{q}_{\Gamma_2} \end{pmatrix} = \begin{pmatrix} \mathbf{A}_{11}\mathbf{p}_1 + \mathbf{A}_{1\Gamma_1}\mathbf{p}_{\Gamma_1} \\ (\ \mathbf{A}_{22}\mathbf{p}_2 + \mathbf{A}_{2\Gamma_2}\mathbf{p}_{\Gamma_2}\ )|_{\Gamma_1} \\ \mathbf{A}_{22}\mathbf{p}_2 + \mathbf{A}_{2\Gamma_2}\mathbf{p}_{\Gamma_2} \\ (\ \mathbf{A}_{11}\mathbf{p}_1 + \mathbf{A}_{1\Gamma_1}\mathbf{p}_{\Gamma_1}\ )|_{\Gamma_2} \end{pmatrix}$$

We can check that provided $\mathbf{p}_1 = \mathbf{p}_2$, $\mathbf{p}_{\Gamma_1} = \mathbf{p}_2|_{\Gamma_1}$, $\mathbf{p}_{\Gamma_2} = \mathbf{p}_1|_{\Gamma_2}$ in the overlapping zone, we obtain the same result as in the monolithic case (29). In general, the last term of the second and forth equations vanishes when one has an overlap with more than one layer of elements.

As in the case of the Dirichlet/Neumann problem, the Dirichlet/Dirichlet coupling must be applied to the right-hand side of the equation as well:

$$\begin{aligned} \mathbf{b}_{\Gamma_1} &= \mathbf{b}_2|_{\Gamma_1}, \\ \mathbf{b}_{\Gamma_2} &= \mathbf{b}_1|_{\Gamma_2}. \end{aligned}$$

At the end of this subsubsection, we will study a one-dimensional example.

Dirichlet/Dirichlet restarted GMRES

The previous method is not exactly a Dirichlet/Dirichlet method in the Schwarz sense. The results of the matrix vector products are equalized but not the unknowns themselves. Another option consists in taking advantage of the structure of the restart GMRES method to exchange the Dirichlet condition on the unknown $\mathbf{x}$ when starting the outer loop of the GMRES method [74]. The method is summarized in Algorithm 1. The Dirichlet/Dirichlet coupling is imposed at step 2 of the algorithm.

Algorithm 1 Dirichlet/Dirichlet restarted GMRES preconditioned by $\mathbf{M}$

```
1: while Convergence not achieved do
2:     Apply Dirichlet/Dirichlet to $\mathbf{x}^0$
3:     Solve $\mathbf{M}\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0$
4:     Compute $\beta = \|\mathbf{r}^0\|_2$, $\mathbf{v}^1 = \mathbf{r}^0/\beta$
5:     for $j = 1, 2, \cdots, m$ do
6:         Solve $\mathbf{M}\mathbf{w} = \mathbf{A}\mathbf{v}^j$
7:         for $k = 1, 2, \cdots, j$ do
8:             $h_{k,j} = (\mathbf{w}, \mathbf{v}^k)$
9:             $\mathbf{w} = \mathbf{w} - h_{k,j}\mathbf{v}^k$
10:        end for
11:        $h_{j+1,j} = \|\mathbf{w}\|_2$, $\mathbf{v}^{j+1} = \mathbf{w}/h_{j+1,j}$
12:    end for
13:    Define $\mathbf{V}^m := [\mathbf{v}^1, \dots, \mathbf{v}^m]$, $\mathbf{H}^m = \{h_{i,j}\}_{1\le i\le j+1; 1\le j\le m}$
14:    Compute $\mathbf{y}^m = \mathrm{argmin}_y \|\beta\mathbf{e}_1 - \mathbf{H}\mathbf{y}\|_2$
15:    Update solution $\mathbf{x}^m = \mathbf{x}^0 + \mathbf{V}^m\mathbf{y}^m$
16:    Restart $\mathbf{x}^0 = \mathbf{x}^m$
17: end while
```

Example of implicit Dirichlet/Neumann and Dirichlet/Dirichlet

Let us solve a simple example to illustrate the implementation of the implicit algebraic Dirichlet/Neumann and Dirichlet/Dirichlet couplings presented previously. We consider the one-dimensional equation

$$-d^2u/dx^2 = 0 \quad \text{on } (0,6),$$

with $u(0) = 0$ and $u(6) = 6$. We consider the two subdomains and numberings illustrated in Figure 19 where the element length is 1. The exact solution is therefore $u = x$.

We are going to solve this problem using the Richardson solver preconditioned by the diagonal matrix. After eliminating the Dirichlet degrees of freedom $u_{D1} = 0$ and $u_{D2} = 6$, the uncoupled systems $\mathbf{Au} = \mathbf{b}$ for the Dirich-

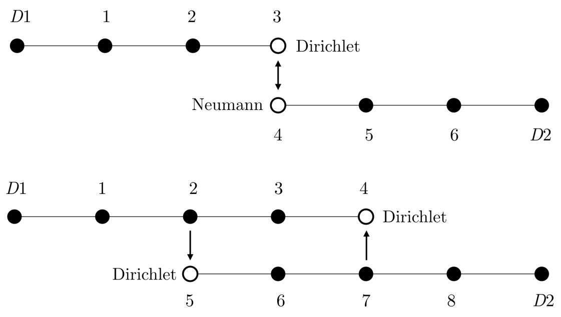


Fig. 19: One-dimensional problem as an illustration of the implicit algebraic methods. (Top) Dirichlet/Neumann. (Bottom) Dirichlet/Dirichlet.

let/Neumann and Dirichlet/Dirichlet couplings read, respectively:

$$\text{D/N system:} \quad \left(\begin{array}{ccc|ccc} 2 & -1 & 0 & & & \\ -1 & 2 & -1 & & \mathbf{0} & \\ 0 & -1 & 1 & & & \\ \hline & & & 1 & -1 & 0 \\ & \mathbf{0} & & -1 & 2 & -1 \\ & & & 0 & -1 & 2 \end{array}\right) \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 6 \end{pmatrix},$$

$$\text{D/D system:} \quad \left(\begin{array}{cccc|cccc} 2 & -1 & 0 & 0 & & & & \\ -1 & 2 & -1 & 0 & & \mathbf{0} & & \\ 0 & -1 & 2 & -1 & & & & \\ 0 & 0 & -1 & 1 & & & & \\ \hline & & & & 1 & -1 & 0 & 0 \\ & \mathbf{0} & & & -1 & 2 & -1 & 0 \\ & & & & 0 & -1 & 2 & -1 \\ & & & & 0 & 0 & -1 & 2 \end{array}\right) \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 6 \end{pmatrix}.$$

The Dirichlet/Neumann and Dirichlet/Dirichlet methods applied to the preconditioned Richardson algorithm are given by Algorithm 2. The domain composition couplings are applied in steps 3 and 6 of the algorithm. We can easily check that these two algorithms give exactly the same sequence $\mathbf{u}^k$ that would be obtained by the preconditioned Richardson method applied to the monolithic problem.

Throughout this section, we have derived implicit coupling methods to couple subdomains at the algebraic level. We pointed out that at each iteration of an iterative solver, the solution is the same as in the monolithic approach. The method is thus a *monolithic coupling*, whenever an iterative solver is considered.

Algorithm 2 Implicit Dirichlet/Neumann and Dirichlet/Dirichlet applied to the preconditioned Richardson method

```
1: Initial condition: $k = 0$, $\mathbf{u}^0$, equalize $\mathbf{u}^0$ on interface and overlapping zone
2: Diagonal: $\mathbf{d} = \mathrm{diag}(\mathbf{A})$
3: Apply transmission conditions on $\mathbf{d}$ and $\mathbf{b}$:
   $$\text{D/N:}\begin{cases} d_4 = d_4 + d_3 & \text{then } d_3 = d_4 \\ b_4 = b_4 + b_3 & \text{then } b_3 = b_4 \end{cases}$$
   $$\text{D/D:}\begin{cases} d_4 = d_7, & d_5 = d_2 \\ b_4 = b_7, & b_5 = b_2 \end{cases}$$
4: while Convergence not achieved do
5:     Compute: $\mathbf{q} = \mathbf{A}\mathbf{u}^k$
6:     Apply Domain Composition Method on $\mathbf{q}$:
       $$\text{D/N:}\left\{ q_4 = q_4 + q_3 \quad \text{then } q_3 = q_4 \right.$$
       $$\text{D/D:}\left\{ q_4 = q_7, \quad q_5 = q_2 \right.$$
7:     Residual: $\mathbf{r}^k = \mathbf{b} - \mathbf{q}$
8:     Solution update: $\mathbf{u}^{k+1} = \mathbf{u}^k + \mathbf{d}^{-1} \cdot \mathbf{r}^k$
9:     $k = k + 1$
10: end while
```

4.2 Non-matching meshes

The previous subsection dealt with matching meshes, for which the node-to-node correspondence between the subdomains enables an easy access to the coefficients of the vectors to exchange between the subdomains. The treatment of non-matching meshes complicates greatly the implementation of domain composition methods, both in terms of data structures and of parallelization issues. We refer the reader to [15] for a good review of some coupling techniques. We will start by introducing the transmission matrices for the Dirichlet and Neumann conditions. Then, we will present two possible ways of computing them, using either interpolation or projection schemes. We will also reinterpret the Dirichlet/Neumann coupling in terms of the interface Schur complement for non-matching meshes. Finally, we will end this subsection by describing the implicit schemes.

Let us introduce some terminology, as shown in Figure 20. 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, Γ_t is the target surface (also referred to as wet surface in the literature) and Γ_s is the source surface. In the two-subdomain context, either $t = 1$ and $s = 2$ or $t = 2$ and $s = 1$.

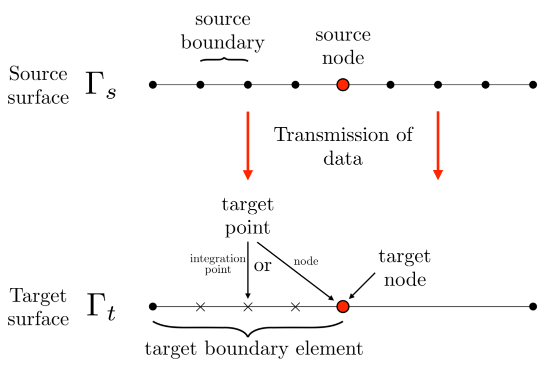


Fig. 20: Coupling terminology: source and target.

4.2.1 Transmission of data

For the sake of clarity, we will concentrate on the Dirichlet/Neumann formulation (16). Let us introduce the transmission matrices $\mathbf{T}^D$ and $\mathbf{T}^N$ which represent the Dirichlet and Neumann conditions, respectively. Equation (16) becomes:

$$\begin{aligned} \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{1\Gamma_1} \\ 0 & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_{\Gamma_1} \end{pmatrix} &= \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{T}^D \mathbf{u}_{\Gamma_2} \end{pmatrix}, \\ \begin{pmatrix} \mathbf{A}_{22} & \mathbf{A}_{2\Gamma_2} \\ \mathbf{A}_{\Gamma_2 2} & \mathbf{A}_{\Gamma_2\Gamma_2} \end{pmatrix} \begin{pmatrix} \mathbf{u}_2 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix} &= \begin{pmatrix} \mathbf{b}_2 \\ \mathbf{b}_{\Gamma_2} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{r}_{\Gamma_2} \end{pmatrix}, \end{aligned} \tag{33}$$

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

$$\mathbf{u}_{\Gamma_1} = \mathbf{T}^D \mathbf{u}_{\Gamma_2}, \quad \mathbf{r}_{\Gamma_2} = \mathbf{T}^N \mathbf{r}_{\Gamma_1},$$

with the residual $\mathbf{r}_{\Gamma_1}$ defined in Equation (15). Note that similar transmission matrices can be introduced for the other mixed methods and for the Dirichlet/Dirichlet method presented in System (17).

We therefore need to define both transmission matrices $\mathbf{T}^D$ and $\mathbf{T}^N$. Note that these matrices are rectangular and:

$$\begin{aligned} \text{size}(\mathbf{T}^D) &= \text{size}(\mathbf{u}_{\Gamma_1}) \times \text{size}(\mathbf{u}_{\Gamma_2}), \\ \text{size}(\mathbf{T}^N) &= \text{size}(\mathbf{r}_{\Gamma_2}) \times \text{size}(\mathbf{r}_{\Gamma_1}). \end{aligned}$$

To obtain these tramsission matrices, two main methods are available, as illustrated in Figure 21:

- Interpolation schemes;

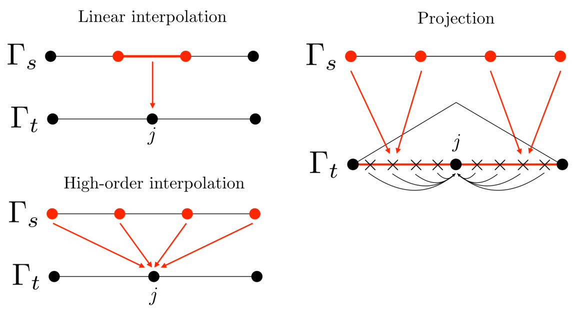


Fig. 21: Scheme to transfer data from the source $\mathit{\Gamma}_s$ to the target $\mathit{\Gamma}_t$.

– Projection schemes.

In the following, we are going to consider a generic interface unknown of subdomain i, noted by u_i at the continuous level, and by $\mathbf{u}_i$ at the algebraic level where the j^{th} component of $\mathbf{u}_i$, $u_{i,j}$, is the value of $\mathbf{u}_i$ at node j. In the previous context, u_i can therefore be the interface unknown or the residual. We will consider a generic transmission matrix $\mathbf{T}$ as well, which stands for either $\mathbf{T}^D$ or $\mathbf{T}^N$.

The transmission conditions can therefore be written:

$$\mathbf{u}_t = \mathbf{T}\mathbf{u}_s. \tag{34}$$

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

As an example, as far as the Dirichlet condition in Equation (33) is concerned, the target surface is the interface $\mathit{\Gamma}_1$ of subdomain 1, while the source solution is $\mathbf{u}_{\mathit{\Gamma}_2}$.

Interpolation schemes

Let us consider a target node i on $\mathit{\Gamma}_t$ with coordinates $\boldsymbol{x}_{t,i}$. We want to obtain a nodal value $u_{t,i}$ on the target from the known values on the source, $u_{s,j}$. The simplest interpolation scheme considers a linear interpolation [4], by first identifying the host boundary element of the target node in the source boundary so that $u_{s,i} = \sum_{j=1}^{n_s} N_{s,j}(\boldsymbol{x}_{t,i})u_{s,j}$, where $N_{s,j}$ are the shape functions on the source. Therefore,

Transmission matrix for linear interpolation

$$T_{ij} = N_{s,j}(\boldsymbol{x}_{t,i})$$

High-order interpolation schemes consider rather a cloud of nodes. Let us mention methods such as radial basis functions [5,15] or Krigging [28].

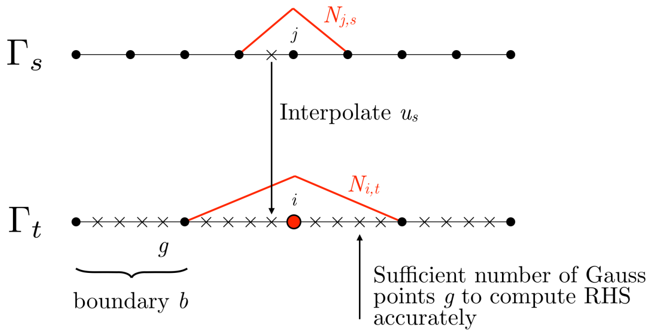


Fig. 22: Projection scheme: nomenclature.

Projection schemes

A projection method imposes continuity in a weak sense and it consists in minimizing $\|u_s - u_t\|^2_{L^2,\Gamma_t}$, that is in solving the following system:

$$\int_{\Gamma_t} u_t \, v_t \, d\Gamma = \int_{\Gamma_t} u_s \, v_t \, d\Gamma \quad \forall \, v_t \in V_t, \tag{35}$$

where V_t is the space of test functions on Γ_t. See Figure 22.

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 u_s as well. When computing the latter integral numerically, the source unknown u_s must be interpolated at the integration point of the target. By introducing the nodal unknowns on the target $u_{t,j}$, we obtain:

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

and therefore we end up with the following system:

$$\mathbf{M}_t \mathbf{u}_t = \mathbf{s}, \tag{36}$$

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 (36) involves the target boundary mass matrix. By choosing a close integration rule (where the Gauss points are located on the nodes) or by lumping the mass matrix, we obtain a diagonal mass matrix ($M_{t,ij} = \delta_{ij} M_{t,ii}$). As far as the right-hand side is concerned, let us start by introducing some notations (see Figure 22). Let $b = 1, \ldots, n_b$ denote the boundary elements on the target. On each boundary, we apply an integration rule with n_g integration points of coordinates $\boldsymbol{x}_{g,b}$ with weight w_g for $g = 1, \ldots, n_g$. It is important that the integration rule is chosen to accurately compute the integral. We have

$$s_i = \sum_{b=1}^{n_b} \sum_{g=1}^{n_g} \left(\sum_{j=1}^{n_s} N_{s,j}(\boldsymbol{x}_{g,b}) u_{s,j} \right) N_{t,i}(\boldsymbol{x}_{g,b}) w_g |J(\boldsymbol{x}_{g,b})|_b,$$

where $|J(\boldsymbol{x}_{g,b})|_b$ is the Jacobian of the boundary b computed at the integration point. Therefore, according to Equation (34), we have

Transmission matrix for L^2-projection

$$T_{ij} = \frac{1}{M_{t,ii}} \sum_{b=1}^{n_b} \sum_{g=1}^{n_g} N_{s,j}(\boldsymbol{x}_{g,b}) N_{t,i}(\boldsymbol{x}_{g,b}) w_g |J(\boldsymbol{x}_{g,b})|_b \tag{37}$$

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 Γ_s [54,15].

4.2.2 Explicit coupling for non-matching meshes

Explicit coupling is carried out just like in the matching mesh case, explained in Section 4.1.2. If we use the same terminology, the iteration-by-subdomain method consists in solving the following two systems for $k = 1, 2, \ldots$ until convergence:

Iteration-by-subdomain Algebraic Dirichlet/Neumann for non-matching meshes

$$\begin{aligned}
&\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{1\Gamma_1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1^{k+1} \\ \mathbf{u}_{\Gamma_1}^{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{T}^D \mathbf{u}_{\Gamma_2}^{k} \end{pmatrix} \\
&\begin{pmatrix} \mathbf{A}_{22} & \mathbf{A}_{2\Gamma_2} \\ \mathbf{A}_{\Gamma_2 2} & \mathbf{A}_{\Gamma_2\Gamma_2} \end{pmatrix} \begin{pmatrix} \mathbf{u}_2^{k+1} \\ \mathbf{u}_{\Gamma_2}^{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_2 \\ \mathbf{b}_{\Gamma_2} \end{pmatrix} \\
&\qquad + \begin{pmatrix} \mathbf{0} \\ \mathbf{T}^N (\mathbf{b}_{\Gamma_1} - \mathbf{A}_{\Gamma_1 1} \mathbf{u}_1^{k+m} - \mathbf{A}_{\Gamma_1\Gamma_1} \mathbf{u}_{\Gamma_1}^{k+m}) \end{pmatrix}
\end{aligned} \tag{38}$$

Similar algorithms can be deduced for the other mixed methods as well as for the Dirichlet/Dirichlet method applied to non-matching meshes.

Relation with Schur complement

Let us see how the formulation (38) can be reinterpreted in terms of the interface Schur complement solution, as we did in Section 4.1.2 for the matching case. Setting $\mathbf{u}_\Gamma = \mathbf{u}_{\Gamma_2}$ in (33), we have:

$$\begin{aligned}
\mathbf{u}_1 &= \mathbf{A}_{11}^{-1}(\mathbf{b}_1 - \mathbf{A}_{1\Gamma_1}\mathbf{T}^D\mathbf{u}_\Gamma),\\
\mathbf{u}_2 &= \mathbf{A}_{22}^{-1}(\mathbf{b}_2 - \mathbf{A}_{2\Gamma_2}\mathbf{u}_\Gamma).
\end{aligned}$$

Substituting these two equations into the second equation of System $(33)_2$, we end up with the following Schur complemnent system for $\mathbf{u}_{\Gamma_2}$:

$$\mathbf{S}\mathbf{u}_{\Gamma_2} = \mathbf{b}_S,$$

with

$$\mathbf{S} = \mathbf{T}^N\mathbf{S}_1\mathbf{T}^D + \mathbf{S}_2, \quad \mathbf{b}_S = \mathbf{T}^N\mathbf{b}_{S_1} + \mathbf{b}_{S_2},$$

where the matrices $\mathbf{b}_{S_i}$ and vectors $\mathbf{S}_i$, for $i = 1, 2$, are given by Equations (20) and (21).

We can show as well that the Gauss-Seidel method applied to System (38) for non-matching meshes, by selecting $m = 1$, is equivalent to solving:

$$\begin{aligned}
\mathbf{u}_{\Gamma_2}^{k+1} &= \mathbf{u}_{\Gamma_2}^k + \mathbf{S}_2^{-1}(\mathbf{b}_S - \mathbf{S}\mathbf{u}_{\Gamma_2}^k),\\
\mathbf{u}_{\Gamma_1}^{k+1} &= \mathbf{T}^D\mathbf{u}_{\Gamma_2}^k.
\end{aligned}$$

We can observe that, if the local Schur complement matrices $\mathbf{S}_1$ and $\mathbf{S}_2$ are symmetric, the total Schur complement matrix $\mathbf{S}$ is symmetric provided $\mathbf{T}^N = (\mathbf{T}^D)^t$.

4.2.3 Implicit coupling for non-matching meshes

We have explained previously how to implement an explicit Dirichlet/Neumann method at the algebraic level for non-matching meshes. The implicit method described in Section 4.1.4 for matching meshes can be generalized to non-matching grids by considering the transmission matrices in such a way that Equations $(31)_1$ and $(31)_2$ become:

$$\begin{cases}
\text{Neumann:} & \mathbf{q}_{\Gamma_2} \Leftarrow \mathbf{q}_{\Gamma_2} + \mathbf{T}^N\mathbf{q}_{\Gamma_1},\\
\text{Dirichlet:} & \mathbf{q}_{\Gamma_1} = \mathbf{T}^D\mathbf{q}_{\Gamma_2}.
\end{cases} \tag{40}$$

For the Dirichlet/Dirichlet method on non-overlapping grids, the counterparts of Equations $(32)_1$ and $(32)_2$ become:

$$\begin{cases}
\text{Dirichlet:} & \mathbf{q}_{\Gamma_1} = \mathbf{T}^{D_1}\mathbf{q}_2,\\
\text{Dirichlet:} & \mathbf{q}_{\Gamma_2} = \mathbf{T}^{D_2}\mathbf{q}_1,
\end{cases}$$

where we have introduced different Dirichlet transmission matrices for each Dirichlet condition.

Example: Dirichlet/Dirichlet for fluid mechanics

Let us go back to the Chimera example presented in Sections 2.1 and 2.2. We now solve the problem with the previously presented Dirichlet/Dirichlet algorithm. The original mesh is the same used for the HERMESH method of Section 2.2, which, before the hole cutting process, is composed of 156740 nodes and 496572 elements. After the hole cutting process, we end up with 650 hole elements in order to guarantee a one-element overlap on both sides of the interfaces. Some results are shown in Figure 23, where we can observe the good continuity of the solution across the interfaces, despite the large difference in mesh sizes between the background and patch. Figure 9 compares the velocity

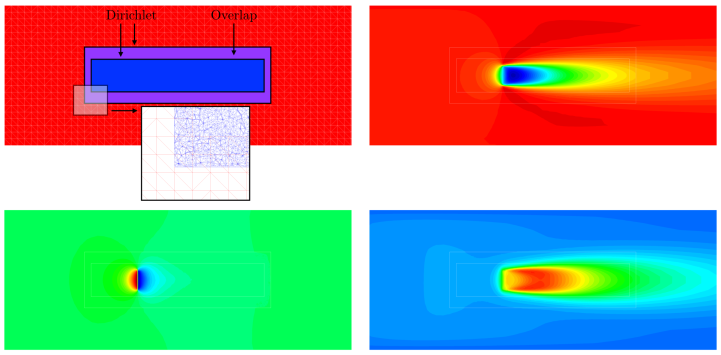


Fig. 23: Dirichlet/Dirichlet applied to Sexbierum example. Horizontal cut. (Top left) Background and patch, detail of the mesh. (Top right) Velocity module. (Bottom left) Pressure. (Bottom right) Turbulent kinetic energy.

deficit downstream of the wind turbine. We observe a very good agreement of the Dirichlet/Dirichlet method with the mesh-based formulations described in Section 2. Finally, let us comment on the mesh and algorithm convergence. Figure 24 (Left) compares the convergence of the GMRES method used to solve the momentum equations, considering both the Dirichlet/Dirichlet method and the one-domain approach. The one-domain solution was obtained by the conforming mesh approach described in Section 2.1. We observe that convergence is very similar. Finally, Figure 24 (Right) compares the residuals of the momentum equation. As in the previous case, convergence is also very similar.

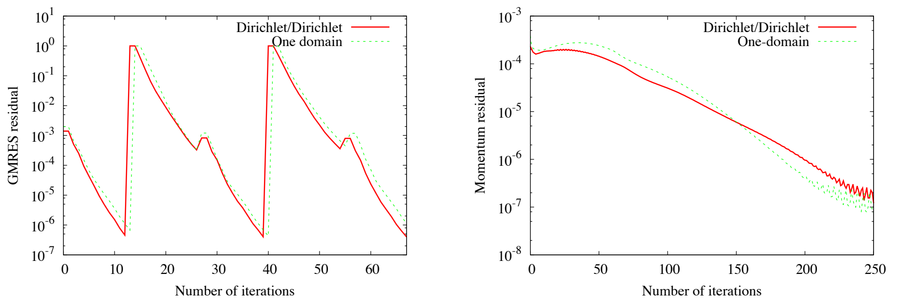


Fig. 24: Dirichlet/Dirichlet applied to Sexbierum example. (Left) Convergence of the GMRES method. (Right) Convergence of the momentum equation.

4.2.4 Resulting coupled algebraic system and symmetry preservation

Let us take a look at the resulting algorithm defined by Equation (33). To proceed, we eliminate $\mathbf{u}_{\Gamma_1}$ by using the second equation. After some algebraic manipulations, we end up with:

$$
\begin{pmatrix} \mathbf{A}_{11} & 0 & \mathbf{A}_{1\Gamma_1}\mathbf{T}^D \\ 0 & \mathbf{A}_{22} & \mathbf{A}_{2\Gamma_2} \\ \mathbf{T}^N\mathbf{A}_{\Gamma_1 1} & \mathbf{A}_{\Gamma_2 2} & \mathbf{A}_{\Gamma_2\Gamma_2}+\mathbf{T}^N\mathbf{A}_{\Gamma_1\Gamma_1}\mathbf{T}^D \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_{\Gamma_2}+\mathbf{T}^N\mathbf{b}_{\Gamma_1} \end{pmatrix} \tag{41}
$$

This sytem can be solved both explicitly or implicitly, as described in Sections 4.1.2 and 4.1.4, respectively.

The selected interpolation or projection scheme will influence not only the accuracy of the result but also the computational cost of the coupling. This last point is especially true in the case of the implicit method, where the algebraic transmission conditions are imposed after each matrix-vector product (Section 4.2.3). This point will be discussed in Section 4.4. Before commenting on the implicit case, let us raise an important issue concerning symmetry.

Let us consider the case of a symmetric problem, for which we have:

$$
\mathbf{A}_{11} = \mathbf{A}_{11}^t,\ \mathbf{A}_{22} = \mathbf{A}_{22}^t,\ \mathbf{A}_{\Gamma_1\Gamma_1} = \mathbf{A}_{\Gamma_1\Gamma_1}^t,\ \mathbf{A}_{\Gamma_2\Gamma_2} = \mathbf{A}_{\Gamma_2\Gamma_2}^t
$$
$$
\mathbf{A}_{\Gamma_1 1} = \mathbf{A}_{1\Gamma_1}^t,\ \mathbf{A}_{\Gamma_2 2} = \mathbf{A}_{2\Gamma_2}^t.
$$

If we are facing a symmetric problem such as the pressure equation, a pure diffusion equation, or even structure mechanics in some cases, we wish to end up with a coupled symmetric problem as well. We observe that in general, due to the presence of the transmission matrices, the coupled system is not symmetric even though the original independent systems are. If the coupled problems are solved explicitly and therefore independently, the transmission data are cast to the right-hand side of the system. Therefore, the lack of symmetry of the coupled system is not an issue. The problem arises for implicit coupling. This point will prevent us from using efficient solvers specially designed for symmetric matrices (CG, deflated CG, etc.).

A simple solution to this problem consists in selecting the transmission matrices such that $\mathbf{T}^D = \mathbf{T}$ and $\mathbf{T}^N = \mathbf{T}^t$. We can easily check that these transmission matrices preserve symmetry and the resulting counterpart of (41) is the symmetric system:

$$\begin{pmatrix} \mathbf{A}_{11} & 0 & \mathbf{A}_{1\Gamma_1}\mathbf{T} \\ 0 & \mathbf{A}_{22} & \mathbf{A}_{2\Gamma_2} \\ \mathbf{T}^t\mathbf{A}_{\Gamma_1 1} & \mathbf{A}_{\Gamma_2 2} & \mathbf{A}_{\Gamma_2\Gamma_2} + \mathbf{T}^t\mathbf{A}_{\Gamma_1\Gamma_1}\mathbf{T} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_{\Gamma_2} + \mathbf{T}^t\mathbf{b}_{\Gamma_1} \end{pmatrix}$$

We will see in Section 4.3.1 that in addition, these transpose transmission matrices enable one to conserve some global quantities across the interface.

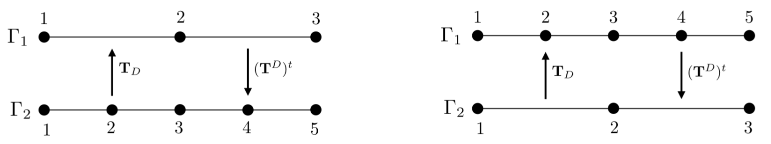
4.3 Conservation of physical quantities

In the context of DCM, conservation refers to the exact transmission, from one subdomain to another, of either local or global quantities. In fact, if the meshes of the subdomains do not match on the interface, a simple interpolation scheme to compute the Dirichlet and Neumann data may be neither locally nor globally conservative. While local conservation is usually treated using high order interpolation schemes, several techniques have been proposed to achieve global conservation in the literature. On the one hand, interpolation schemes are local operators and do not, in general, conserve any specific global quantity. On the other hand, by choosing properly the integral rule used to compute the right-hand side in projection schemes, we observe that taking $v_t = 1$ in Equation (35), we automatically conserve the integral of the unknown, and therefore the average value of it.

But what else can we conserve and how? We will now present four possible techniques to conserve global quantities.

4.3.1 Flux integral conservation via transposed transmission matrices

Some global quantities can be automatically conserved depending on the selected transmission matrices $\mathbf{T}^D$ and $\mathbf{T}^N$. In [4,15], it is shown that for Fluid-Structure interactions (FSI), if Ω_1 is the fluid subdomain and Ω_2 the solid



$$\begin{pmatrix} u_{1,1} \\ u_{1,2} \\ u_{1,3} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_{2,1} \\ u_{2,2} \\ u_{2,3} \\ u_{2,4} \\ u_{2,5} \end{pmatrix} \qquad \begin{pmatrix} u_{1,1} \\ u_{1,2} \\ u_{1,3} \\ u_{1,4} \\ u_{1,5} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ 0 & 1 & 0 \\ 0 & 1/2 & 1/2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_{2,1} \\ u_{2,2} \\ u_{2,3} \end{pmatrix}$$

$$\begin{pmatrix} r_{2,1} \\ r_{2,2} \\ r_{2,3} \\ r_{2,4} \\ r_{2,5} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{1,1} \\ r_{1,2} \\ r_{1,3} \end{pmatrix} \qquad \begin{pmatrix} r_{2,1} \\ r_{2,2} \\ r_{2,3} \end{pmatrix} = \begin{pmatrix} 1 & 1/2 & 0 & 0 & 0 \\ 0 & 1/2 & 1 & 1/2 & 0 \\ 0 & 0 & 0 & 1/2 & 1 \end{pmatrix} \begin{pmatrix} r_{1,1} \\ r_{1,2} \\ r_{1,3} \\ r_{1,4} \\ r_{1,5} \end{pmatrix}$$

$$\Downarrow$$

$$\sum_i r_{\Gamma_2,i} = \sum_i r_{\Gamma_1,i}$$

Fig. 25: Conservation of the total residual using $\mathbf{T}^N = (\mathbf{T}^D)^t$ for a simple example.

subdomain, then, by choosing the Neumann transmission matrix as the transposed of the Dirichlet one, i.e.,

$$\mathbf{T}^N = (\mathbf{T}^D)^t, \tag{42}$$

and such that $\sum_j T^D_{ij} = 1 \ \forall i$, 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 the reaction nodal vector satisfies $\sum_i r_{\Gamma_2,i} = \sum_i r_{\Gamma_1,i}$.

For example, linear interpolation and projection (if the right-hand side is accurately computed) for $\mathbf{T}^D$ both satisfy $\sum_j T^D_{ij} = 1 \ \forall i$, 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 across the interface. In the case of the Navier-Stokes equations, the conserved variable is the global traction, that is the force. See also [22], where force conservation is imposed via a projection method for FSI.

Figure 25 shows a simple example where the Neumann transmission matrix is given by Equation (42) and the Dirichlet transmission matrix is based on a simple linear interpolation. Two cases are considered: coarse and fine Dirichlet domains coupled with fine and coarse Neumann domains, respectively. We can easily check that $\sum_i r_{\Gamma_2,i} = \sum_i r_{\Gamma_1,i}$ is satisfied.

4.3.2 Conservation via constraint

Another methodology to conserve any kind of global quantity [49] consists in solving a constrained minimization problem of the form:

$$\begin{cases} \text{minimize} & \|\mathbf{u}_t - \mathbf{T}\mathbf{u}_s\|^2, \\ \text{under the constraint} & \mathbf{c}^t\mathbf{u}_t = c. \end{cases} \tag{43}$$

The idea of solving this problem is to look for the *nearest* solution to Equation (34) under a scalar constraint represented by Equation $(43)_2$, where $\mathbf{c}$ 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 [49] in order to conserve a zero mass across the interface when interpolating the velocity. In this case, if $\mathbf{u}_t$ is the velocity nodal vector, one has:

$$\text{Zero mass conservation:} \quad c_{2(i-1)+k} = \int_{\Gamma_t} N_{t,i} \, n_k \, d\Gamma, \quad c = 0, \quad \text{for } k = 1, 2,$$

where k indicates the dimension (velocity in x and y) and n_k is the k-th component of the outward normal vector $\boldsymbol{n}$.

The system (43) is solved by introducing the Lagrange multiplier λ of the constraint. The Lagrangian is given by

$$L(\mathbf{u}_t, \lambda) = \|\mathbf{u}_t - \mathbf{T}\mathbf{u}_s\|^2 - \lambda(\mathbf{c}^t\mathbf{u}_t - c).$$

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

$$\begin{bmatrix} \mathbf{I} & -\mathbf{c} \\ \mathbf{c}^t & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u}_t \\ \mu \end{bmatrix} = \begin{bmatrix} \mathbf{T}\mathbf{u}_s \\ c \end{bmatrix}. \tag{44}$$

By performing the operation $[\mathbf{c}^t \cdot (44)_1 - (44)_2]$, we obtain the multiplier μ. Then, by substituting this value into the first equation, we finally end up with:

$$\mathbf{u}_t = \mathbf{T}\mathbf{u}_s + \left(\frac{c - \mathbf{c}^t \, \mathbf{T}\mathbf{u}_s}{\mathbf{c}^t \, \mathbf{c}} \right) \mathbf{c}.$$

The first term is the unconstrained transmission operation $\mathbf{u}_t = \mathbf{T}\mathbf{u}_s$, 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 $\mathbf{T}\mathbf{u}_s$ already satisfies the constraint, the correction is zero, which makes the scheme consistent. Note finally that the correction is distributed in $\mathbf{u}_t$ according to vector $\mathbf{c}$, as given by the last term of the equation.

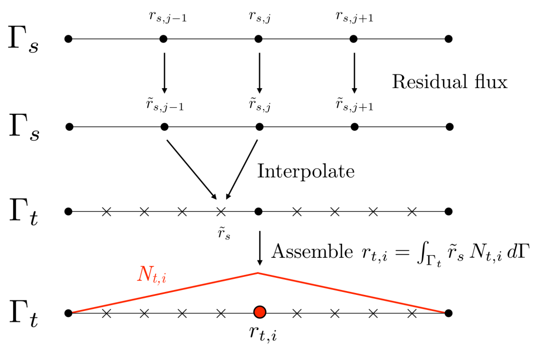


Fig. 26: Residual transmission matrix.

4.3.3 Conservation residual assembly via accurate projection

We already observed that the residual of the governing equation $\mathbf{r}_t$ (Equation (15)) 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 $\tilde{\mathbf{r}}$. Table 2 shows the physical correspondence of the residual $\mathbf{r}$ and residual flux for three different equations (the units depending on the scaling of the equation but we give here the units coming from the most common choices in MKS). The nodal residual on the target is thus assembled

Equation	Residual name [unit]	Residual flux name [unit]
Temperature	Heat power [W]	Heat flux [W/m^2]
Navier-Stokes	Force [N]	Traction [N/m^2]
Structure	Force [N]	Normal stress [N/m^2]

Table 2: Physical quantity represented by the residual $\mathbf{r}$ and the residual flux $\tilde{\mathbf{r}}$ for different equations.

from this residual flux as:

$$r_{t,i} = \int_{\Gamma_t} \tilde{r}_s N_{t,i} \, d\Gamma, \tag{45}$$

as illustrated in Figure 26.

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 $\mathbf{M}_s$ (which is diagonal) to scale the residual to obtain the residual flux as:

$$\tilde{r}_{s,i} = r_{s,i}/M_{s,ii}, \tag{46}$$

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

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

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:

Transmission matrix for total force conservation

$$T_{ij} = \frac{1}{M_{s,jj}} \sum_{b=1}^{n_b} \sum_{g=1}^{n_g} N_{s,j}(\boldsymbol{x}_{g,b}) N_{t,i}(\boldsymbol{x}_{g,b}) w_g |J(\boldsymbol{x}_{g,b})|_b \tag{47}$$

We observe that the difference with the projection transmission matrix given by Equation (37) 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 (45) together with Equation (46)? Let us compute the total residual (total heat power, total force, etc.):

$$\begin{aligned}
\text{Total residual} = \sum_{i=1}^{n_t} r_{t,i} &= \int_{\Gamma_t} \tilde{r}_s \left(\sum_{i=1}^{n_t} N_{t,i} \right) d\Gamma, \\
&= \sum_{j=1}^{n_s} \tilde{r}_{s,j} \int_{\Gamma_t} N_{s,j} \, d\Gamma, && \left(\textstyle\sum_{i=1}^{n_t} N_{t,i} = 1\right)\Big) \\
&= \sum_{j=1}^{n_s} \tilde{r}_{s,j} M_{s,jj} \, d\Gamma, && \text{(lumped mass matrix)} \\
&= \sum_{j=1}^{n_s} r_{s,j} \, d\Gamma. && \text{(Equation (46))}
\end{aligned}$$

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.

Example: Dirichlet/Neumann for solid mechanics

We consider here a beam under the action of gravity. As a reference solution, the problem is solved using a hybrid mesh of hexahedra and tetrahedra. The results of the displacement at a given time step are shown in Figure 27 (Left). Then we consider the Dirichlet/Neumann method using linear interpolation for the Dirichlet data and residual flux conservative assembly given by Equation (47), with a sufficiently accurate integration rule. Note that the mesh of subdomain 1 is twice coarser than the mesh of subdomain 2. The results of the Dirichlet/Neumann method are shown on Figure 27 (Right). Finally,

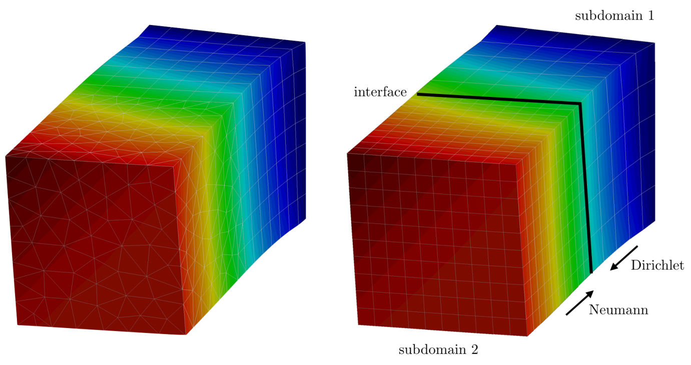


Fig. 27: Beam under gravity. (Left) One-domain. (Right) Dirichlet/Neumann using flux integral (total force) conservation.

Figure 28 compares the evolution of the displacement of one point for the two methods. We observe perfect agreement.

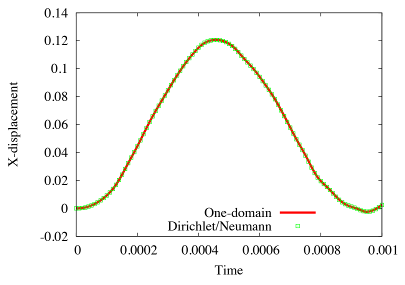


Fig. 28: Beam under gravity. Evolution of the displacement at one node.

Example: Dirichlet/Neumann for fluid mechanics

We consider the solution of the backward facing step shown in Figure 29. The one-domain solution is obtained on fine and coarse meshes, whose details are shown on the top of the figure, and compared to a two-domain coupling, whose

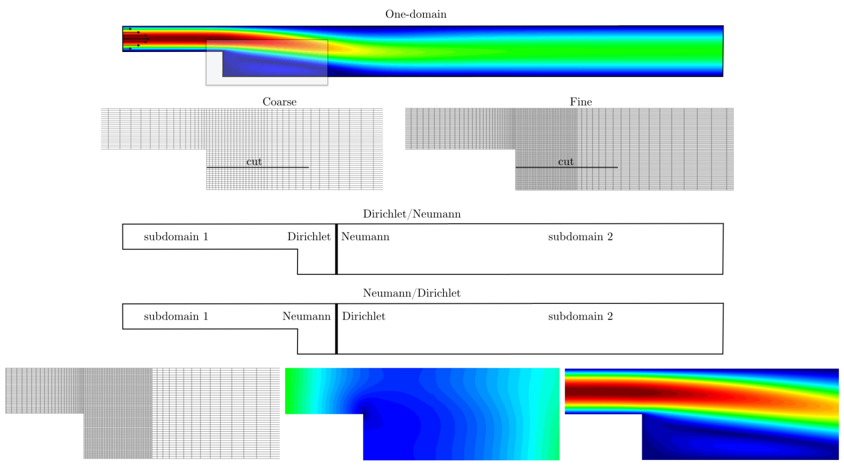


Fig. 29: Backward facing step. (Top) One-domain solution, fine and coarse mesh. (Bottom) Dirichlet/Neumann and Neumann/Dirichlet coupling.

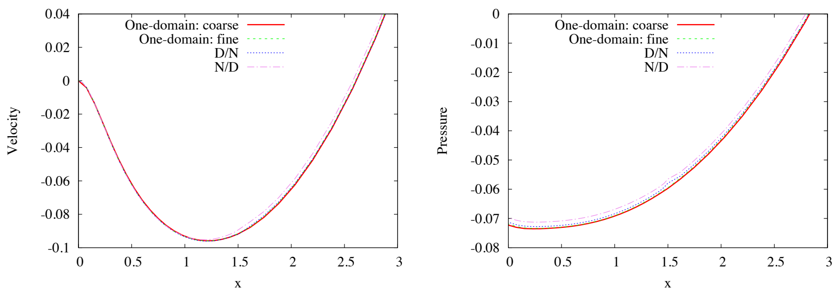


Fig. 30: Backward facing step. (Left) Velocity. (Right) Pressure.

meshes are shown on the bottom of the figure. Dirichlet and Neumann conditions are applied on the interfaces to obtain two different couplings, Dirichlet/Neumann and Neumann/Dirichlet. The Neumann condition is obtained by residual projection using the residual flux assembly.

Figure 30 shows the solution obtained on a horizontal cut passing through the recirculation zone. We observe that the Dirichlet/Neumann gives better results than the Neumann/Dirichlet, both for velocity and pressure.

Finally, Figure 31 compares the convergence of the GMRES method used to solve the momentum equations. The convergence of the implicit Dirichlet/Neumann method is between the fine and coarse one-domain convergences.

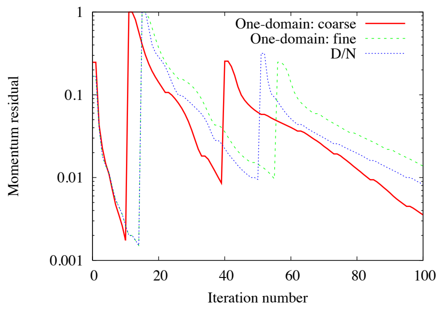


Fig. 31: GMRES convergence of the momentum equation.

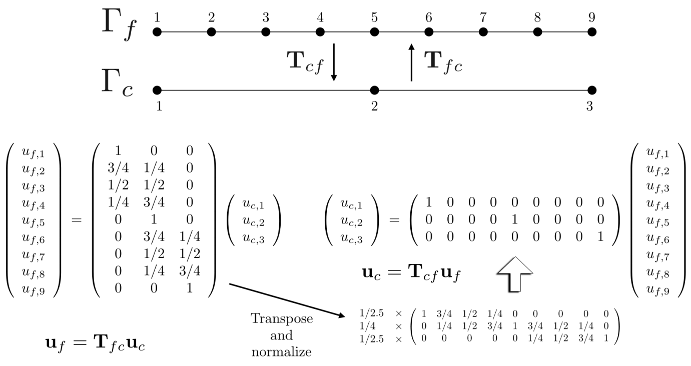

Γ_f
Γ_c
$\mathbf{T}_{cf}$
$\mathbf{T}_{fc}$
$\mathbf{u}_f = \mathbf{T}_{fc}\mathbf{u}_c$
$\mathbf{u}_c = \mathbf{T}_{cf}\mathbf{u}_f$
Transpose and normalize

Fig. 32: Normalized transposition. Linear interpolation from coarse/fine to fine/coarse grids.

4.3.4 Conservation via normalized transposition

Let us consider fine and coarse meshes, identified by subscripts f and c, respectively. Let $\mathbf{T}_{fc}$ be the interpolation matrix from the coarse to the fine mesh, and $\mathbf{T}_{cf}$ the interpolation matrix from the fine to the coarse mesh. If $\mathbf{u}_f$ and $\mathbf{u}_c$ are the nodal unknowns on the fine and coarse meshes respectively, then:

$$\mathbf{u}_f = \mathbf{T}_{fc}\mathbf{u}_c, \quad \mathbf{u}_c = \mathbf{T}_{cf}\mathbf{u}_f. \tag{48}$$

In [47], a strategy based on the transmission matrices defined in Equation (48) is proposed. Let us take a look at Figure 32. We observe that the coarse to fine transmission matrix $\mathbf{T}_{fc}$ is almost full, whereas the fine to coarse counterpart

$\mathbf{T}_{cf}$ is almost empty and therefore misses a lot of information contained in $\mathbf{u}_f$. The idea is to take the transpose of matrix $\mathbf{T}_{fc}$ and then normalize the rows in order to be globally conservative:

$$(\mathbf{T}_{cf})_{ij} = \frac{(\mathbf{T}_{fc})_{ji}}{\sum_k (\mathbf{T}_{fc})_{ki}}.$$

4.4 Implementation

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, described in [82]. One important point of interest here is the parallelization strategy. The mesh is partitioned using METIS [68], and the main communication operations present in the iterative solvers are [66]:

– Global communications with `MPI_ALLREDUCE` to compute scalar products;
– Non-blocking point-to-point communications with `MPI_ISEND` and `MPI_IRECV` to assemble the results of matrix-vector products.

Additional communications may be required to assemble complex preconditioners, but they are not of interest in the following. We will now briefly explain the steps required to build implicit and explicit couplings in this parallel context.

4.4.1 Assembly of Dirichlet and Neumann conditions

We are considering the implementations of Neumann and Dirichlet transmission conditions for non-matching meshes, given by Equations (39) and (40), respectively. From the point of view of the target, the idea is to modify its boundary array $\mathbf{q}^{(t)}$ using the values of its n_{neigh} source neighbors $\mathbf{q}^{(s_p)}$ with $p = 1, \ldots n_{\text{neigh}}$. Then, for Neumann and Dirichlet conditions, Equations (39) and (40) read, respectively:

$$\text{Neumann:} \quad \mathbf{q}^{(t)} = \mathbf{q}^{(t)} + \sum_{p=1}^{n_{\text{neigh}}} \mathbf{T}^{s_p}\mathbf{q}^{(s_p)},$$

$$\text{Dirichlet:} \quad \mathbf{q}^{(t)} = \sum_{p=1}^{n_{\text{neigh}}} \mathbf{T}^{s_p}\mathbf{q}^{(s_p)}.$$

There are therefore two issues: the construction of the transmission matrix and the communication of the arrays in the distributed memory context.

The construction of the transmission matrix should be carried out in parallel. Some hints on how this can be done are given in [48]. In general, it has

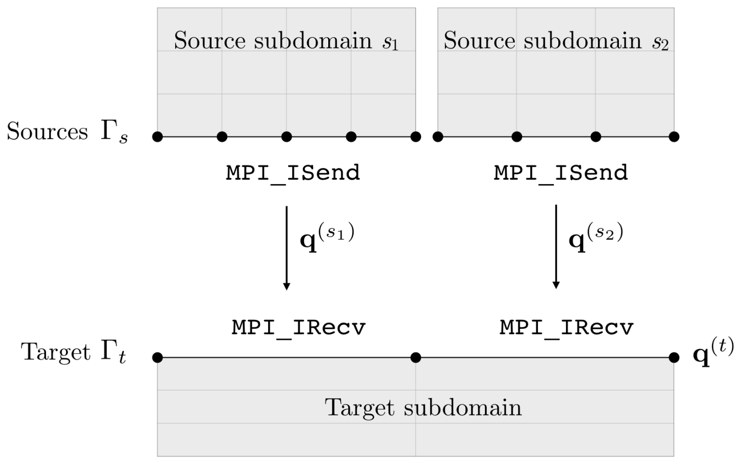


Fig. 33: Exchange between source and target subdomains.

contributions both from the target and the sources, but it is eventually assembled at the target. Therefore, each time a Neumann or a Dirichlet condition is to be applied, the target needs the result arrays $\mathbf{q}^{(s_p)}$ from all its neighbors. Obviously, the size of the array should be restricted only to the nodes involved in the coupling. This is achieved via `MPI_ISEND` and `MPI_IRECV` functions, as illustrated in Figure 33.

4.4.2 Explicit vs *implicit coupling*

For the implicit coupling, which operations take place inside the iterative solver, the same instance of the code Alya is used. For the explicit coupling, different instances of the code are used. Therefore, in order to be able to solve partitioned coupled problems, it is necessary to enable the subdomains to communicate inside an instance of Alya code for the implicit coupling, or between different instances of Alya code for the explicit coupling. These two possiblities are illustrated in Figure 34. Referring to the example of the figure, the MPI execution commands would be:

– Implicit coupling:

```
mpirun -n 7 code1.x physics1
```

– Explicit coupling:

```
mpirun -n 3 code1.x physics1 : -n 4 code2.x physics2
```

We note that in the case of implicit coupling, as shown in Figure 34 (Left), METIS does not necessarily partitions the sudbomains independently. We will

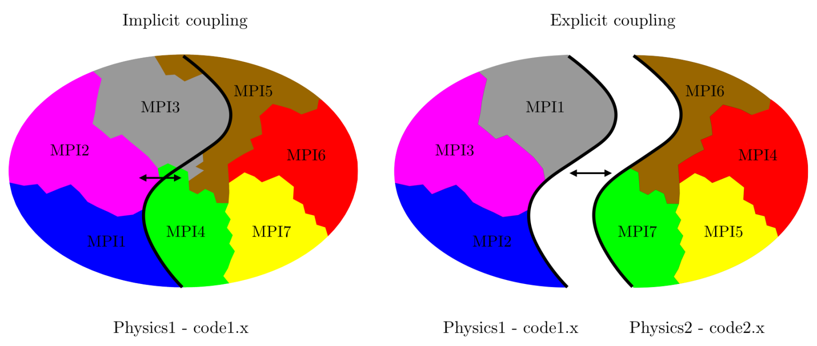


Fig. 34: MPI ranks for implicit and explicit couplings.

now explain how communicators can be built to be able to communicate between different partitions involved in the couplings.

4.4.3 Distributed memory context

In a distributed memory context, communicators are necessary to communicate between the different target and source MPI processes in charge of the different partitions. In a given coupling, not necessarily all the partitions are involved. Thus specific communicators are necessary. To be able to perform these coupling communications between subdomains, a communicator hierarchy is built, as illustrated in Figure 35.

In addition, we define *colors*: a color is an integer associated to a combination of a code (instance of Alya) and a subdomain. For example, referring to last figure, if we have three instances of Alya, the first two having one single subdomain and the third one two subdomains, then we have in total four colors. Subdomain 1 of Alya 1 has color 1, subdomain 1 of Alya 2 has color 2, and subdomains 1 and 2 of Alya 3 have color 3 and color 4, respectively. Each of these colors has its own MPI communicator, obtained through successive calls to the `MPI_COMM_SPLIT` function, starting from the original `MPI_COMM_WORLD` communicator.

A coupling can therefore be determined between colors. To be able to exchange data, communicators are created between the desired colors. In the example of Figure 35, we perform an explicit coupling between color 1 and color 2, and an implicit coupling between color 3 and color 4. Note that in this case no implicit coupling is permitted between different instances of the code.

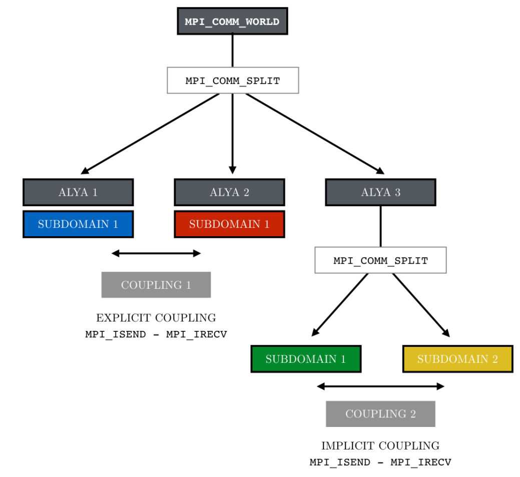


Fig. 35: MPI communicators hierarchy in Alya code.

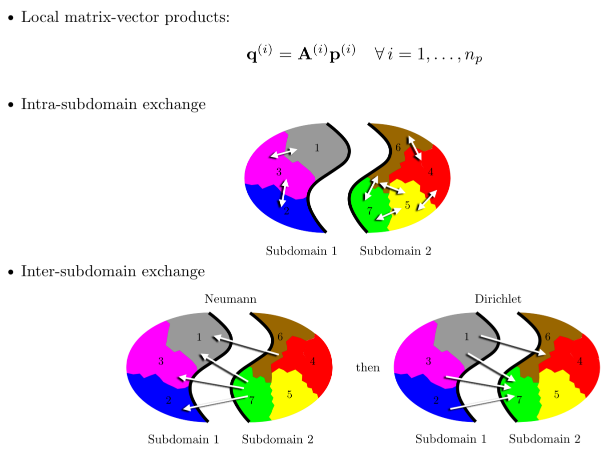

• Local matrix-vector products:
$\mathbf{q}^{(i)} = \mathbf{A}^{(i)}\mathbf{p}^{(i)} \quad \forall\, i = 1, \ldots, n_p$
• Intra-subdomain exchange
Subdomain 1
Subdomain 2
• Inter-subdomain exchange
Neumann
then
Dirichlet
Subdomain 1
Subdomain 2
Subdomain 1
Subdomain 2

Fig. 36: Two-subdomain coupling with seven partitions.

4.4.4 *Implicit coupling: matrix-vector product*

As we explained in Section 4.1.4, the implicit coupling should be carried out once the matrix-vector product has been carried out in each subdomain. However, to do this last operation in parallel, an exchange between the MPI partitions is necessary. Figure 36 explains the mechanism. First of all, the n_p partitions carry out their local matrix-vector product. Then, an intra-subdomain exchange is necessary to recover the same result one would obtain in sequential. Finally, an additional exchange is required to impose first the Neumann condition and then the Dirichlet condition of the coupling: this is the inter-subdomain exchange.

5 Conclusions

We have presented a survey on domain composition methods for computational mechanics problems. These methods were classified as mesh-based, equation-based and algebraic-based. For these categories, we reviewed some of the methods available in the literature.

We dedicated special attention to the algebraic-based methods as they have the advantage to be equation independent, more easily implementable and parallelizable. Explicit and implicit couplings were derived, for both disjoint and overlapping subdomains. In addition, several conservation strategies were analyzed which enable to conserve global physical quantities. Some implementation aspects were finally given to achieve algebraic coupling on distributed memory supercomputers.

Considering all these aspects, we pointed out that the implicit algebraic coupling has the great benefit of concentrating all the coupling operations inside the matrix-vector product operations of the iterative solvers. This great characteristic is desirable from an implementation point of view: the numericist can concentrate on the equation modeling and assembly; the point-to-point communications are located in a single operation, namely the matrix-vector product.

6 Compliance with Ethical Standards

This article does not contain any studies with human participants or animals performed by any of the authors. For this type of study formal consent is not required. The authors declare that they have no conflict of interest.

References

1. Alonso, A., Trotta, R., Valli, A.: Coercive domain decomposition algorithms for advection-diffusion equations and systems. J. Comput. Appl. Math. **96**, 51–76 (1998)
2. Auge, A., Kapurkin, A., Lube, G., Otto, F.C.: A note on domain decomposition of singularly perturbed elliptic problems. In: P.E. Bjørstad, M. Espedal, D. Keyes (eds.) Ninth International Conference on Domain Decomposition Methods, pp. 163–170. ddm.org (1998)
3. Avila, M., Folch, A., Houzeaux, G., Eguzkitza, B., Prieto, L., Cabezón, D.: A parallel CFD model for wind farms. Procedia Computer Science **18**, 2157–2166 (2013)
4. Beckert, A.: Coupling fluid (CFD) and structural (FE) models using finite interpolation elements. Aerosp. Sci. Technol. **4**, 13–22 (2000)
5. Beckert, A., Wendland, H.: Multivariate interpolation for fluid-structure-interaction problems using radial basis functions. Aerosp. Sci. Technol. **5**, 125–134 (2001)
6. Behr, M., Tezduyar, T.: The shear-slip mesh update method. Comp. Meth. Appl. Mech. Eng. **174**, 261–274 (1999)

7. Belytschko, T., Krongauz, Y., Organ, D., Fleming, M., Krysl, P.: Meshless methods: an overview and recent developments. Comp. Meth. Appl. Mech. Eng. **139**(1), 3–47 (1996)
8. Belytschko, T., Organ, D., Krongauz, Y.: A coupled finite element-element-free Galerkin method. Computational Mechanics **17**(3), 186–195 (1995)
9. Ben Belgacem, F., Maday, Y.: The mortar element method for three-dimensional finite elements. R.A.I.R.O. Modél. Math. Anal. Numér. **31**(2), 289–302 (1997)
10. Bernardi, C., Maday, Y., Patera, A.: Domain decomposition by the mortar element method. In: H. Kaper, M. Garbey (eds.) Asymptotic and Numerical Methods for Partial Differential Equations with Critical Parameters, vol. 384, pp. 269–286. Kluwer (1993)
11. Bernardi, C., Maday, Y., Rapetti, F.: Basics and some applications of the mortar element method. GAMM-Mitt **28**(2), 97–123 (2005)
12. Biotto, C., Peiró, J.: A zonal Euler/viscous solver for compressible flows. In: J. Pereira, A. Sequeira (eds.) Proceedings of the V European Conference on Computational Fluid Dynamics, ECCOMAS CFD 2010 (2010)
13. Bjørstad, P.E., Widlund, O.B.: Iterative methods for the solution of elliptic problems on regions partitioned into substructures. SIAM J. Numer. Anal. **23**, 1097–1120 (1986)
14. Blacker, T., Bohnhoff, W., Edwards, T.: Cubit mesh generation environment. volume 1: Users manual. Tech. rep., Sandia National Labs., Albuquerque, NM (United States) (1994)
15. de Boer, A., van Zuijlen, A., Bijl, H.: Review of coupling methods for non-matching meshes. Comp. Meth. Appl. Mech. Eng. **196**, 1515–1525 (2007)
16. Bourgat, J.F., Glowinski, R., Le Tallec, P., Vidrascu, M.: Variational formulation and algorithm for trace operator in domain decomposition calculations. In: T. Chan, R. Glowinski, J. Périaux, O. Widlund (eds.) Domain Decomposition Methods for Partial Differential Equations, pp. 3–16. SIAM, Philadelphia, PA (1989)
17. Brezzi, F., Canuto, C., Russo, A.: A self-adaptive formulation for the Euler/Navier-Stokes coupling. Comp. Meth. Appl. Mech. Eng. **73**(1989), 317–330 (1989)
18. Brezzi, F., Marini, L.: A three-field domain decomposition method. In: Domain Decomposition Methods in Science and Engineering, the Sixth International Conference on Domain Decomposition, *Contemporary Mathematics*, vol. 157, pp. 27–34. American Mathematical Society, Como (I) (1994)
19. Carlenzoli, C., Quarteroni, A.: Adaptive domain decomposition methods for advection-diffusion problems. In: I. Babuška (ed.) Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations, *IMA Volumes in Mathematics and its Applications*, vol. 75, pp. 165–186. Springer Verlag (1995)
20. Casoni, E., Jérusalem, A., Samaniego, C., Eguzkitza, B., Lafortune, P., Tjahjanto, D., Sáez, X., Houzeaux, G., Vźquez, M.: Alya: Computational solid mechanics for supercomputers. Archives of Computational Methods in Engineering pp. 1–20 (2014)
21. Cebral, J., Löhner, R., Choyke, P., Yim, P.: Merging of intersecting triangulations for finite element modeling. Journal of biomechanics **34**(6), 815–819 (2001)
22. Cebral, J.R., Löhner, R.: Conservative load projection and tracking for fluid-structure problems. AIAA Journal **35**(4), 687–692 (1997)
23. Ciccoli, M.C.: Adaptive domain decomposition algorithms and finite volume/finite element approximation for advection-diffusion equations. J. Sci. Comput. **11**(4), 229–341 (1996)
24. Ciccoli, M.C., Trotta, R.: Multidomain finite elements and finite volumes for advection-diffusion equations. In: P.E. Bjørstad, M. Espedal, D. Keyes (eds.) Ninth International Conference on Domain Decomposition Methods, pp. 540–547. ddm.org (1998)
25. Clark, B., Hanks, B., Ernst, C.: Conformal assembly meshing with tolerant imprinting. In: Proceedings of the 17th International Meshing Roundtable, pp. 267–280. Springer (2008)
26. Cleijne, J.: Results of Sexbierum wind farm. Report 92-388 **1** (1992)
27. Codina, R., Houzeaux, G.: Verification and Validation Methods for Challenging Multiphysics Problems, chap. Implementation Aspects of Coupled Problems in CFD involving time dependent domains, pp. 99–123. Theory and Engineering Applications of Computational Methods. CIMNE (2006)
28. Deutsch, C.V.: Geostatistical Reservoir Modeling. Oxford University Press (2002)

29. Discacciati, M., Gervasio, P., Giacomini, A., Quarteroni, A.: The Interface Control Domain Decomposition method for Stokes-Darcy coupling. SIAM J. Numer. Anal. **54**(2), 1039–1068 (2016)
30. Discacciati, M., Gervasio, P., Quarteroni, A.: Heterogeneous mathematical models in fluid dynamics and associated solution algorithms. In: Multiscale and adaptivity: modeling, numerics and applications, *Lecture Notes in Math.*, vol. 2040, pp. 57–123. Springer, Heidelberg (2012)
31. Discacciati, M., Gervasio, P., Quarteroni, A.: The Interface Control Domain Decomposition (ICDD) method for elliptic problems. SIAM J. Control Optim. **51**(5), 3434–3458 (2013)
32. Discacciati, M., Gervasio, P., Quarteroni, A.: Interface control domain decomposition methods for heterogeneous problems. Int. J. Num. Meth. Fluids **76**(8), 471–496 (2014)
33. Eguzkitza, B.: HERMESH: A geometrical domain composition method in computational mechanics. Ph.D. thesis, Universitat Politècnica de Catalunya, Barcelona (Spain) (2014)
34. Eguzkitza, B., Houzeaux, G., Aubry, R., Vázquez, M.: A parallel coupling strategy for the Chimera and domain decomposition methods in computational mechanics. Computers & Fluids **80**, 128–141 (2013)
35. Eguzkitza, B., Houzeaux, G., Calmet, H., Vázquez, M., Soni, B., Aliabadi, S., Bates, A., Doorly, D.: A gluing method for non-matching meshes. Computers & Fluids **110**, 159–168 (2013)
36. Farhat, C., Roux, F.: A method of finite element tearing and interconnecting and its parallel solution algorithm. Int. J. Num. Meth. Eng. **32**(6), 1205–1227 (1991)
37. Funaro, D., Quarteroni, A., Zanolli, P.: An iterative procedure with interface relaxation for domain decomposition methods. SIAM J. Numer. Anal. **25**, 1213–1236 (1988)
38. Gander, M., Halpern, L., Japhet, C., Martin, V.: Advection diffusion problems with pure advection approximation in subregions. In: O. Widlund, D. Keyes (eds.) Domain Decomposition Methods in Science and Engineering XVI, Lecture Notes in Computational Science and Engineering, pp. 239–246. Springer (2007)
39. Gander, M., Wanner, G.: The origins of the alternating Schwarz method. In: J. Erhel, M. Gander, L. Halpern, G. Pichot, T. Sassi, O. Widlund (eds.) Domain Decomposition Methods in Science and Engineering XXI, Lecture Notes in Computational Science and Engineering, pp. 415–422. Springer (2013)
40. Gastaldi, F., Gastaldi, L.: On a domain decomposition for the transport equation: Theory and finite element approximation. IMA J. Numer. Anal. **14**, 111–135 (1993)
41. Gastaldi, F., Gastaldi, L., Quarteroni, A.: ADN and ARN domain decomposition methods for advection-diffusion equations. In: P.E. Bjørstad, M. Espedal, D. Keyes (eds.) Ninth International Conference on Domain Decomposition Methods, pp. 334–341. ddm.org (1998)
42. Gastaldi, F., Quarteroni, A.: On the coupling of hyperbolic and parabolic systems: analytical and numerical approach. Applied Numer. Math. **6**(1), 3–31 (1989)
43. Gervasio, P., Lions, J.L., Quarteroni, A.: Heterogeneous coupling by virtual control methods. Numerische Mathematik **90**(2), 241–264 (2001)
44. Glowinski, R., Dinh, Q., Périaux, J.: Domain decomposition methods for nonlinear problems in fluid dynamics. Comp. Meth. Appl. Mech. Eng. **40**(1), 27–109 (1983)
45. Glowinski, R., Le Tallec, P.: Augmented Lagrangian interpretation of the nonoverlapping Schwarz alternating method. In: T.F. Chan, R. Glowinski, J. Périaux, O.B. Widlund (eds.) Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, pp. 224–231. SIAM, Philadelphia (USA) (1990)
46. Hoppe, R., Iliash, Y., Kuznetsov, Y., Vassilevski, Y., Wohlmuth, B.: Analysis and parallel implementation of adaptive mortar element methods. East-West J. Numer. Math. **6**, 223–248 (1998)
47. Houzeaux, G.: A geometrical domain decomposition method in computational fluid dynamics. Ph.D. thesis, Universitat Politècnica de Catalunya, Barcelona (Spain) (2002)
48. Houzeaux, G., Cajas, J., Eguzkitza, B., Vázquez, M.: Techniques for Parallel, Distributed and Cloud Computing in Engineering, *Computational Science, Engineering and Technology Series*, vol. 36, P. Iványi and B.H.V. Topping (Editors) edn., chap. Chapter 4: Parallel Implementation of Domain Composition Methods, pp. 61–92. Saxe-Coburg Publications (2015)

49. Houzeaux, G., Codina, R.: Transmission conditions with constraints in finite element domain decomposition method for flow problems. Commun. Numer. Meth. Eng. **17**, 179–190 (2001). Available at inter science
50. Houzeaux, G., Codina, R.: A Chimera method based on a Dirichlet/Neumann(Robin) coupling for the Navier-Stokes equations. Comp. Meth. Appl. Mech. Eng. **192**(31-32), 3343 – 3377 (2003)
51. Houzeaux, G., Codina, R.: An iteration-by-subdomain overlapping Dirichlet/Robin domain decomposition method for advection-diffusion problems. Journal of Computational and Applied Mathematics **158**(2), 243 – 276 (2003)
52. Houzeaux, G., Eguzkitza, B., Aubry, R., Owen, H., Vázquez, M.: A Chimera method for the Navier-Stokes equations. Int. J. Num. Meth. Fluids **75**, 155–183 (2014)
53. Huerta, A., Fernández-Méndez, S.: Enrichment and coupling of the finite element and meshless methods. Int. J. Num. Meth. Eng. **50**, 507–524 (2000)
54. Jaiman, R., Jiao, X., Geubelle, P., Loth, E.: Conservative load transfer along curved fluid-solid interface with non-matching meshes. J. Comput. Phys. **218**, 372–397 (2006)
55. Kim, H.: Interface element method (IEM) for a partitioned system with non-matching interfaces. Comp. Meth. Appl. Mech. Eng. **191**(29), 3165–3194 (2002)
56. Le Tallec, P., Tidriri, M.: Convergence analysis of domain decomposition algorithms with full overlapping for the advection diffusion problems. Math. Comput. **68**(226), 585–606 (1999)
57. Li, S., Liu, W.: Meshfree and particle methods and their applications. Applied Mechanics Reviews **55**(1), 1–34 (2002)
58. Lions, J.L., Pironneau, O.: Algorithmes parallèles pour la solution de problèmes aux limites. C. R. Acad. Sci. Paris, Série I **327**, 947–352 (1998)
59. Lions, P.L.: On the Schwarz alternating method I. In: R. Glowinski, G.H. Golub, G.A. Meurant, J. Périaux (eds.) First International Symposium on Domain Decomposition Methods for Partial Differential Equations, pp. 1–42. SIAM, Philadelphia (USA) (1988)
60. Lions, P.L.: On the Schwarz alternating method II. In: T. Chan, R. Glowinski, J. Périaux, O. Widlund (eds.) Domain Decomposition Methods, pp. 47–70. SIAM, Philadelphia (USA) (1989)
61. Lions, P.L.: On the Schwarz alternating method III: a variant for nonoverlapping subdomains. In: T.F. Chan, R. Glowinski, J. Périaux, O.B. Widlund (eds.) Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, pp. 202–223. SIAM, Philadelphia (USA) (1990)
62. Liou, M., Kao, K.: Progress in grid generation: From Chimera to DRAGON grids. Tech. Rep. NASA Technical Memorandum 106709, ICOMP-94-19, NASA (1994)
63. Liu, W., Uras, R., Chen, Y.: Enrichment of the finite element method with the reproducing kernel particle method. J. Appl. Mech **64**(4), 861–870 (1997)
64. Lo, S., Wang, W.: A fast robust algorithm for the intersection of triangulated surfaces. Engineering with Computers **20**(1), 11–21 (2004)
65. Maday, Y., Magoulès, F.: Absorbing interface conditions for domain decomposition methods: A general presentation. Comp. Meth. Appl. Mech. Eng. **195**, 3880–3900 (2006)
66. Magoulès, F., Roux, F.X., Houzeaux, G.: Parallel Scientific Computing. Computer Engineering Series. Wiley-ISTE (2015)
67. Marini, L., Quarteroni, A.: An iterative procedure for domain decomposition methods: A finite element approach. In: R. Glowinski, G.H. Golub, G.A. Meurant, J. Périaux (eds.) First International Symposium on Domain Decomposition Methods for Partial Differential Equations, pp. 129–143. SIAM, Philadelphia (USA) (1988)
68. METIS, family of multilevel partitioning algorithms. URL `http://glaros.dtc.umn.edu/gkhome/views/metis`
69. Nataf, F.: On the use of open boundary conditions in block Gauss-Seidel methods for the convection-diffusion equation. Tech. Rep. RI284, Centre de Mathématiques Appliquées, Ecole Polytechnique (1993)
70. Nataf, F., Nier, F.: Convergence rate of some domain decomposition methods for overlapping and nonoverlapping subdomains. Num. Math. **75**, 357–377 (1997)
71. Owen, S.J., Saigal, S.: Formation of pyramid elements for hexahedra to tetrahedra transitions. Comput. Meth. Appl. M. **190**(34), 4505 – 4518 (2001)

72. Quarteroni, A., Valli, A.: Domain decomposition methods for partial differential equations. Numerical Mathematics and Scientific Computation. The Clarendon Press, Oxford University Press, New York (1999). Oxford Science Publications
73. Rabczuk, T., Xiao, S., Sauer, M.: Coupling of mesh-free methods with finite elements: basic concepts and test results. Commun. Numer. Meth. Eng. **22**(10), 1031–1065 (2006)
74. Saad, Y.: Iterative Methods for Sparse Linear Systems. SIAM (2003)
75. Code_Saturne. URL http://code-saturne.org/cms
76. Si, H.: Tetgen, a delaunay-based quality tetrahedral mesh generator. ACM Trans. Math. Softw. **41**(2), 11:1–11:36 (2015)
77. Smith, B.F., Bjørstad, P.E., Gropp, W.D.: Domain decomposition. Cambridge University Press, Cambridge (1996)
78. Staten, M.L., Shepherd, J.F., Ledoux, F., Shimada, K.: Hexahedral mesh matching: Converting non-conforming hexahedral-to-hexahedral interfaces into conforming interfaces. Int. J. Num. Meth. Eng. **82**(12), 1475–1509 (2010)
79. Tian, R., Yagawa, G.: Non-matching mesh gluing by meshless interpolationan alternative to Lagrange multipliers. Int. J. Num. Meth. Eng. **71**(4), 473–503 (2007)
80. Toselli, A., Widlund, O.: Domain Decomposition Methods - Algorithms and Theory. Springer Series in Computational Mathematics. Springer Berlin Heidelberg (2004)
81. Trotta, R.: Multidomain finite elements for advection-diffusion equations. Applied Numer. Math. **21**, 91–118 (1996)
82. Vázquez, M., Houzeaux, G., Koric, S., Artigues, A., Aguado-Sierra, J., Arís, R., Mira, D., Calmet, H., Cucchietti, F., Owen, H., Taha, A., Burness, E.D., Cela, J.M., Valero, M.: Alya: Multiphysics engineering simulation towards exascale. J. Comput. Sci. **14**, 15–27 (2016)
83. White, D., Saigal, S., Owen, S.: An overset-grid method for 3D unsteady incompressible flows. Int. J. Num. Meth. Eng. **59**(14), 1839–1860 (2004)