Extension of the parallel Sparse Matrix Vector Product (SpMV) for the implicit coupling of PDEs on non-matching meshes

G. Houzeaux[a,*], R. Borrell[a], J.C Cajas[a], M. Vázquez[a]

[a]*Barcelona Supercomputing Center, Nexus II Building c/Jordi Girona, 29 08034 Barcelona (Spain)*

Abstract

The Sparse Matrix Vector Product (SpMV) is one of the main operations of iterative solvers, and, in a parallel context, it is also the siege of point-to-point communications between the neighboring MPI processes. The parallel SpMV is built in such a way that it gives, up to round off errors, the same result as its sequential counterpart. In this regards, nodes on the interfaces (or halo nodes if halo are considered) are duplicated nodes of the same original mesh. It is therefore limited to matching meshes. In this work, we generalize the parallel SpMV to glue the solution of non-matching (non-conforming) meshes through the introduction of transmission matrices. This extension of the SpMV thus enables the implicit and parallel solution of partial differential equations on non-matching meshes, as well as the implicit coupling of multi-physics problems, such as fluid-structure interactions. The proposed method is developed similarly to classical parallelization techniques and can therefore

*Corresponding author

Email addresses: guillaume.houzeaux@bsc.es (G. Houzeaux), ricard.borrell@bsc.es (R. Borrell), juan.cajas@bsc.es (J.C Cajas), mariano.vazquez@bsc.es (M. Vázquez)

be implemented by modifying few subroutines of an already MPI-based code. According to the proposed framework, the classical parallelization technique appears as a particular case of this general setup.

Keywords: Parallel sparse-matrix vector product, SpMV, MPI, Parallelization, Non-matching meshes, Coupling

1. Introduction

At the algebraic level, historical methods to glue non-matching meshes are the mortar method [1, 2] and the Finite Element Tearing and Interconnecting (FETI) method [3], where the continuity of the solution is imposed through a Lagrange multiplier. See [4] for a comparison of both methods. The main drawback of these strategies is, apart from introducing additional unknowns to the original problem, their non-trivial implementations. The alternative method we propose was introduced in [5] and this work concentrates on implementation aspects, in a parallel computing environment. It is based on extending the parallel matrix-vector product through the introduction of transmission matrices, to express Dirichlet and Neumann couplings between non-matching subdomains. Also, these transmission can be built in such a way to obtain local and global conservation properties. The resulting method is implicit and can be implemented on the top of already existing parallelization methods for the sparse matrix-vector product (SpMV). Figure 1 shows some applications of the method, from the coupling of one single physics on several subdomains to the coupling of different sets of equations, like the case of fluid-structure interactions. The only requirements are that the different sets of equations are solved with the same iterative solver for

the same variable (e.g. velocity), as the coupling is carried out at the SpMV level.

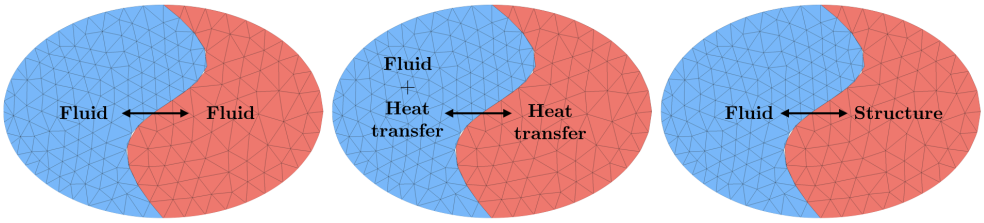


Figure 1: Example of applications of the proposed methodology.

The parallel version of a SpMV is constructed in such a way to give the same result as its sequential counterpart. The way parallel SpMV is carried out depends on whether full-row or partial-row matrices are built on each MPI partition [6]. On the one hand, if full row matrices are considered, the nodes and associated matrix rows are assigned exclusively to one MPI partition. To construct such matrices, halo elements or extra-communications are thus required. In the finite element context, partial-row matrices are quite common as local matrices are assembled from element matrices coming from a partition into disjoint sets of elements. The rows of the duplicated interface nodes are thus not fully assembled on the interfaces. In this case, parallelization of the SpMV consists in exchanging the local results of the SpMV to assemble the interface contributions coming from the neighbors. This work extends this technique to cases where the nodes on the interfaces do not coincide. The proposed methodology thus generalizes the concept of coupling, for matching and non-matching meshes.

To introduce the method, we will start by considering the particular case of two subdomains. In Section 2, we will present the classical parallelization strategy used when dealing with partial-row local matrices and matching meshes. This strategy will be reinterpreted in terms of a Domain Decomposition method (DD), based on a Dirichlet/Neumann coupling to couple the different local meshes. In Section 3, this DD framework will enable us to devise the same strategy for non-matching meshes, by introducing transmission matrices to transfer Dirichlet and Neumann data from one mesh to the other. The resulting methodology can thus be used for both couplings between matching meshes, where each mesh represent a partition, and between non-matching meshes. In this context, the SpMV for matching meshes is just a particular case of this extended SpMV. Then, we will give in Section 5 some hints on how other operations of iterative solvers can integrate the non-matching context as well. We will finally generalize the proposed strategy to an arbitrary number of subdomains in Section 4, by integrating in the same framework matching subdomains (as in the case of classical parallelization techniques) and non-matching subdomains as proposed here. The method will be illustrated through the analysis of the trace of a typical SpMV.

The algorithms presented in this work will be given in such a way that operations can be carried out locally on each subdomain and when needed, communications will be indicated. The proposed framework should thus enable one to introduce non-matching meshes coupling in an already existing parallel code, where point-to-point communications are the main glue to obtain a global solution.

2. Revisiting the parallel SpMV

In this section we will reinterpret the parallel SpMV in the context of domain decomposition methods. This framework will help us to devise the version for non-matching degrees of freedom coming from the gluing of non-conforming meshes. This will be done in next section. First of all, we will describe the classical strategy to parallelize the matrix-vector product. For this, we consider the following algebraic system coming from the discretization of a Partial Differential Equation (PDE) in a domain Ω:

$$\mathbf{A}\mathbf{u} = \mathbf{b}. \tag{1}$$

2.1. Classical SpMV

For the sake of simplification, we consider a partitioning of Ω into two subdomains Ω_1 and Ω_2. 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 denoted by $\mathbf{u}_\Gamma$. By performing a simple node reordering, System (1) can be written as:

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

In a parallel context, by considering a partition into disjoint subsets of elements, interface nodes are duplicated and the latter system is never fully assembled. See Figure 2 (Left). Instead, we obtain two independent systems

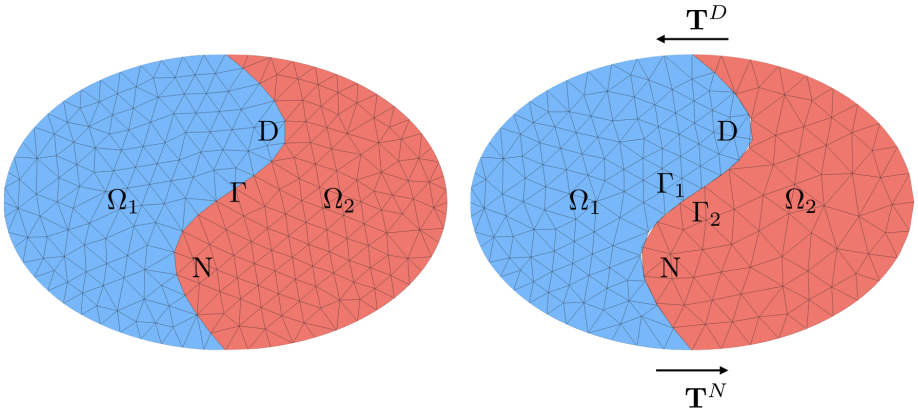


Figure 2: Two-subdomain decomposition with matching and non-matching meshes on the interface.

for each subdomains:

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{1\Gamma} \\ \mathbf{A}_{\Gamma 1} & \mathbf{A}_{\Gamma\Gamma}^{(1)} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_\Gamma^{(1)} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_\Gamma^{(1)} \end{pmatrix},$$

$$\begin{pmatrix} \mathbf{A}_{22} & \mathbf{A}_{2\Gamma} \\ \mathbf{A}_{\Gamma 2} & \mathbf{A}_{\Gamma\Gamma}^{(2)} \end{pmatrix} \begin{pmatrix} \mathbf{u}_2 \\ \mathbf{u}_\Gamma^{(2)} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_\Gamma^{(2)} \end{pmatrix},$$

where we have $\mathbf{A}_{\Gamma\Gamma} = \mathbf{A}_{\Gamma\Gamma}^{(1)} + \mathbf{A}_{\Gamma\Gamma}^{(2)}$ and $\mathbf{b}_\Gamma = \mathbf{b}_\Gamma^{(1)} + \mathbf{b}_\Gamma^{(2)}$. From now on, we will append a superscript (i) to indicate a partial matrix or vector obtained locally in subdomain i.

Let us consider the matrix-vector product $\mathbf{y} = \mathbf{A}\mathbf{x}$ in this two-domain context. The parallelization of this product is based on the distributivity property of the multiplication. On the one hand, the product for interior

nodes is straightforward as each subdomain is able to perform it independently, that is $\mathbf{y}_i = \mathbf{A}_{ii}\mathbf{x}_i + \mathbf{A}_{i\Gamma}\mathbf{x}_\Gamma^{(i)}$ for $i = 1, 2$, provided $\mathbf{x}_\Gamma^{(1)} = \mathbf{x}_\Gamma^{(2)} = \mathbf{x}_\Gamma$. For the interface we have:

$$\begin{aligned}\mathbf{y}_\Gamma &= \mathbf{A}_{\Gamma 1}\mathbf{x}_1 + \mathbf{A}_{\Gamma 2}\mathbf{x}_2 + \mathbf{A}_{\Gamma\Gamma}\mathbf{x}_\Gamma, \\ &= (\mathbf{A}_{\Gamma 1}\mathbf{x}_1 + \mathbf{A}_{\Gamma\Gamma}^{(2)}\mathbf{x}_\Gamma) + (\mathbf{A}_{\Gamma 2}\mathbf{x}_2 + \mathbf{A}_{\Gamma\Gamma}^{(2)}\mathbf{x}_\Gamma), \\ &= \mathbf{y}_\Gamma^{(1)} + \mathbf{y}_\Gamma^{(2)}, \end{aligned} \tag{3}$$

where $\mathbf{y}_\Gamma^{(i)}$ can be calculated independently for $i = 1, 2$. In the MPI context, the assembled result on the interface $\mathbf{y}_\Gamma$ is obtained by exchanging the local values $\mathbf{y}_\Gamma^{(1)}$ between neighbors, through the MPI non-blocking functions `MPI_Isend` and `MPI_Irecv`. A classical and non-optimized implementation is shown in Algorithm 1. In practice, communication of the interface values

Algorithm 1 SpMV for matching meshes.

1: Compute local results for $i = 1, 2$:

$$\mathbf{y}_i = \mathbf{A}_{ii}\mathbf{x}_i + \mathbf{A}_{i\Gamma}\mathbf{x}_\Gamma,$$

$$\mathbf{y}_\Gamma^{(i)} = \mathbf{A}_{\Gamma i}\mathbf{x}_i + \mathbf{A}_{\Gamma\Gamma}^{(i)}\mathbf{x}_\Gamma.$$

2: Exchange results $\mathbf{y}_\Gamma^{(1)}$ and $\mathbf{y}_\Gamma^{(2)}$ between subdomains.

3: Assemble results in subdomains 1 and 2:

$$\mathbf{y}_\Gamma = \mathbf{y}_\Gamma^{(1)} + \mathbf{y}_\Gamma^{(2)}.$$

$\mathbf{y}_\Gamma^{(i)}$ can be overlapped with the computation of the interior result $\mathbf{y}_i$ [7].

2.2. SpMV as a domain (de)composition method

The parallelization technique used previously to obtain the right answer on the interface can be reinterpreted in terms of a domain decomposition method. Let us duplicate formally the interface Γ into Γ_1 and Γ_2 and its associated unknowns, $\mathbf{u}_{\Gamma_1}$ and $\mathbf{u}_{\Gamma_2}$, respectively. We then set the following Dirichlet/Neumann problem at the algebraic level [5]:

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{1\Gamma_1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_{\Gamma_1} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix}$$
$$\begin{pmatrix} \mathbf{A}_{22} & \mathbf{A}_{2\Gamma_2} \\ \mathbf{A}_{\Gamma_2 2} & \mathbf{A}_{\Gamma_2\Gamma_2} \end{pmatrix} \begin{pmatrix} \mathbf{u}_2 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_2 \\ \mathbf{b}_{\Gamma_2} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{r}_{\Gamma_1} \end{pmatrix} \tag{4}$$

where the residual is

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

The Dirichlet condition is the second equation that states that $\mathbf{u}_{\Gamma_1} = \mathbf{u}_{\Gamma_2}$ and the Neumann condition consists in assembling Neumann data $\mathbf{r}_{\Gamma_1}$ of subdomain 1 in subdomain 2 interface equation. We can easily check that this system is equivalent to system (2). Last system can be rewritten as:

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

A matrix vector $\mathbf{y} = \mathbf{A}\mathbf{x}$ product using this matrix gives on the interface:

$$\mathbf{y}_{\Gamma_2} = (\mathbf{A}_{\Gamma_1 1}\mathbf{x}_1 + \mathbf{A}_{\Gamma_1\Gamma_1}\mathbf{x}_{\Gamma 1}) + (\mathbf{A}_{\Gamma_2 2}\mathbf{x}_2 + \mathbf{A}_{\Gamma_2\Gamma_2}\mathbf{x}_{\Gamma 2}),$$

$$\mathbf{y}_{\Gamma_1} = \mathbf{y}_{\Gamma_2}.$$

In practice, this SpMV can be carried out exactly as in Equation 3, by doing local matrix-vector products and then exchange the solution.

We have briefly shown how the substructuring method classically used to implement the parallel SpMV can be reinterpreted in terms of a Dirichlet/Neumann method at the algebraic level. Based on this domain decomposition framework, we devise in next section an equivalent method for non-matching meshes, by introducing transmission matrices.

Finally, it should be stressed that despite the fact that we use the formalism of the Dirichlet/Neumann method, the proposed implementation has nothing to do with the classical implementations presented in the litterature. In general, the method is implemented in a staggered way (*a la* Jacobi o *a la* Gauss-Seidel), sometimes as a preconditioner, sometimes as solver [8]. In our case the method is implicit and the associated matrix is exactly the same as the monolithic one, upon elimination of $\mathbf{u}_{\Gamma_1}$. Thus the convergence of any iterative solver will be the same as the monolithic case, contrary to the case of staggered methods where the coupling does not even necessarily converge.

3. Extended SpMV to non-matching meshes

We now consider non-matching meshes on the interface, as depicted in the right part of Figure 2. For the sake of calrity, we still consider only two subdomains.

3.1. Domain decomposition framework

As degrees of freedom do not coincide, we need to introduce some transmission matrices to express the couplings between the unknowns on the subdomain interfaces. Let us introduce transmission matrices for the Dirichlet

and Neumann conditions, $\mathbf{T}^D$ and $\mathbf{T}^N$ respectively, as explained in [5]. In this non-matching case, Equation (4) becomes:

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

$$\begin{pmatrix} \mathbf{A}_{22} & \mathbf{A}_{2\Gamma_2} \\ \mathbf{A}_{\Gamma_2 2} & \mathbf{A}_{\Gamma_2\Gamma_2} \end{pmatrix} \begin{pmatrix} \mathbf{u}_2 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_2 \\ \mathbf{b}_{\Gamma_2} \end{pmatrix} + \begin{pmatrix} \mathbf{0} \\ \mathbf{T}^N \mathbf{r}_{\Gamma_1} \end{pmatrix}.$$

The Dirichlet condition is now $\mathbf{u}_{\Gamma_1} = \mathbf{T}^D \mathbf{u}_{\Gamma_2}$ while the Neumann data transforms into $\mathbf{T}^N \mathbf{r}_{\Gamma_1}$. These matrices are rectangular and their sizes are:

$$\text{size}(\mathbf{T}^D) = \text{size}(\mathbf{u}_{\Gamma_1}) \times \text{size}(\mathbf{u}_{\Gamma_2}), \tag{5}$$

$$\text{size}(\mathbf{T}^N) = \text{size}(\mathbf{u}_{\Gamma_2}) \times \text{size}(\mathbf{u}_{\Gamma_1}). \tag{6}$$

Monolithic system. As done in the matching case, this system of equations can be recast in the following monolithic form:

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{1\Gamma_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & -\mathbf{T}^D \\ \mathbf{0} & \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{2\Gamma_2} \\ \mathbf{T}^N \mathbf{A}_{\Gamma_1 1} & \mathbf{T}^N \mathbf{A}_{\Gamma_1\Gamma_1} & \mathbf{A}_{\Gamma_2 2} & \mathbf{A}_{\Gamma_2\Gamma_2} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_{\Gamma_1} \\ \mathbf{u}_2 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{0} \\ \mathbf{b}_2 \\ \mathbf{b}_{\Gamma_2} + \mathbf{T}^N \mathbf{b}_{\Gamma_1} \end{pmatrix} \tag{7}$$

Note that upon elimination of $\mathbf{u}_{\Gamma_1}$, we obtain the following equivalent system:

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{A}_{1\Gamma_1}\mathbf{T}^D \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{2\Gamma_2} \\ \mathbf{T}^N\mathbf{A}_{\Gamma_1 1} & \mathbf{A}_{\Gamma_2 2} & \mathbf{A}_{\Gamma_2\Gamma_2} + \mathbf{T}^N\mathbf{A}_{\Gamma_1\Gamma_1}\mathbf{T}^D \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_{\Gamma_2} \end{pmatrix} = \begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \mathbf{b}_{\Gamma_2} + \mathbf{T}^N\mathbf{b}_{\Gamma_1} \end{pmatrix} \tag{8}$$

3.2. Transmission matrices

The selection of the transmission matrices depends on the interpolation or projection schemes used to express the couplings among the unknowns of the subdomain interfaces, that is to transmit the Dirichlet and Neumann data. One judicious choice consists in opting for $\mathbf{T}^N = (\mathbf{T}^D)^t$ in order to preserve the symmetry of the original system, as can be checked in last system of equations. Figure 3 shows examples of transmission matrix, computed using a linear interpolation and L_2 projection, from coarse-to-fine and fine-to-coarse meshes transmissions. Among others, two important properties that should satisfy the transmission matrices are the conservation of a constant unknown and the conservation of the total residual. The first property is satisfied by $\mathbf{T}^D$ whenever the sum of the coefficients of each row is equal to one. The second property concerns the residual $\mathbf{r}_{\Gamma_1}$, which represents the reacting "force" to the Dirichlet condition. In order to conserve this "force", we can show that the sum of the coefficients of each column should be equal to one

Fine-to-coarse transmission matrix

Linear interpolation

$$\mathbf{T}^D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

L_2 projection

$$\mathbf{T}^D = \begin{pmatrix} 5/12 & 1/2 & 1/12 & 0 & 0 \\ 1/24 & 1/4 & 5/12 & 1/4 & 1/24 \\ 0 & 0 & 1/12 & 1/2 & 5/12 \end{pmatrix}$$

Coarse-to-fine transmission matrix

Linear interpolation

$$\mathbf{T}^D = \begin{pmatrix} 1 & 0 & 0 \\ 0.5 & 0.5 & 0 \\ 0 & 1.0 & 0 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{pmatrix}$$

L_2 projection

$$\mathbf{T}^D = \begin{pmatrix} 5/6 & 1/6 & 0 \\ 0.5 & 0.5 & 0 \\ 1/12 & 5/6 & 1/12 \\ 0 & 0.5 & 0.5 \\ 0 & 1/6 & 5/6 \end{pmatrix}$$

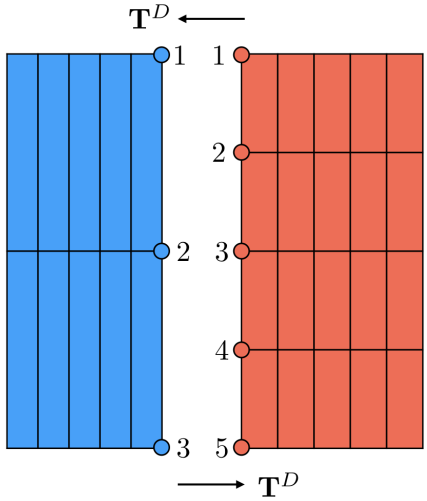


Figure 3: Example of transmission matrices.

as well. To summarize:

$$\text{Constant field conservation} : \sum_b \mathbf{T}^D_{ab} = 1 \quad \forall a$$

$$\text{Total residual conservation} : \sum_a \mathbf{T}^N_{ab} = 1 \quad \forall b.$$

The first property is inherited by both linear interpolation and L_2 projection. The second property is automatically satisfied if $\mathbf{T}^D$ satisfies the first one and $\mathbf{T}^N = (\mathbf{T}^D)^t$. See [5] for a description of the properties of different transmission matrices.

To illustrate these conservation properties when computing transmission matrices, let us consider the example of Figure 4. It consists of a deforming solid fixed on the bottom left node and under a force applied on the top right node. A large deformation problem governed by a linear constitutive model

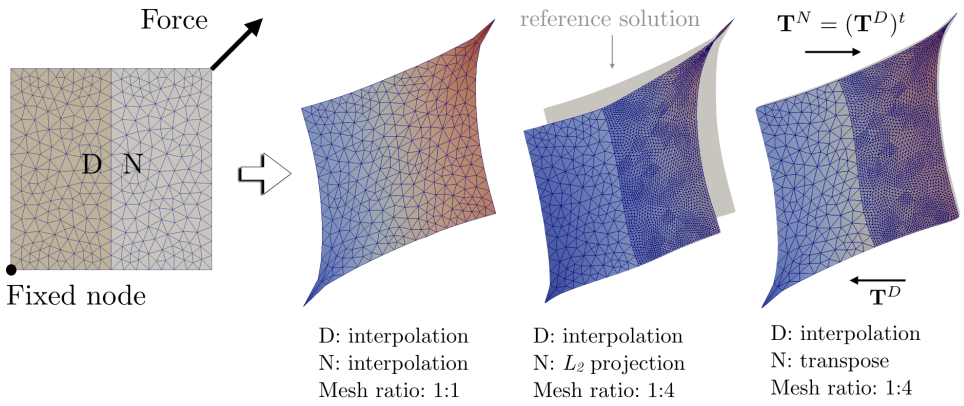


Figure 4: Large deformation of a solid. Dirichlet/Neumann for coinciding meshes and meshes with mesh ratio 1:4, using different methods to compute the transmission matrices.

is solved. A Dirichlet condition is imposed on the left part of the interface and a Neumann condition on the right part. When meshes coincide, when using linear interpolation for both the Dirichlet and Neumann conditions, the solution is the same as the one-domain solution. Now let us consider non-matching meshes, with a ratio of 4 from the Dirichlet to the Neumann subdomains. The Dirichlet condition is based on a simple linear interpolation, while for the Neumann condition, L_2 projection and the transpose of the Dirichlet transmission matrix are compared. We observe a good agreement with the reference solution obtain on a mesh with the same size as the Neumann subdomain. The transpose is not only very practical from the implementation point of view, but inherits some nice conservation properties. It conserves symmetry as well as the total force, as explained in [5].

3.3. Partitioned formulation

System (7) couples non-matching subdomains. To obtain an implicit coupling, we have basically two options. The first option attacks this monolithic system by assembling the connection matrices $\mathbf{T}^N\mathbf{A}_{\Gamma_1 1}$, $\mathbf{T}^N\mathbf{A}_{\Gamma_1\Gamma_1}$ and right-hand side $\mathbf{T}^N\mathbf{b}_{\Gamma_1}$, which, in practice, can be non trivial to implement. The second option consists in achieving the coupling through the matrix-vector product, involving the transmission matrices. This can be done as follows:

$$\begin{aligned}\mathbf{y}_{\Gamma_2} &= (\mathbf{A}_{\Gamma_2 2}\mathbf{x}_2 + \mathbf{A}_{\Gamma_2\Gamma_2}\mathbf{x}_{\Gamma_2}) + (\mathbf{T}^N\mathbf{A}_{\Gamma_1 1}\mathbf{x}_1 + \mathbf{T}^N\mathbf{A}_{\Gamma_1\Gamma_1}\mathbf{x}_{\Gamma_1}),\\ &= \mathbf{y}^{(2)}_{\Gamma_2} + \mathbf{T}^N\mathbf{y}^{(1)}_{\Gamma_1},\end{aligned} \tag{9}$$

followed by a Dirichlet step which consists in imposing $\mathbf{y}_{\Gamma_1} = \mathbf{T}^D\mathbf{y}_{\Gamma_2}$. A straightforward implementation is shown in Algorithm 2.

However these sequence is not efficient, as it involves two separate communications, given by steps 2 and 4 of the algorithm. Instead, Algorithm 3 joins these communications while being strictly equivalent to the previous one. In practice, the communications shown in step 3 of last algorithm can be overlapped with the computations of $\mathbf{y}_1$ and $\mathbf{y}_2$.

What about the Dirichlet condition?. We have developed a strategy to obtain an implicit coupling for non-conforming meshes by modifying the matrix-vector product. But why is this equivalent to the matrix-vector product one would obtain the monolithic system (7). By observing the result after the Neumann step we can easily check that step 3 gives the same result as Equation (9). We also guarantee that the result of the matrix-vector product also satisfies the Dirichlet condition on $\mathbf{x}_{\Gamma_1}$. But can we make sure that we get the same solution as that of the monolithic system through iterative solvers

Algorithm 2 SpMV for non-matching degrees of freedom: method 1

1: Compute local results for $i = 1, 2$:

$$\mathbf{y}_i = \mathbf{A}_{ii}\mathbf{x}_i + \mathbf{A}_{i\Gamma_i}\mathbf{x}_{\Gamma_i},$$

$$\mathbf{y}_{\Gamma_i}^{(i)} = \mathbf{A}_{\Gamma_i i}\mathbf{x}_i + \mathbf{A}_{\Gamma_i\Gamma_i}\mathbf{x}_{\Gamma_i}.$$

2: Subdomain 1 sends $\mathbf{y}_{\Gamma_1}^{(1)}$ to subdomain 2.

3: Assemble result in subdomain 2:

$$\mathbf{y}_{\Gamma_2} = \mathbf{y}_{\Gamma_2}^{(2)} + \mathbf{T}^N\mathbf{y}_{\Gamma_1}^{(1)}.$$

4: Subdomain 2 sends $\mathbf{y}_{\Gamma_2}$ to subdomain 1.

5: Apply Dirichlet condition in subdomain 1:

$$\mathbf{y}_{\Gamma_1} = \mathbf{T}^D\mathbf{y}_{\Gamma_2}.$$

Algorithm 3 SpMV for non-matching degrees of freedom: method 2

1: Compute local results for $i = 1, 2$:

$$\mathbf{y}_i = \mathbf{A}_{ii}\mathbf{x}_i + \mathbf{A}_{i\Gamma_i}\mathbf{x}_{\Gamma_i},$$

$$\mathbf{y}_{\Gamma_i}^{(i)} = \mathbf{A}_{\Gamma_i i}\mathbf{x}_i + \mathbf{A}_{\Gamma_i\Gamma_i}\mathbf{x}_{\Gamma_i}.$$

2: Exchange results $\mathbf{y}_{\Gamma_1}^{(1)}$ and $\mathbf{y}_{\Gamma_2}^{(2)}$ between subdomains.

3: Assemble results in subdomain 1 and 2:

$$\mathbf{y}_{\Gamma_2} = \mathbf{y}_{\Gamma_2}^{(2)} + \mathbf{T}^N\mathbf{y}_{\Gamma_1}^{(1)},$$

$$\mathbf{y}_{\Gamma_1} = \mathbf{T}^D\left(\mathbf{y}_{\Gamma_2}^{(2)} + \mathbf{T}^N\mathbf{y}_{\Gamma_1}^{(1)}\right).$$

iterations?

In general, Krylov solvers [9] are based on simple solution updates like:

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \alpha \mathbf{p}^k.$$

This equation corresponds to the solution update of the classical Conjugate Gradient method. Let us consider the first iteration with $k = 1$:

$$\mathbf{p}^0 = \mathbf{b} - \mathbf{A}\mathbf{u}^0,$$

$$\mathbf{u}^1 = \mathbf{u}^0 + \alpha \mathbf{p}^0,$$

and compute the updates on the interfaces. We need two initial conditions which satisfy the following relations to make it work:

1. RHS satisfies the Dirichlet condition: $\mathbf{b}_{\Gamma 1} = \mathbf{T}^D \mathbf{b}_{\Gamma 2}$,
2. Initial solution satisfies the Dirichlet condition: $\mathbf{u}^0_{\Gamma 1} = \mathbf{T}^D \mathbf{u}^0_{\Gamma 2}$. (10)

We have

$$\begin{aligned} \mathbf{p}^0_{\Gamma 1} &= \mathbf{b}_{\Gamma_1} - \mathbf{A}\mathbf{u}^0 \mid_{\Gamma_1}, \\ &= \mathbf{T}^D \mathbf{b}_{\Gamma_2} - \mathbf{T}^D (\mathbf{A}\mathbf{u}^0 \mid_{\Gamma_2}), \\ &= \mathbf{T}^D (\mathbf{b}_{\Gamma_2} - \mathbf{A}\mathbf{u}^0 \mid_{\Gamma_2}), \\ &= \mathbf{T}^D \mathbf{p}^0_{\Gamma 2}, \end{aligned} \tag{11}$$

where we have used Equation (10) and step 5 of Algorithm 2, or, equivalently, step 3 of Algorithm 3. We observe that the Dirichlet condition is directly transferred to the conjugate direction $\mathbf{p}$. Then the update of the solution $\mathbf{u}$

on Γ_1 yields:

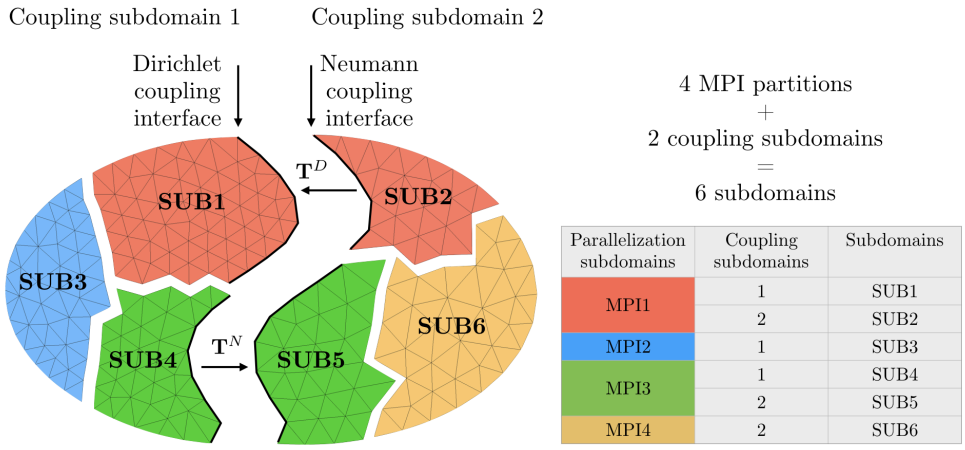
$$\begin{aligned}
\mathbf{u}^1_{\Gamma 1} &= \mathbf{u}^0_{\Gamma 1} + \alpha \mathbf{p}^0_{\Gamma 1}, \\
&= \mathbf{T}^D \mathbf{u}^0_{\Gamma 2} + \alpha \mathbf{T}_D \mathbf{p}^0_{\Gamma 2}, \\
&= \mathbf{T}^D (\mathbf{u}^0_{\Gamma 2} + \alpha \mathbf{p}^0_{\Gamma 2}), \\
&= \mathbf{T}^D \mathbf{u}^1_{\Gamma 2},
\end{aligned}$$

where we have used Equations (10) and (11). Thus, the first update satisfies the Dirichlet condition as well. We could show that, recursively, the new updates will always satisfy the Dirichlet condition.

Key message. If meshes are coinciding on the interface, then $\mathbf{T}^D = \mathbf{T}^N = \mathbf{I}$, and we recover the classical parallel implementation of the SpMV given by Equation (3) with $\mathbf{y}_{\Gamma_1} = \mathbf{y}_{\Gamma_2} = \mathbf{y}_\Gamma$. The proposed technique is thus a *generalization of the parallel SpMV* for non-matching meshes.

4. Extension to an arbitrary number of subdomains

Let us consider a parallel context with n_{MPI} MPI partitions, referred to as *parallelization subdomains*, and which consist of disjoint sets of elements. Let us consider also n_{cou} *coupling subdomains*, possibly non-matching, and coupled through Dirichlet and Neumann conditions. The intersections between the parallelization and coupling subdomains define a more general disjoint set of elements, simply referred to as *subdomain*. Figure 5 shows an example of four parallelization subdomains involving a non-matching coupling between two coupling subdomains. In practice, we start with an MPI partition into n_{MPI} parallelization subdomains. Then, each MPI partitions is divided into



Parallelization subdomains	Coupling subdomains	Subdomains
MPI1	1	SUB1
	2	SUB2
MPI2	1	SUB3
MPI3	1	SUB4
	2	SUB5
MPI4	2	SUB6

Figure 5: Example of four parallelization subdomain and two coupling subdomains.

subdomains if it involves elements belonging to several coupling subdomains. In the example of the figure, we end up with six subdomains.

To generalize the coupling algorithm for parallelization and coupling subdomains, we have to discriminate between Dirichlet and Neumann nodes. In this general context, the relation between Neumann and Dirichlet nodes should be understood as a master-worker relation, the master being the Neumann node, as shown in Algorithm 2:

> *Neumann nodes accumulate residuals, and Dirichlet nodes transmit these residuals.*

Figure 6 shows the process for selecting Dirichlet and Neumann nodes when parallelization and coupling subdomains coexist.

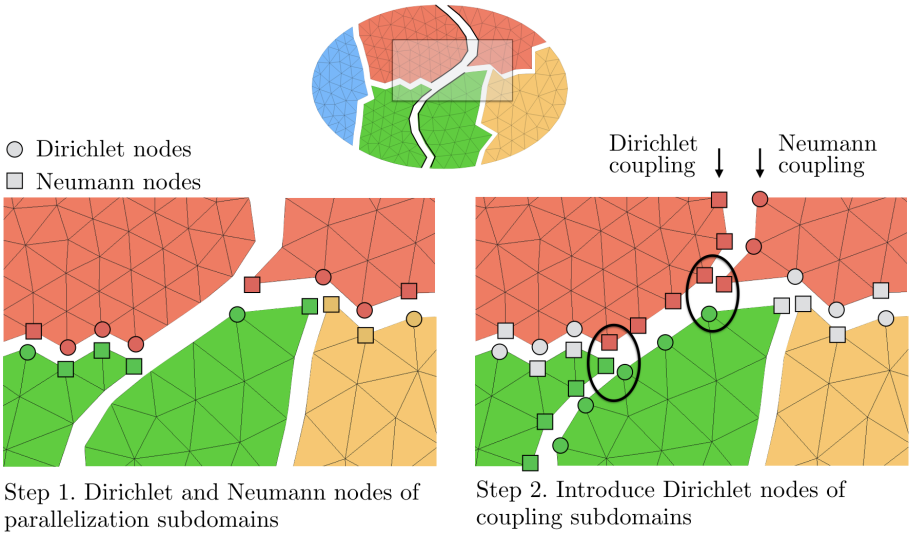


Figure 6: Selection of Dirichlet and Neumann nodes in two steps.

Step 1: selection of parallelization Dirichet and Neumann nodes. Let us start with the couplings between the parallelization subdomains, used in a classical parallelization strategy. The Dirichlet and Neumann nodes discrimination consists in selecting one of the duplicated nodes on the interfaces, mark it as Neumann, and then mark the duplicates as Dirichlet nodes. This is an explicit or implicit common practice in parallel codes, as illustrated on the left part of the figure. In fact, for the SpMV, the relation between duplicated interface nodes does not need to be explicit, as transmission matrices are equal to identity (see Algorithm 1). The distinction between these nodes is only necessary in the scalar product, where only the contributions of the interior and Neumann nodes are required to avoid duplication of the scalar product on the interfaces, as explained in Section 5.1.

Step 2: selection of coupling Dirichet and Neumann nodes. To fully establish a coupling for non-matching meshes, we have to select the Dirichlet and Neumann sides as well as the algorithms to compute the transmission matrices (interpolation, projection, transpose). The second step of the Dirichlet and Neumann node selection consists in marking as Dirichlet nodes the nodes located on the Dirichlet side of the coupling interface. In particular, Neumann nodes selected in the first step and located on the Dirichlet side must be converted into Dirichlet nodes (see the bottom marked zone on the right part of Figure 6). On the Neumann side, we then mark all the non-Dirichlet nodes as Neumann nodes (see the top marked zone on the right part of Figure 6).

By performing steps 1 and 2, we thus ensure that Neumann nodes are uniquely defined on parallelization and coupling subdomains interfaces.

The complete algorithm to couple the subdomains is very simple to describe, and a bit cumbersome to write formally. In brief, the Neumann nodes accumulate the residuals coming from its neighbors, through transmission matrices. The Dirichlet nodes are related to their Neumann counterparts through matrix $\mathbf{T}^D$ and thus depend on the same neighbors as its master (Neumann) node. Nos let us go to the formal algorithm.

Let n_i be the number of neighbors of subdomain i, including itself, and n_{Γ_i} be the size (number of nodes) of the interface of i. For the sake of simplicity, the transmission matrix $\mathbf{T}^N_{ij}$ which transmits data from subdomain j to subdomain i is dimensioned as $n_{\Gamma_i} \times n_{\Gamma_j}$. In fact, only a sub-block of $\mathbf{T}^N_{ij}$ would be necessary as only subsets of Γ_i and Γ_j give non-zero coefficients on

the matrix. The rows corresponding to Dirichlet nodes and the rows which do not involve neighbors j are set to zero. To account for its local residual, we also set $\mathbf{T}_{ii}^N = \mathbf{I}$. Using this formalism, the Neumann condition reads:

$$\mathbf{y}_{\Gamma_i}^N = \sum_{j=1}^{n_i} \mathbf{T}_{ij}^N \mathbf{y}_{\Gamma_j}^{(j)}. \tag{12}$$

As far as the Dirichlet transmission matrix $\mathbf{T}_{ij}^D$ is concerned, the rows corresponding to the Neumann nodes are set to zero, and we set $\mathbf{T}_{ii}^D = \mathbf{0}$. The Dirichlet condition on Γ_i coming from subdomain j is thus

$$\mathbf{y}_{\Gamma_{ij}}^D = \mathbf{T}_{ij}^D \mathbf{y}_{\Gamma_j}^N = \mathbf{T}_{ij}^D \left(\sum_{k=1}^{n_j} \mathbf{T}_{jk}^N \mathbf{y}_{\Gamma_k}^{(k)} \right),$$

where we have used Equation (12). Now we need to sum over all the neighbors of i in order to take into account all the Dirichlet conditions coming from all the Dirichlet neighbors so that

$$\mathbf{y}_{\Gamma_i}^D = \sum_{j=1}^{n_i} \mathbf{y}_{\Gamma_{ij}}^D = \sum_{j=1}^{n_i} \mathbf{T}_{ij}^D \left(\sum_{k=1}^{n_j} \mathbf{T}_{jk}^N \mathbf{y}_{\Gamma_k}^{(k)} \right).$$

Therefore, as $\mathbf{T}_{ij}^N$ has no null rows on Neumann nodes and $\mathbf{T}_{ij}^D$ has no null rows on Dirichlet nodes, we can sum up $\mathbf{y}_{\Gamma_i}^N$ and $\mathbf{y}_{\Gamma_i}^D$ to obtain $\mathbf{y}_{\Gamma_i} = \mathbf{y}_{\Gamma_i}^N + \mathbf{y}_{\Gamma_i}^D$. We finally end up with algorithm 4 which describes the extended SpMV for parallelization and coupling subdomains, and which consists of a generalization of Algorithm 3.

This algorithm illustrates the extended SpMV, but this should not be implemented as is. In fact, when receiving the contribution of a neighbor j (step 3), not all the interface result of j is needed, but only that where columns of the transmission matrices are non-zero. In addition, just like in

Algorithm 4 Extended SpMV.

1: Compute local results on each MPI partition i:

$$\mathbf{y}_i^{(i)} = \mathbf{A}_{ii}\mathbf{x}_i + \mathbf{A}_{i\Gamma_i}\mathbf{x}_{\Gamma_i},$$

$$\mathbf{y}_{\Gamma_i}^{(i)} = \mathbf{A}_{\Gamma_i i}\mathbf{x}_i + \mathbf{A}_{\Gamma_i\Gamma_i}\mathbf{x}_{\Gamma_i}.$$

2: Send $\mathbf{y}_{\Gamma_i}^{(i)}$ to all neighbors j.

3: Receive $\mathbf{y}_{\Gamma_j}^{(j)}$ from all neighbors j.

4: Assemble the result on the interface Γ_i:

$$\mathbf{y}_{\Gamma_i} = \sum_{j=1}^{n_i} \mathbf{T}_{ij}^N \mathbf{y}_{\Gamma_{ji}}^{(j)} + \sum_{j=1}^{n_i} \mathbf{T}_{ij}^D \left(\sum_{k=1}^{n_j} \mathbf{T}_{jk}^N \mathbf{y}_{\Gamma_k}^{(k)} \right).$$

classical parallelization techniques, interface communications represented in steps 2 and 3 can be overlapped with the SpMV for interior nodes using the non-blocking MPI communication subroutines.

Let us analyze the performance of Algorithm 4. We consider the geometry illustrated in Figure 7 and compare the SpMV for three cases, all on 64 CPUs. Case 1 consists of a one domain simulation with only parallelization coupling. Case 2 involves two separate domains with only parallelization coupling. Finally, case 3 involves the same two separate domains, with parallelization coupling in each, and non-matching coupling between then. For the three cases, the total number of elements and nodes is around 4M, which results in an average of 62500 nodes per MPI partitions. By comparing case 3 with 2, we wish to measure the extra-communications due to the coupling. By comparing case 3 with case 1, we wish to point out the fact that the

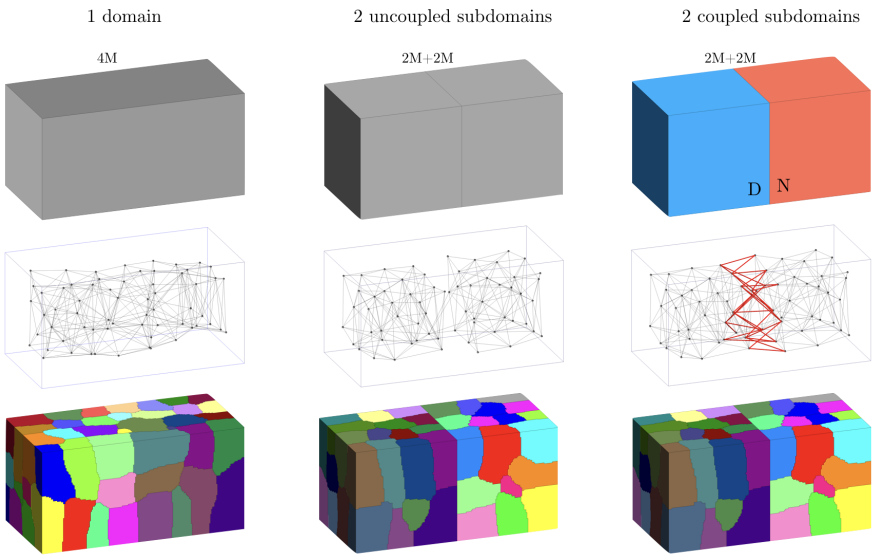


Figure 7: Example to test the performance of the extended SpMV. (Top) Geometries and mesh sizes. (Mid.) Connectivities between the different MPI partitions due to parallelization and coupling. (Bot.) Partitioning.

partitioner (herein METIS[10]) sees two non-connected geometries and is thus unable to minimize the interface sizes on the non-matching meshes. See for example [11] for coupling aware partitioning techniques.

On the middle part of the figure, we show the connectivities between the different subdomains. Each sphere represents a subdomain and its location corresponds to the subdomain center of gravity, while the lines symbolize connections between two subdomains. We can observe the extra connections between the MPI partitions in case 3 with respect to case 2. By taking a look at the MPI partititions, we also observe that the partitions of cases 2 and 3 are not aware of the coupling. In fact, in this work, the transmission matrices are computed in parallel after the partitioning.

Figure 8 shows three different traces of one single SpMV, representing the different tasks (colors) carried out by the different preocesses (y-axis) along time (x-axis). The blue color represents the SpMV on the interfaces (Dirichlet and Neumann). The white color represents the non-blocking communications using the MPI functions `MPI_ISend` and `MPI_IRecv`. The red color, which dominates the computation, is the SpMV for the interior nodes. Finally, the purple represents the `MPI_Waitall` which finalizes the SpMV. The top trace is that of the one-domain problem (case 1), the middle one the two-domain problem without coupling (case 2) and the bottom one the two-domain problem with coupling (case 3). We can observe the effects of the extra-communications involved in case 3. These communications do not exist in case 2, but they do in case 1. However, in this last case, METIS was able to take into account the minimization of the interfaces and thus to reduce

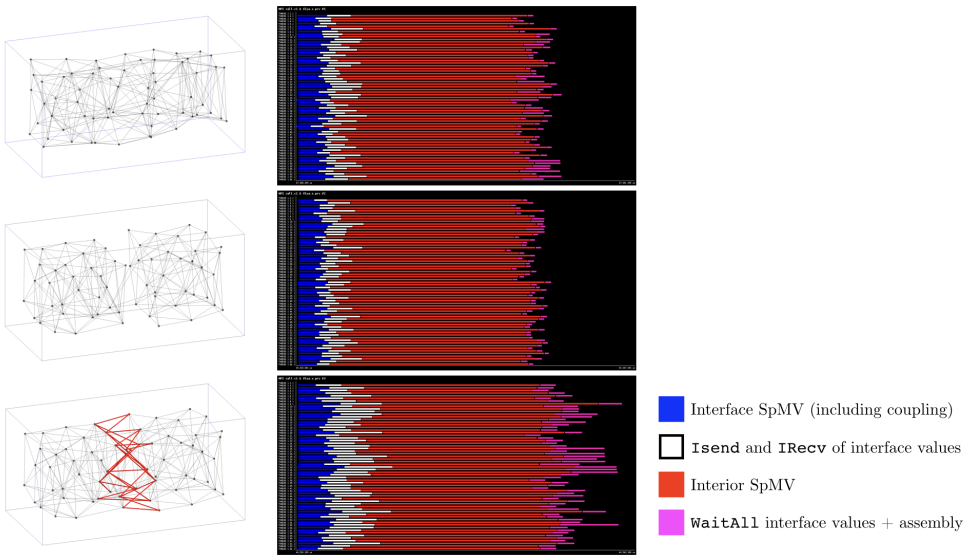


Figure 8: Traces of the SpMV on the geometries presented in Figure 7. The size of the window represents 2.7 ms. From Top to Bot.: Case 1: uncoupled problem represented by Algorithm 1; Case 2: coupled problem using a two-step communication scheme represented by Algorithm 2; Case 3: coupled problem using the one-step communication scheme represented by Algorithm 4.

communications. Even if the impact of communications in case 3, is limited, the traces suggest that a coupling aware partitioning should be used [11].

Note on the cost. The extra cost induced by the use of transmission matrices depends on the interpolation or projection used, as well as the mesh sizes on both sides of the interface. Noting that in the case of matching meshes the transmission matrices are unity, Equations (5) and (6) give an estimate of this extra cost. For example, in the case of boundary interpolation from a bilinear surface, the number of operations is four times higher than in the case of matching meshes, where only node-to-node information is needed.

5. Beyond SpMV: other solver operations

Matrix vector-product is obviously not the only operation of iterative solvers. In the parallel implementation of iterative solvers (e.g. Krylov solvers [9]), if one wants to get the same parallel sequence as the sequential one given by Equation (8), some additional implementation details are necessary. We will treat two of them in the context of non-matching meshes, namely the scalar product and the preconditioning phase. We will remain in the two-domain context for the sake of simplification, the extension to any number of subdomains being straightforward.

5.1. Scalar product

In classical parallelization techniques, the scalar product can be computed in different ways in order to obtain the same as the sequential one. One common approach consists in dividing the ownership of the interface nodes

between neighbors so that the scalar product contributions are not duplicated and its calculation is equally distributed.

In the domain decomposition context, the interpretation can be made differently. In fact, the Dirichlet unknowns are virtually eliminated from the solution process. Therefore, one could argues that the scalar product should be exclusively carried out on the Neumann interface. Using this domain decomposition jargon, the ownership aforementioned is equivalent to declaring the node as Neumann node. A generic scalar product $\alpha = \mathbf{x} \cdot \mathbf{y}$ can be computed as in Algorithm 5, which gives the same result as in the sequential (monolithic) case given by system of equations (8).

Algorithm 5 Scalar product for non-matching meshes.

1: Compute local results:

$$\alpha^{(1)} = \mathbf{x}_1 \cdot \mathbf{y}_1,$$

$$\alpha^{(2)} = \mathbf{x}_2 \cdot \mathbf{y}_2 + \mathbf{x}_{\Gamma 2} \cdot \mathbf{y}_{\Gamma 2}.$$

2: Sum up the two contributions:

$$\alpha = \alpha^{(1)} + \alpha^{(2)}.$$

5.2. Preconditioning

General case. The monolithic matrix $\mathbf{A}$ of our couple system is given by Equation (8). The preconditioner $\mathbf{M}$ should ideally be built based on this matrix. However, we mentioned in previous sections that it may not be convenient to build such a matrix, and this is the reason why we devised the implicit method by extending the parallel SpMV to non-matching meshes. In

practice, each subdomain could approximate its local interface matrices using only their local contributions. During the preconditioner step, symbolized by the solution of a generic problem $\mathbf{Mx} = \mathbf{y}$, the following operations are carried out on each subdomain interface:

$$\mathbf{x}_{\Gamma_1} = \mathbf{M}^{-1}\mathbf{y} \mid_{\Gamma_1},$$

$$\mathbf{x}_{\Gamma_2} = \mathbf{M}^{-1}\mathbf{y} \mid_{\Gamma_2},$$

However, as pointed out at the end of Section 3, the preconditioner, as well as any operation should satisfy the important property that $\mathbf{x}_{\Gamma_1} = \mathbf{T}^D\mathbf{x}_{\Gamma_2}$. However, this is not always possible, and some communication step is likely to be required to impose this Dirichlet condition once $\mathbf{x}_{\Gamma_2}$ has been updated. We will see in a moment that even for a simple preconditioner this is a hard task. Therefore, in general, the preconditioning step can be computed as shown in Algorithm 6. In this algorithm, communications are assumed to

Algorithm 6 Scalar product for non-matching degrees of freedom

1: Compute partial local preconditioning:

$$\mathbf{x}_{\Gamma_2} = \mathbf{M}^{-1}\mathbf{y} \mid_{\Gamma_2},$$

$$\mathbf{x}_1 = \mathbf{M}^{-1}\mathbf{y} \mid_1 .$$

2: Subdomain 2 sends $\mathbf{x}_{\Gamma_2}$ to subdomain 1.

3: Subdomain 1 waits for $\mathbf{x}_{\Gamma_2}$.

4: Finish preconditioning:

$$\mathbf{x}_2 = \mathbf{M}^{-1}\mathbf{y} \mid_2,$$

$$\mathbf{x}_{\Gamma 1} = \mathbf{T}^D\mathbf{x}_{\Gamma 2}.$$

be non-blocking, and subdomain 2 can go to step 4 of the algorithm just

after sending its interface contribution in step 2. This communication step is penalizing as it does not appear in classical parallelization methods, where transmission matrices are equal to identity. Let us examine the simplest preconditioner, the diagonal preconditioner.

Diagonal preconditioning. The question is whether or not we can construct two local diagonal preconditioners on the interfaces while avoiding communication. On the Neumann interface, referring to Equation (8), we have

$$\mathbf{D}_{\Gamma 2} = \mathrm{diag}(\mathbf{A}_{\Gamma_2\Gamma_2}) + \mathrm{diag}(\mathbf{T}^N \mathbf{A}_{\Gamma_1\Gamma_1} \mathbf{T}^D).$$

Thus, provided subdomain 2 has both transmission matrices, the diagonal can be computed. The Dirichlet side is more problematic. We wish to devise a diagonal matrix $\mathbf{D}_{\Gamma 1}$ to solve the following system

$$\mathbf{D}_{\Gamma 1} \mathbf{x}_{\Gamma_1} = \mathbf{y}_{\Gamma_1},$$

having in mind that $\mathbf{y}_{\Gamma_1}$ satisfies the Dirichlet condition $\mathbf{y}_{\Gamma_1} = \mathbf{T}^D \mathbf{y}_{\Gamma_2}$, and with the requirement that eventually $\mathbf{x}_{\Gamma_1} = \mathbf{T}^D \mathbf{x}_{\Gamma_2}$. Therefore

$$\mathbf{D}_{\Gamma 1} \mathbf{T}^D \mathbf{x}_{\Gamma_2} = \mathbf{T}^D \mathbf{y}_{\Gamma_2},$$

$$\mathbf{D}_{\Gamma 1} \mathbf{T}^D \mathbf{x}_{\Gamma_2} = \mathbf{T}^D \mathbf{D}_{\Gamma 2} \mathbf{x}_{\Gamma_2},$$

which should be satisfied for all $\mathbf{x}_{\Gamma_2}$, so that we end up with the following equation for $\mathbf{D}_{\Gamma 1}$:

$$\mathbf{D}_{\Gamma 1} \mathbf{T}^D = \mathbf{T}^D \mathbf{D}_{\Gamma 2}. \tag{13}$$

Matrix $\mathbf{T}^D$ is in general a rectangular matrix so that a classical inverse does not necessarily exist. The right inverse $(\mathbf{T}^D)^{-1,\mathrm{right}}$ of a rectangular

matrix is a matrix such that $\mathbf{T}^D(\mathbf{T}^D)^{-1,\text{right}} = \mathbf{I}$. One example is the pseudo inverse defined as:

$$(\mathbf{T}^D)^{-1,\text{right}} := (\mathbf{T}^D)^t[\,\mathbf{T}^D\,(\mathbf{T}^D)^t\,]^{-1}.$$

If we assume such a matrix exists, then Equation (13) reads

$$\mathbf{D}_{\Gamma 1} = \mathbf{T}^D\mathbf{D}_{\Gamma 2}(\mathbf{T}^D)^{-1,\text{right}}.$$

Fist of all, we do not have the guarantee that such an inverse exists. For example, only the fine to coarse transmission matrices given in Figure 3 are invertible. They are:

$$\text{Linear interpolation}: \quad (\mathbf{T}^D)^{-1,\text{right}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$L_2 \text{ projection}: \quad (\mathbf{T}^D)^{-1,\text{right}} = \begin{pmatrix} 394/305 & -4/5 & 94/305 \\ 1269/1220 & 3/10 & -171/1220 \\ -7/10 & 11/5 & -7/10 \\ -171/1220 & 3/10 & 1269/1220 \\ 94/305 & -4/5 & 394/305 \end{pmatrix}$$

In addition, should this matrix exist, $\mathbf{D}_{\Gamma 1}$ is also unlikely to be diagonal, which was our first requirement. Therefore, even in the simple case of a diagonal preconditioning, we eventually need communication to ensure that $\mathbf{x}_{\Gamma_1} = \mathbf{T}^D\mathbf{x}_{\Gamma_2}$.

6. Conclusions

The parallel solution of PDEs employing iterative methods like Krylov methods as algebraic solvers relies mainly on the parallelization of the sparse matrix-vector product (SpMV). If the local matrices to each MPI partition come from element integrations on disjoint sets of elements, the matrix rows of interface nodes are only partial. Then, the parallel version of the SpMV consists in performing local SpMVs, and then exchanging and assembling the results between neighbors on the interfaces.

We have developed in this work an extension of this parallel SpMV to account for non-matching meshes. This was achieved by introducing transmission matrices to express the couplings between the non-matching unknowns on the interface. This method was then merged with the classical parallel version of the SpMV in order to construct a more general and unified SpMV. This extended SpMV enables the implicit and parallel solution of PDEs for matching and non-matching meshes.

As implemented in this work, the coupling between non-matching subdomains, represented by the calculation of the transmission matrices, is carried out after the partitioning used for parallelization purpose. The partitioning is thus not aware of this coupling, and cannot minimize the communications. As a future work, coupling aware partitioning will be investigated. In addition, the method can be used to couple multiphysics problems (solving different sets of PDEs) implicitly and on non-matching meshes, provided they make use of the same iterative solvers.

[1] C. Bernardi, Y. Maday, A. Patera, Domain Decomposition by the Mortar Element Method, Springer Netherlands, Dordrecht, 1993, pp. 269–

286. doi:10.1007/978-94-011-1810-1_17.
URL https://doi.org/10.1007/978-94-011-1810-1_17

[2] C. Bernardi, Y. Maday, F. Rapetti, Basics and some applications of the mortar element method, GAMM-Mitt 28 (2) (2005) 97–123. doi:10.1002/gamm.201490020.
URL http://dx.doi.org/10.1002/gamm.201490020

[3] C. Farhat, F.-X. Roux, A method of finite element tearing and interconnecting and its parallel solution algorithm, Int. J. Num. Meth. Eng. 32 (6) (1991) 1205–1227. doi:10.1002/nme.1620320604.
URL http://dx.doi.org/10.1002/nme.1620320604

[4] C. Lacour, Y. Maday, Two different approaches for matching nonconforming grids: the mortar element method and the feti method, BIT Numer. Math. 37 (3) (1997) 720–739. doi:10.1007/BF02510249.
URL https://link.springer.com/article/10.1007/BF02510249

[5] G. Houzeaux, J. Cajas, M. Discacciati, B. Eguzkitza, A. Gargallo-Peiró, M. Rivero, M. Vázquez, Domain decomposition methods for domain composition purpose: Chimera, overset, gluing and sliding mesh methods, Arch. Comp. Meth. Eng. 24 (4) (2017) 1033–1070. doi:10.1007/s11831-016-9198-8.
URL https://link.springer.com/article/10.1007/s11831-016-9193-0

[6] G. Houzeaux, R. Borrell, Y. Fournier, M-Garcia-Gasulla, J. Göbbert, E. Hachem, V. Mehta, Y. Mesri, H. Owen, M. Vázquez, Computational

Fluid Dynamics, Adela Ionescu Edition, Intech, 2017, Ch. High performance computing: dos and don'ts.

[7] F. Magoulès, F.-X. Roux, G. Houzeaux, Parallel Scientific Computing, Computer Engineering Series, Wiley-ISTE, 2015.
URL http://eu.wiley.com/WileyCDA/WileyTitle/productCd-1848215819.html

[8] Domain decomposition methods for partial differential equations, Numerical mathematics and scientific computation, Oxford University Press, New York, 1999.

[9] Y. Saad, Iterative Methods for Sparse Linear Systems, SIAM, 2003. doi:10.1137/1.9780898718003.
URL https://doi.org/10.1137/1.9780898718003

[10] K. Lab, Metis - serial graph partitioning and fill-reducing matrix ordering.
URL http://glaros.dtc.umn.edu/gkhome/metis/metis/overview

[11] M. Predari, Load Balancing for Parallel Coupled Simulations, Theses, Université de Bordeaux, LaBRI ; Inria Bordeaux Sud-Ouest (Dec. 2016).
URL https://hal.inria.fr/tel-01518956