A parallel non-invasive mixed domain decomposition - Implementation and applications to mechanical assemblies

Paul Oumaziz, Pierre Gosselet, Pierre-Alain Boucard, Stéphane Guinard

A parallel non-invasive mixed domain decomposition - Implementation and applications to mechanical assemblies

Paul Oumaziz[1,2], Pierre Gosselet[2], Pierre-Alain Boucard[2], Stéphane Guinard[3]

Abstract

This paper proposes to confront a mixed domain decomposition method with industrial computations, in particular the simulation of quasi-static assemblies with frictional contact between the parts. The method is implemented in a non-invasive manner around an industrial finite element software. The performance of the algorithm is studied on industrial assemblies.

1 Introduction

It is a truism to say that industrialists are eager to conduct reliable simulations of their structures and therefore they need robust computational methods capable of handling models constituted by large amounts of degrees of freedom and involving complex nonlinear phenomena.

Domain decomposition methods (DDM) are a natural way to distribute the computational load on several cores. They also offer a framework where efficient iterative solvers can be implemented. In particular recent progress made it possible to design solvers with controlled convergence rate for linear symmetric positive definite systems for many DDM: see for instance [1] for overlapping methods, [2] for FETI and BDD methods, [3, 4, 5] for FETI-DP and BDDC; also multipreconditioning proved to be an efficient alternative [6] with possible application in more general contexts [7].

This paper focuses on problems involving frictionous contact between parts in small strain. Such a situation corresponds in particular to assemblies modeled at the scale of the connector (bolt....). For the domain decomposition simulation, a first possibility is to somehow decouple the contact from the domain decomposition by dealing with the contact inside subdomains, this was done in the alternate Schwarz case [8] or with the Balancing Domain Decomposition [9]. In the view of dealing with assemblies, it seems more natural to aim at non-overlapping DDM able to deal with contact at the interfaces between subdomains. In this context, an important body of work is constituted by the contributions of Dostal and co-workers around the FETI(DP) method [10, 11, 12, 13], where the contact problem is rephrased as a constrained minimization problem dealt with a projected Krylov solver enriched by an efficient subdomain-based preconditioner and adapted coarse space. The alternative investigated in this paper is the Latin method [14], which in the DDM language can be viewed as a non-overlapping optimized Schwarz method [15, 16], with dedicated treatment of the interface in order to insert complex mechanical behaviors.

Because they involve new practices and also because they require deep modification of the source code, DDM have not been massively adopted in commercial finite element software. In order to alleviate the latter problem, non-invasive implementations have been proposed, in particular since python driving of most software is possible. These implementations permit to demonstrate the methods on industrial problems and to exploit computational clusters, at the cost of reduced performance compared to true HPC implementations.

As an example, the global/local coupling [17, 18, 19, 20] brings accuracy to the method of submodeling which is very popular amongst industrialists. It was applied as a complete solver in [21]. In fact the global/local coupling can be

[1]Núcleo Científico Multidisciplinario-DI, Universidad de Talca, Camino Los Niches km1, Curicó, Chile
[2]LMT / ENS Cachan / CNRS / Université Paris-Saclay, 61 avenue du président Wilson, 94235 Cachan Cedex, France
[3]Airbus SAS, CRT/XRXF,18 rue Marius Terce, 31300 Toulouse, FRANCE

viewed as a particular non-overlapping Schwarz method where the Robin condition is managed by a coarse representation of the domain [22]. This idea was exploited in [23] for the non-invasive implementation of the Latin method, the Robin conditions being simulated through the addition of elements on the interfaces of subdomains. In [24], the extensibility of the non-invasive Latin method was studied by using the framework of [25].

In this paper, we propose to demonstrate the efficiency of the non-invasive Latin method of [23, 24] with semi-industrial applications. Here semi-industrial refers to the fact that extremely fine models that would require HPC are out of the reach of our implementation. We restrict our study to infinitesimal strain and we make use of conforming meshes. Note that the Latin method can naturally be coupled with the mortar approach [26, 27] in order to handle non-conforming interfaces, see [25] for instance.

The paper is organized as follows: Section 2 presents the non-invasive Latin method; Section 3 explains the parallel implementation; finally Section 4 illustrates the implementation with several semi-industrial simulations involving many frictionous contact surfaces.

2 The substructured reference problem

We consider a family of non-overlapping subdomains $(\Omega_E)_{E\in[1,N]}$ of $\mathbb{R}^3$ under quasi-static isotherm evolution and small perturbation assumptions. The subdomains are assumed to be made out of isotropic linear elastic material. Classical Lagrange conforming finite elements are used to approximate the subdomains' displacement field. For subdomain Ω_E, we let $\mathbf{K}_E$ be the stiffness matrix, $\mathbf{u}_E$ be the vector of nodal displacement, $\mathbf{f}_{dE}$ be the generalized forces associated to the given loads. We suppose that the degrees of freedom submitted to Dirichlet boundary conditions are treated by elimination.

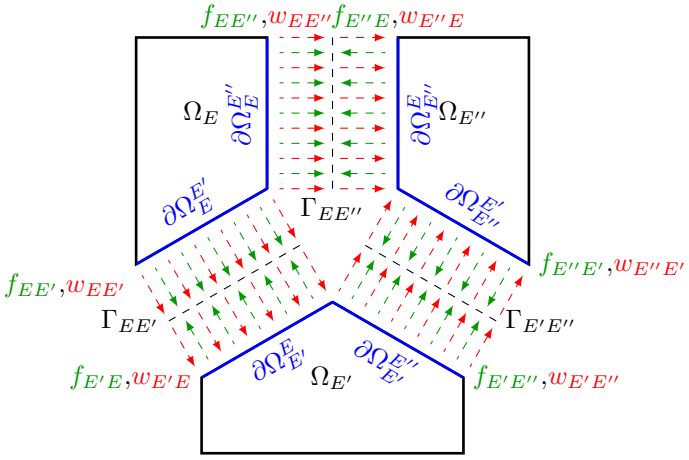


Figure 1: Force and displacement fields defined on the edges of the subdomains

Subdomain Ω_E interacts with its neighbor $\Omega_{E'}$ through the interface $\Gamma_{EE'} = \partial\Omega_E \cap \partial\Omega_{E'}$ (Figure 1). The interface $\Gamma_{EE'}$ is granted its own finite element discretization, and displacements and force distributions associated with its parent subdomains are defined: $(\mathbf{w}_{EE'}, \mathbf{w}_{E'E})$ and $(\mathbf{f}_{EE'}, \mathbf{f}_{E'E})$. These fields are connected to the subdomains by trace relations materialized by the operator $\mathbf{N}_{EE'}$. We note $\mathcal{E}$ the set of all subdomains and $\mathcal{G}$ the set of all interfaces.

To describe complex load sequences and to handle the dependency of the solution with respect to history, a pseudo-time is used. Thus nodal velocities $(\dot{\mathbf{w}}_{EE'}, \dot{\mathbf{u}}_E)$ are introduced with dot notations. These velocities are simply linked with displacement distributions through a backward Euler integration. If needed, the time step is indicated by a superscript. Typically:

$$\dot{\mathbf{u}}_E^{t+\Delta t} = \frac{\mathbf{u}_E^{t+\Delta t} - \mathbf{u}_E^t}{\Delta t} \tag{1}$$

We use concatenated operators in order to easily manage many interfaces:

$$\mathbf{w}_E = \begin{pmatrix} \vdots \\ \mathbf{w}_{EE'} \\ \vdots \end{pmatrix}, \quad \mathbf{f}_E = \begin{pmatrix} \vdots \\ \mathbf{f}_{EE'} \\ \vdots \end{pmatrix}, \quad \mathbf{N}_E = \begin{pmatrix} \ddots & & \\ & \mathbf{N}_{EE'} & \\ & & \ddots \end{pmatrix}, \tag{2}$$

where E' spans all the neighbors of subdomain E

The mechanical problems to be solved on the subdomains can be written as:

$$\forall \Omega_E \in \mathcal{E},\ \forall t, \quad \begin{cases} \mathbf{K}_E \mathbf{u}_E^t = \mathbf{f}_{dE}^t + \mathbf{N}_E^T \mathbf{f}_E^t \\ \mathbf{w}_E^t = \mathbf{N}_E \mathbf{u}_E^t \end{cases} \tag{3}$$

The mechanical behavior of the interfaces is given by a relation of the form:

$$\forall \Gamma_{EE'} \in \mathcal{G}, \quad \begin{cases} \mathbf{f}_{EE'}^t + \mathbf{f}_{E'E}^t = 0,\ \forall t \\ \mathbf{f}_{EE'}^t \in \mathbf{b}_{EE'} \left(\dot{\mathbf{w}}_{E'E}^{t'} - \dot{\mathbf{w}}_{EE'}^{t'}, t' \leqslant t; \mathbf{w}_{E'E}^0 - \mathbf{w}_{EE'}^0 \right),\ \forall t \\ \text{Initial conditions} \end{cases} \tag{4}$$

where the first relation is the balance of forces and the second relation represents the constitutive law of the mechanical behavior. The use of set notations for the behavior enables us to encompass most situations, among others perfect interfaces, contact, friction [28], cohesion[29, 30].

Block notations are used to simplify the writing of all the relations. As said earlier $\mathbf{x}_E$ represents the gathering of all the $(\mathbf{x}_{EE'})_{E'}$. $\mathbf{x}$ will represent the gathering of all the $(\mathbf{x}_E)$ defined on the subdomains, same procedure applies to operators:

$$\mathbf{x} = \begin{pmatrix} \vdots \\ \mathbf{x}_E \\ \vdots \end{pmatrix}, \qquad \mathbf{K} = \begin{pmatrix} \ddots & & 0 \\ & \mathbf{K}_E & \\ 0 & & \ddots \end{pmatrix}, \text{ where } E \text{ spans all subdomains.} \tag{5}$$

We introduce operators which permit to make neighboring subdomains communicate: the operator $\mathbf{A}$ makes sums of interface vectors whereas the operator $\mathbf{B}$ makes differences:

$$\begin{aligned} (\mathbf{A}\mathbf{f})_{|\Gamma_{EE'}} &= \mathbf{f}_{EE'} + \mathbf{f}_{E'E} \\ (\mathbf{B}\mathbf{w})_{|\Gamma_{EE'}} &= \mathbf{w}_{E'E} - \mathbf{w}_{EE'} \end{aligned} \tag{6}$$

These operators are orthogonal in the sense that $\ker(\mathbf{A}) = \text{range}(\mathbf{B}^T)$.

Thus, the discrete substructured problem can be written as:

$$\forall t, \begin{cases} \mathbf{K}\mathbf{u}^t = \mathbf{f}_d^t + \mathbf{N}^T \mathbf{f}^t & \text{Equilibrium of the subdomains} \\ \mathbf{w}^t = \mathbf{N}\mathbf{u}^t & \text{Trace of the subdomain displacement} \\ \mathbf{f}^t \in \mathbf{B}^T \mathbf{b}(\mathbf{B}\dot{\mathbf{w}}^{t'}, t' \leqslant t, \mathbf{B}\mathbf{w}^0) & \text{Interfaces' behavior} \end{cases} \tag{7}$$

3 The non-invasive Latin method

3.1 Principle of the Latin method

So as to solve the substructured problem we use the Latin method [14]. Two sets of interface distributions are defined :

$$\begin{aligned} &\mathcal{A} : (\mathbf{f}^t, \dot{\mathbf{w}}^t)_t \text{ solutions to } \forall t \begin{cases} \mathbf{K}\mathbf{u}^t = \mathbf{f}_d^t + \mathbf{N}^T \mathbf{f}^t \\ \dot{\mathbf{w}}^t = \mathbf{N}\dot{\mathbf{u}}^t \\ \dot{\mathbf{u}}^t = (\mathbf{u}^t - \mathbf{u}^{t-\Delta t})/\Delta t, + \text{ initial condition} \end{cases} \\ &\mathcal{L} : (\widehat{\mathbf{f}}^t, \widehat{\dot{\mathbf{w}}}^t)_t \text{ solutions to } \ \forall t\ \widehat{\mathbf{f}}^t \in \mathbf{B}^T \mathbf{b}(\mathbf{B}\widehat{\dot{\mathbf{w}}}^t, t' \leqslant t, \mathbf{B}\widehat{\dot{\mathbf{w}}}^t) \end{aligned} \tag{8}$$

$\mathcal{A}$ groups the solutions to the linear equations set on the subdomains, $\mathcal{A}$ is an affine space called *admissible space*. $\mathcal{L}$ is a manifold containing the solutions verifying the behavior of the interfaces. In general the relations defining $\mathcal{L}$ are point-wise independent both in time and space.

From the initialization with $s_0 \in \mathcal{A}$, an iterative scheme is used to reach the solution at the intersection of $\mathcal{A}$ and $\mathcal{L}$. Partial solutions are alternatively searched in $\mathcal{A}$ and in $\mathcal{L}$. This defines an iteration of the algorithm in two steps:

- the first step consists in searching an approximation $\widehat{s}_n = \left(\widehat{\mathbf{w}}_n, \widehat{\mathbf{f}}_n\right)$ in $\mathcal{L}$ starting from an approximation $s_n = (\mathbf{w}_n, \mathbf{f}_n)$ in $\mathcal{A}$. This stage is called *local stage*, it is computed by adding the relation of search direction:

$$\mathbf{f}_n - \widehat{\mathbf{f}}_n - \mathbf{k}_V^+ \left(\dot{\mathbf{w}}_n - \widehat{\dot{\mathbf{w}}}_n\right) = 0 \tag{9}$$

- the second step is about finding an approximation $s_{n+1} = (\mathbf{w}_{n+1}, \mathbf{f}_{n+1})$ in $\mathcal{A}$ starting from an approximation $\widehat{s}_n = \left(\widehat{\mathbf{w}}_n, \widehat{\mathbf{f}}_n\right)$ in $\mathcal{L}$. This stage is called *linear stage* and it is computed by adding the relation of search direction:

$$\mathbf{f}_{n+1} - \widehat{\mathbf{f}}_n + \mathbf{k}_V^- \left(\dot{\mathbf{w}}_{n+1} - \widehat{\dot{\mathbf{w}}}_n\right) = 0 \tag{10}$$

$\mathbf{k}_V^+$ and $\mathbf{k}_V^-$ are two operators chosen by the user. $\mathbf{k}_V^+$ is chosen to be diagonal in order to benefit from the local properties of the relations defining $\mathcal{L}$.

In order to ensure the convergence, a relaxation step is applied at the end of the linear stage:

$$s_{n+1} \leftarrow s_{n+1} + \alpha(s_{n+1} - s_n) \tag{11}$$

with $0 < \alpha \leqslant 1$. The iterative method is represented graphically on Figure 2, where $\tilde{s}_{n+1}$ is the result of the linear stage before relaxation.

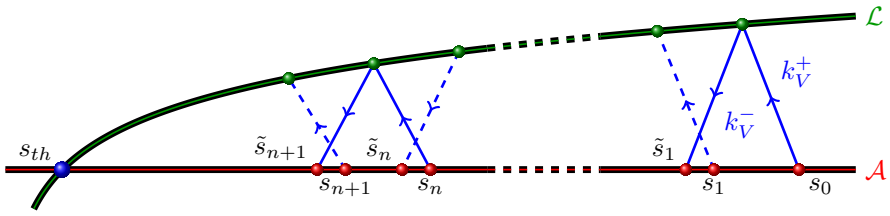


Figure 2: Latin method – ($\tilde{s}$ is the approximation before relaxation)

Proposition 1. *The convergence of* $(s_{n+1} + s_n)/2$ *is proved in [14] when* $\mathbf{k}_V^- = \mathbf{k}_V^+$ *are symmetric positive definite operators and* $\alpha < 1$ *for maximal monotonic behaviors* $\mathbf{b}$.

Generally an indicator of error is used to quantify the energetic distance between $\mathcal{A}$ and $\mathcal{L}$ for two successive partial solutions. This indicator is defined as follows:

$$\eta = \frac{\|\tilde{s}_{n+1} - \hat{s}_n\|^2_{\mathbf{k}_V^-}}{\|\tilde{s}_{n+1} + \hat{s}_n\|^2_{\mathbf{k}_V^-}} = \frac{\left(\dot{\mathbf{w}}_{n+1} - \widehat{\dot{\mathbf{w}}}\right)^T \mathbf{k}_V^- \left(\dot{\mathbf{w}}_{n+1} - \widehat{\dot{\mathbf{w}}}\right) + \left(\mathbf{f}_{n+1} - \widehat{\mathbf{f}}\right)^T \mathbf{k}_V^{-1} \left(\mathbf{f}_{n+1} - \widehat{\mathbf{f}}\right)}{\left(\dot{\mathbf{w}}_{n+1} + \widehat{\dot{\mathbf{w}}}\right)^T \mathbf{k}_V^- \left(\dot{\mathbf{w}}_{n+1} + \widehat{\dot{\mathbf{w}}}\right) + \left(\mathbf{f}_{n+1} + \widehat{\mathbf{f}}\right)^T \mathbf{k}_V^{-1} \left(\mathbf{f}_{n+1} + \widehat{\mathbf{f}}\right)} \tag{12}$$

3.2 Non-invasive multi-scale linear stage

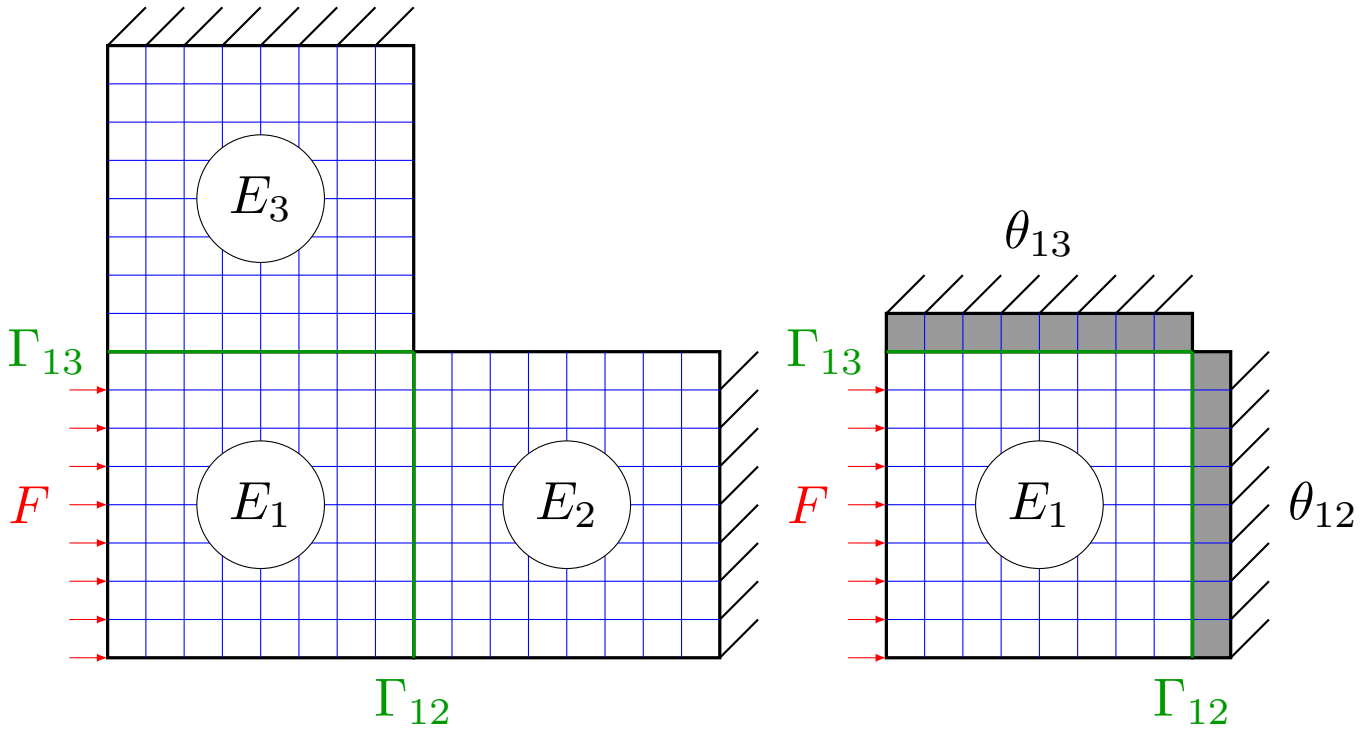
3.2.1 Non-invasive linear stage

When combining the linear equations (8) of $\mathcal{A}$ together with the linear search direction (10), it comes out that solving the linear stage consists in solving independent Robin problems on the subdomains:

$$\left(\mathbf{K}+\mathbf{N}^T\frac{\mathbf{k}_V^-}{\Delta t}\mathbf{N}\right)\mathbf{u}^{t+\Delta t}=\mathbf{f}_d^{t+\Delta t}+\mathbf{N}^T\left(\widehat{\mathbf{f}}^{t+\Delta t}+\mathbf{k}_V^-\widehat{\dot{\mathbf{w}}}^{t+\Delta t}+\frac{\mathbf{k}_V^-}{\Delta t}\mathbf{w}^t\right) \tag{13}$$

$$\text{Then compute } \begin{cases}\dot{\mathbf{w}}^{t+\Delta t} & =\mathbf{N}\dot{\mathbf{u}}^{t+\Delta t}=\mathbf{N}(\mathbf{u}^{t+\Delta t}-\mathbf{u}^t)/\Delta t\\ \mathbf{f}^{t+\Delta t} & =\widehat{\mathbf{f}}^{t+\Delta t}+\mathbf{k}_V^-\left(\widehat{\dot{\mathbf{w}}}^{t+\Delta t}-\dot{\mathbf{w}}^t\right)\end{cases} \tag{14}$$

The search direction leads to a modification of the stiffness operator $\mathbf{K}$ by $\mathbf{N}^T\frac{\mathbf{k}_V^-}{\Delta t}\mathbf{N}$ which is a non-standard term in industrial finite element software. In [23], we proposed a particular choice of the search direction to facilitate its implementation. The idea is to build the Robin extra stiffness by adding a layer of elements on the interface. This layer of elements will be also called a sole, it is illustrated on Figure 3. The intensity of the Robin parameter is adjusted by modifying the mechanical properties of the sole.



(a) Example with 3 sudomains

(b) Subdomain with soles

Figure 3: Example of a subdomain with its soles

3.2.2 Multi-scale approach

In order to make the method scalable, long range effects need to be propagated, this is done via a multiscale extension [31, 32, 29]. We briefly recall the algorithm detailed in [24]. The idea is to verify a weak form of the equilibrium of interface forces during the linear stage. Because of Saint-Venant's principle, a macro basis of displacements $\mathbb{W}$ containing rigid

body motions is chosen, completed with simple deformation modes as extension or shearing modes. This new condition is written:

$$\mathbb{W}^T \mathbf{A}\mathbf{f}^{t+\Delta t} = 0 \tag{15}$$

and it is taken into account in the search direction through the Lagrange's multiplier α:

$$\mathbf{f}^{t+\Delta t} - \widehat{\mathbf{f}}^{t+\Delta t} + \mathbf{k}_V^- \left(\dot{\mathbf{w}}^{t+\Delta t} - \widehat{\dot{\mathbf{w}}}^{t+\Delta t} \right) + \mathbf{k}_V^- \mathbf{A}^T \mathbb{W} \alpha = 0 \tag{16}$$

Thus the problem to be solved at the linear stage consists in computing $\left(\dot{\mathbf{w}}^{t+\Delta t}, \mathbf{f}^{t+\Delta t}\right)$ knowing $\left(\dot{\mathbf{w}}^{t}, \mathbf{f}^{t}\right)$ and $\left(\widehat{\dot{\mathbf{w}}}^{t+\Delta t}, \widehat{\mathbf{f}}^{t+\Delta t}\right)$ verifying the following system:

$$\begin{aligned}
&\mathbf{K}\mathbf{u}^{t+1} = \mathbf{f}_d + \mathbf{N}^T \mathbf{f}^{t+\Delta t} \\
&\mathbb{W}^T \mathbf{A}\mathbf{f}^{t+\Delta t} = 0 \\
&\mathbf{w}^{t+\Delta t} = \mathbf{N}\mathbf{u}^{t+1} \\
&\mathbf{f}^{t+\Delta t} - \widehat{\mathbf{f}}^{t+\Delta t} + \mathbf{k}_V^- \left(\dot{\mathbf{w}}^{t+\Delta t} - \widehat{\dot{\mathbf{w}}}^{t+\Delta t} \right) + \mathbf{k}_V^- \mathbf{A}^T \mathbb{W} \alpha = 0 \\
&\dot{\mathbf{w}}^{t+\Delta t} = \left(\mathbf{w}^{t+\Delta t} - \mathbf{w}^{t} \right) / \Delta t
\end{aligned} \tag{17}$$

The algorithm to solve this system is fully detailed in [24], beside extra communications and matrix operations during the initialization, it involves one extra global reduction per iteration.

3.3 Description of the local stage

Two types of interfaces are considered in our studies: perfect interfaces describing an interaction between subdomains of a same part, and contact interfaces describing physical interaction of subdomains of different parts. These two kinds of interfaces lead to two specific solving.

3.3.1 Perfect interface

For the perfect interfaces the problem consists in finding $\left(\widehat{\dot{\mathbf{w}}}^{t+\Delta t}, \widehat{\mathbf{f}}^{t+\Delta t}\right)$ knowing $\left(\widehat{\dot{\mathbf{w}}}^{t}, \widehat{\mathbf{f}}^{t}\right)$ and $\left(\dot{\mathbf{w}}^{t+\Delta t}, \mathbf{f}^{t+\Delta t}\right)$ verifying the system:

$$\begin{aligned}
&\mathbf{A}\widehat{\mathbf{f}}^{t+\Delta t} = 0 && \text{Equilibrium of forces} \\
&\mathbf{B}\widehat{\dot{\mathbf{w}}}^{t+\Delta t} = 0 && \text{Continuity of displacement} \\
&\widehat{\mathbf{f}}^{t+\Delta t} - \mathbf{f}^{t+\Delta t} - \mathbf{k}_V^+ \left(\widehat{\dot{\mathbf{w}}}^{t+\Delta t} - \dot{\mathbf{w}}^{t+\Delta t} \right) = 0 && \text{Search direction}
\end{aligned} \tag{18}$$

The solution is explicit and equal to:

$$\begin{aligned}
\widehat{\dot{\mathbf{w}}}^{t+\Delta t} &= \mathbf{A}^T \left(\mathbf{A}\mathbf{k}_V^+ \mathbf{A}^T \right)^{-1} \left[\mathbf{A}\mathbf{k}_V^+ \dot{\mathbf{w}}^{t+\Delta t} - \mathbf{A}\mathbf{f}^{t+\Delta t} \right] \\
\widehat{\mathbf{f}}^{t+\Delta t} &= \mathbf{B}^T \left(\mathbf{B}\mathbf{k}_V^{+\,-1} \mathbf{B}^T \right)^{-1} \left[\mathbf{B}\mathbf{k}_V^{+\,-1} \mathbf{f}^{t+\Delta t} - \mathbf{B}\dot{\mathbf{w}}^{t+\Delta t} \right]
\end{aligned} \tag{19}$$

The computation of the velocity consists in solving a problem on two assembled soles with a Neumann conditions on the interface between them.

3.3.2 Contact interface

For the contact interface the problem consists in finding $\left(\widehat{\dot{\mathbf{w}}}^{t+\Delta t}, \widehat{\mathbf{f}}^{t+\Delta t}\right)$ knowing $\left(\widehat{\dot{\mathbf{w}}}^{t}, \widehat{\mathbf{f}}^{t}\right)$ and $\left(\dot{\mathbf{w}}^{t+\Delta t}, \mathbf{f}^{t+\Delta t}\right)$ verifying the system:

$$\begin{aligned}
&\mathbf{B}_n \widehat{\mathbf{w}}_n^{t+\Delta t} + \mathbf{j}_n \geqslant 0 && \text{Non interpenetration}\\
&\mathbf{A}\widehat{\mathbf{f}}^{t+\Delta t} = 0 && \text{Equilibirum of force}\\
&\mathbf{B}_n \widehat{\mathbf{f}}_n^{t+\Delta t} \geqslant 0 && \text{Positive reaction force}\\
&\mathbf{B}_\tau \widehat{\mathbf{f}}_\tau^{t+\Delta t} = 0 && \text{If frictionless contact}\\
&\left(\mathbf{B}_n \widehat{\mathbf{f}}_n^{t+\Delta t}\right)^T \left(\mathbf{B}_n \widehat{\mathbf{w}}_n^{t+\Delta t} + \mathbf{j}_n\right) = 0 && \text{Signorini conditions}\\
&\widehat{\mathbf{f}}^{t+\Delta t} - \mathbf{f}^{t+\Delta t} - \mathbf{k}_V^+ \left(\widehat{\dot{\mathbf{w}}}^{t+\Delta t} - \dot{\mathbf{w}}^{t+\Delta t}\right) = 0 && \text{Search direction}\\
&\widehat{\mathbf{w}}^{t+\Delta t} = \widehat{\mathbf{w}}^{t} + \Delta t \widehat{\dot{\mathbf{w}}}^{t+\Delta t}, \qquad \widehat{\mathbf{w}}^0 \text{ given} && \text{Time integration}
\end{aligned} \tag{20}$$

where $\mathbf{j}_n$ is an initial gap. Choosing $\mathbf{j}_n$ to be negative is one of the possibilities to simulate initial tension. We assume that the displacements and forces are written in a normal/tangential basis at each node of the interface, the n and τ subscripts refer to the normal and tangential components, respectively.

In the case of frictional contact the tangential condition $\mathbf{B}_\tau \widehat{\mathbf{f}}_\tau^{t+\Delta t} = 0$ is replaced by the Coulomb relations which separate the sticking (first line) and sliding (second line) cases:

$$\begin{cases}
\|\widehat{f}_\tau^{t+\Delta t}\| < \mu |\widehat{f}_n^{t+\Delta t}| \text{ then } B_\tau \widehat{\dot{w}}_\tau^{t+\Delta t} = 0\\
\|\widehat{f}_\tau^{t+\Delta t}\| = \mu |\widehat{f}_n^{t+\Delta t}| \text{ then } \exists \lambda > 0,\ B_\tau \widehat{\dot{w}}_\tau^{t+\Delta t} = -\lambda \left(B_\tau B_\tau^T\right)^{-1} B_\tau \widehat{f}_\tau^{t+\Delta t}
\end{cases} \tag{21}$$

where the use of normal (non bold) type refers to node-wise quantities, and μ is Coulomb's coefficient of friction.

It is possible to define two node-wise indicators C_n and G_τ which permit to determine the status of nodes at the interfaces: C_n gives information about the normal status (contact); in case of contact, G_τ gives information about the tangential status (sticking/sliding). They are computed with known quantities and once they are determined the solution is explicit. The details of the solutions can be found in [23].

4 Parallel implementation

4.1 General environment of the developed tools

The chosen paradigm is to develop only with tools used by our industrial partners EDF and Airbus. In particular, we use *code_aster* finite element software [33] and *salome platform* to generate and manage meshes within the MED format. These two piecses of software are driven by python scripts permitting to develop a generic process, for the management of meshes, the mechanical computation and the post-processing.

In this part we describes the ideas of *code-aster* until the version 14. The further version may change the implementation of *code-aster*. The scripts are written in *python* with specific syntax to call the commands of *code-aster*. As the source of *code-aster* is written in *fortran* mostly, it includes a so-called *supervisor* (Figure 4) whose goal is to make the communications between the *python* and the *fortran*. The memory is shared with *python*. However, contrary to *python*, the aster objects are global in memory which lead to specific implementations that can be found in [34].

4.2 Ideas of the developed tools

The tool we developed is composed of two parts:

- Preparation: starting from a monolithic mesh given by our industrial partners, the script detects all the group of nodes and elements needed for the further computations and distribute the data into individual files for each subdomain (data of the subdomain and its interfaces). If needed parts are subdivided into smaller subdomains.

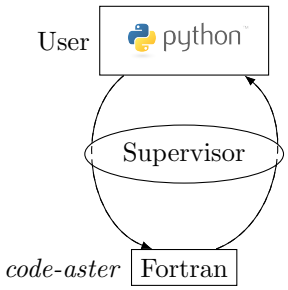


Figure 4: Principle of the supervisor

- Computation: parallel implementation of the Latin method which uses as input the meshes and persistent python objects previously created.

Remark. The recognition of the neighboring subdomains and the automatic construction of the interfaces and soles are crucial to have an automatic pre-processing phase. We have made the choice to detect the potential neighbors through the common boundary nodes of the subdomains. This easily permits to construct the interfaces and the associated soles. The automatic detection of neighbors for non-conforming interfaces is definitely not possible with that method and would require more evolved strategies.

An XML input file permits to define options and parameters both for the method and the structure. It is used to drive these two parts of the tools. This implementation is summed up in Figure 5.

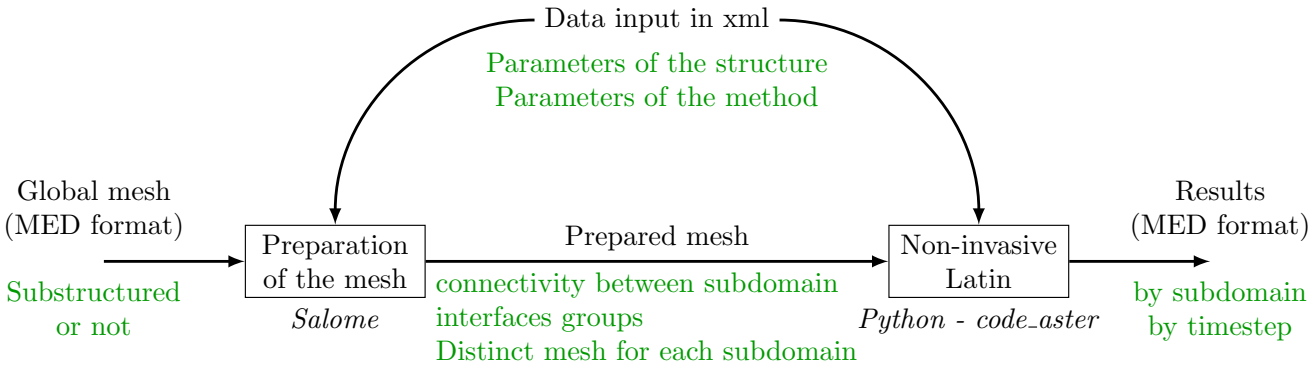


Figure 5: Description of the global implementation

4.3 Parallel processing

The non-invasive tool has been developed to launch parallel instances of python, each one driving a sequential version of the finite element software *code_aster* [33]. The parallel instances of python are synchronized by the MPI protocol through the python package *mpi4py* [35]. Each MPI instance stands for a subdomain with its own interfaces. This principle is illustrated in Figure 6 with a simple four subdomain decomposition.

At the beginning of the computation, exchanges are required in order to assemble the coarse problem on each subdomain. Then each iteration of the Latin method is highly parallel and involves only one all-neighbors and one all-reduce communications:

1. The local is decomposed in two steps:

   (a) An all-neighbor communication to exchange interface fields.

(b) A parallel computation to solve the local problems corresponding to its interfaces. Note that our implementation does not fully exploit the point-wise nature of the local equations and a finer degree of parallelism could be achieved (using one thread per interface node for instance).

2. The linear stage is decomposed in four steps:

   (a) A parallel computation on each subdomain with the Robin condition.

   (b) A global reduction to determine the macro lack-of-balance.

   (c) A parallel computation to solve a coarse problem and to get the macro correction to be added to the displacement field.

Remark. In our case only sequential version of *code-aster* are used for each subdomain. One could think about calling parallel instance of *code-aster* for each subdomain, multi-threading or parallel solver included in *code-aster* would be a next step to improve the performance.

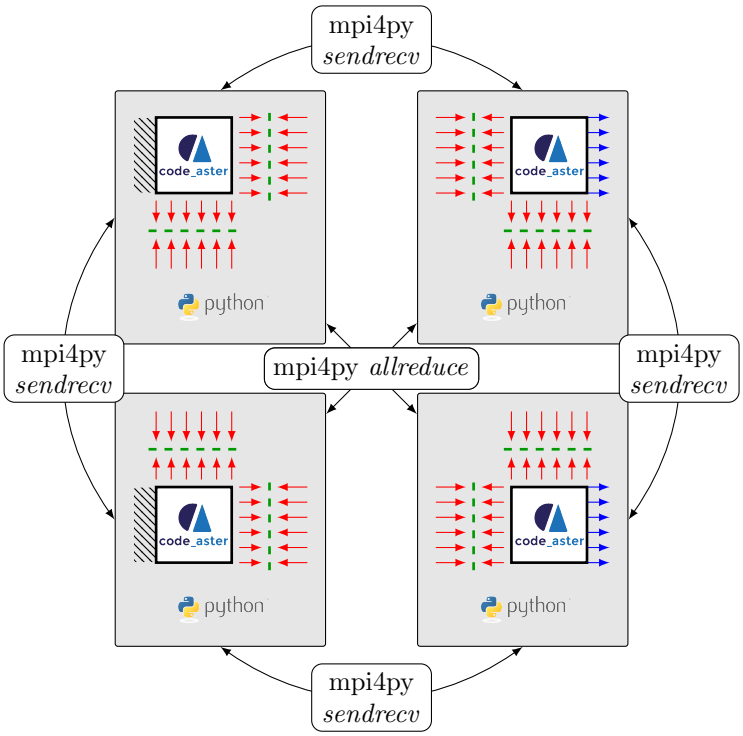


Figure 6: Principle of the implementation

Some results of scalability and computation time on academic test cases can be found in [24].

4.4 Automatic choice of search direction

The search direction, or in other word the Robin condition, is a parameter which strongly influences the convergence of the algorithm. It should represent the stiffness of the complement of the subdomain in the whole structure.

We propose a simple and automatic procedure to set this parameter for each interface. For parallel efficiency reason, our heuristic is only based on local data readily available to the processor and it does not require communication. Our method aims at having the interfaces develop a stiffness comparable to the subdomain's in their normal direction. Proposing more elaborate strategy will be the subject of future work.

Let us thus consider one interface of one subdomain. In our non-invasive implementation, the parameter of search direction is constructed by weighting the Young modulus of the sole whereas the Poisson coefficient is chosen to be the

same as for the subdomain. During the preparation step, bounding boxes of soles and subdomains are determined (see Figure 7). Let $\Delta\text{SD} = [\Delta X, \Delta Y, \Delta Z]$ denote the bounding box of the subdomain, and $\Delta s = [\Delta x, \Delta y, \Delta z]$ the sole's. The mean normal to the interface is determined and denoted by $\underline{n}$. The Young modulus of the sole E_s is chosen with respect to the Young modulus of the subdomain E_{SD} as follows:

$$\frac{E_s}{|\Delta s \cdot \underline{n}|} = \frac{E_{\text{SD}}}{|\Delta\text{SD} \cdot \underline{n}|} \tag{22}$$

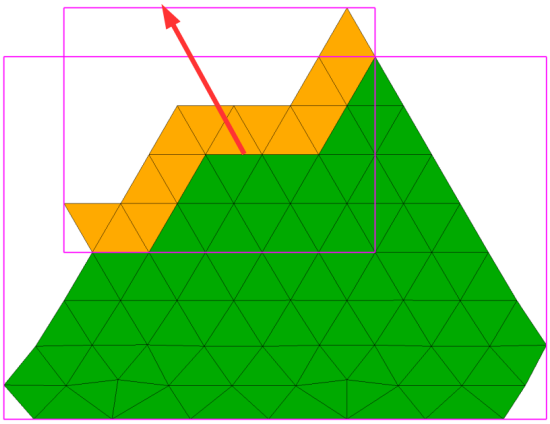


Figure 7: Example of *bounding box* with the subdomain in green and the sole in orange and the mean normal of the interface

Remark. This automatic choice of weighting of the Young modulus does not work with all configurations and sometimes a manual choice is required. For example if the interface is circular, the mean normal is null. Moreover for warped subdomains, the bounding box may not represent the subdomain and the stiffness may not be approximated correctly.

5 Numerical examples

In this section, two numerical examples from industrial structures are presented. The first one corresponds to a simplified joint of a spatial launcher under two cycles of load in order to illustrate the accumulated sliding at the frictionous interfaces. The second one is a joint which assembles the wings to the fuselage of an Airbus aircraft. With this example we illustrate the issue of overloaded subdomains which slow down the computation and we propose a simple load balancing strategy by re-decomposition.

5.1 Simplified joint of spatial launcher

The objective of this example is to illustrate the behavior of a joint composed of three bolts under two cycles of loading-unloading. It involves 30 subdomains linked through 53 interfaces (Figure 8). The three screws are decomposed in two parts linked with preloaded interfaces. This permits to ensure an initial state of compression between flanges. Contact interfaces are considered in the central part of the structure whereas perfect interfaces are chosen in the rest of the structure.

The structure is clamped on one side and a traction load is applied on the opposite side. The load follows two cycles as presented in Figure 9. 17 time steps are considered. The structure is discretized into 3 million degrees of freedom. All the parameters are given in Table 1.

The evolution of the error indicator is shown in Figure 10. Even though the multiscale approach speeds up the convergence at the first iteration, its effect decreases very quickly. The many non-linearities of frictional contact as well as the many timesteps slow down the convergence: more than 120 iterations are required to reach a criterion of 10^{-5}.

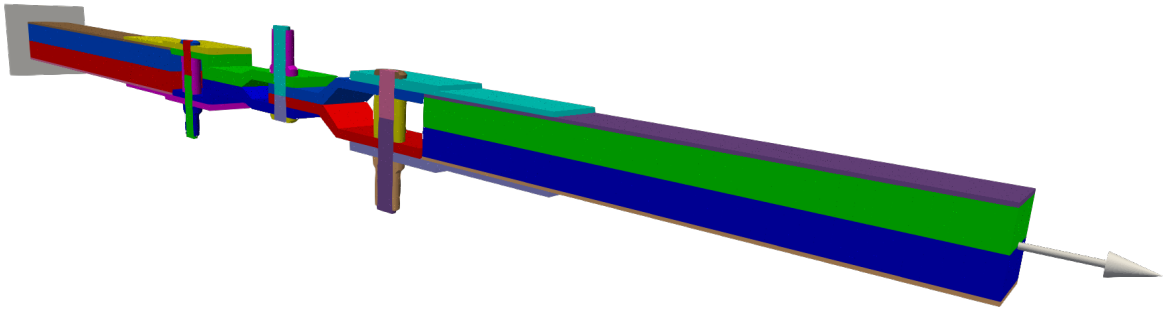


Figure 8: Description of the joint

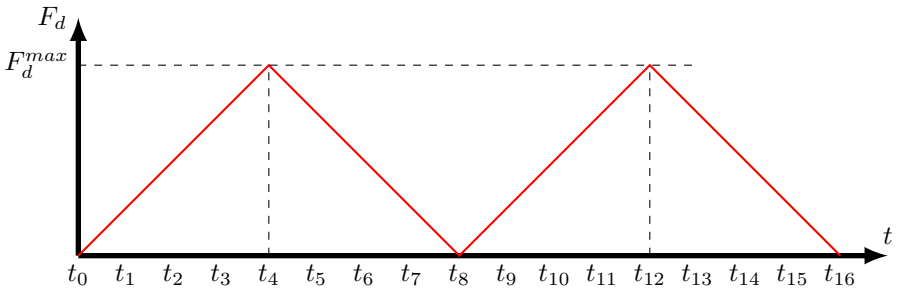


Figure 9: Load history

Parameters	Values
Young modulus E	100 GPa
Poisson ratio ν	0.3
F_d^{max}	10 MPa
preload (left - middle - right)	0.3 - 0.2 - 0.4 mm
Δt	1s
dofs	3 millions

Table 1: Parameters

In Figure 11, we observe the preload of bolts at the first timestep. The screws are solicited with tensile stress whereas compression is observed in the surrounding plates.

Von Mises stress at the timesteps t_4 and t_{12} (when the structure is the most loaded) is represented in Figure 12. Sliding happens between plates and it causes shear stress in the screws, resulting in the structure being more solicited at the 12^{th} timestep than at the 4^{th}.

Figure 13 presents the evolution of the longitudinal displacement at the end of the structure as a function of the load, we observe the accumulated sliding. During the loading phase, sliding appears and remains during the unloading phase.

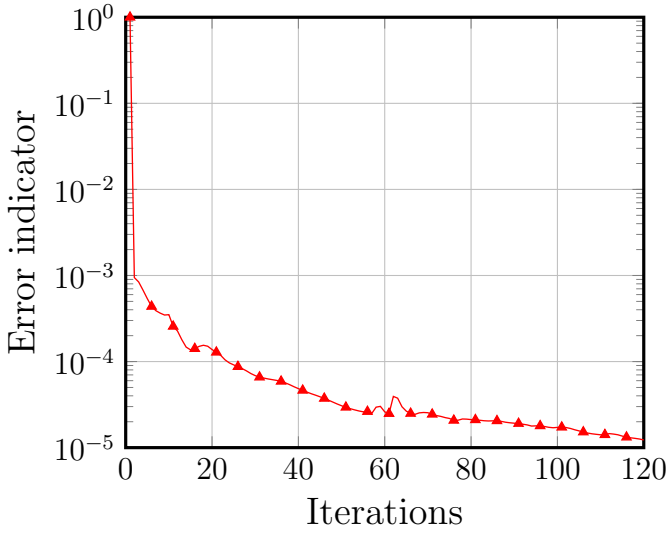


Figure 10: Evolution of the error indicator

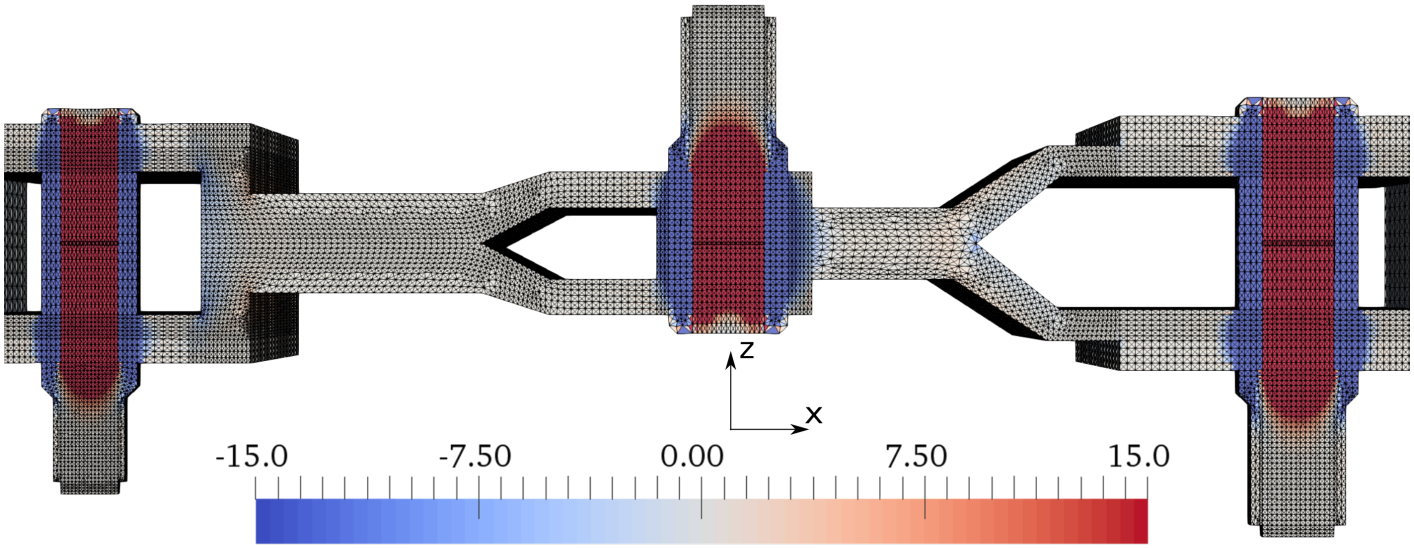


Figure 11: Preload of bolts - stress σ_{zz} (MPa)

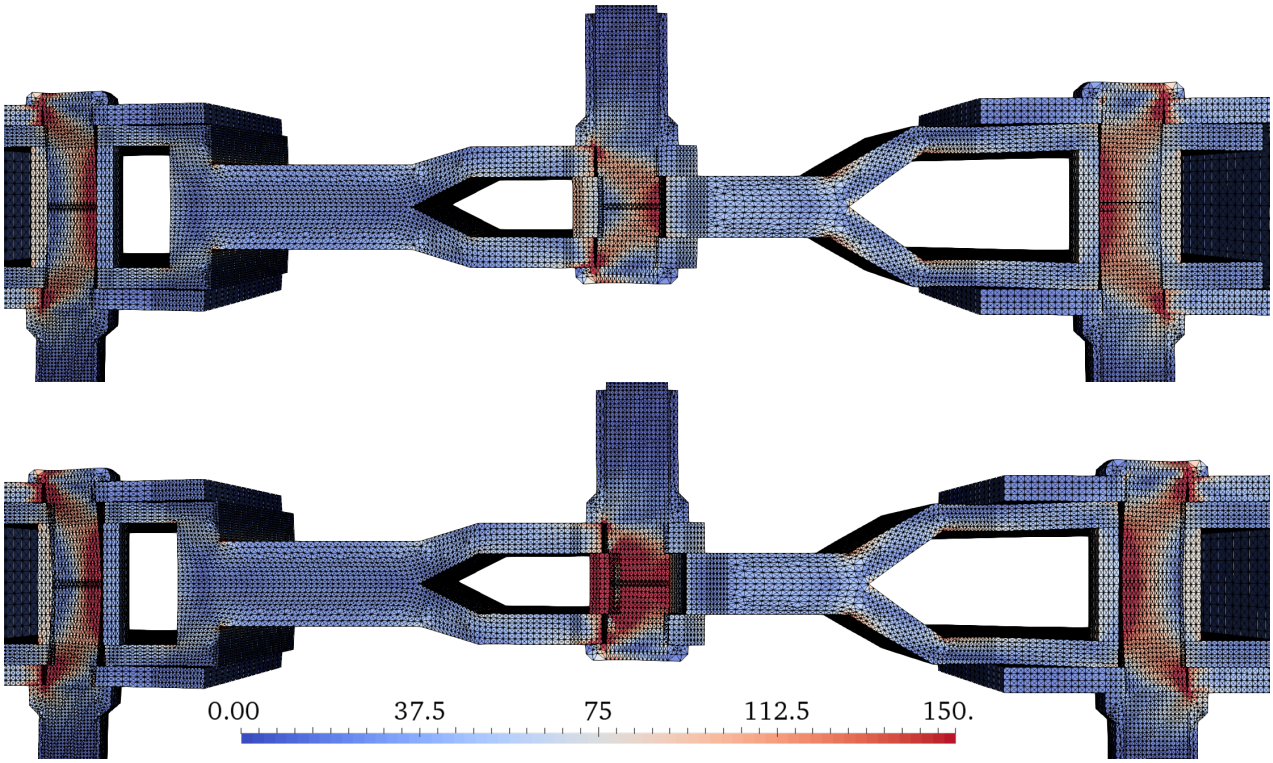


Figure 12: Von Mises stress (MPa) at timesteps t_4 (top) and t_{12} (bottom).

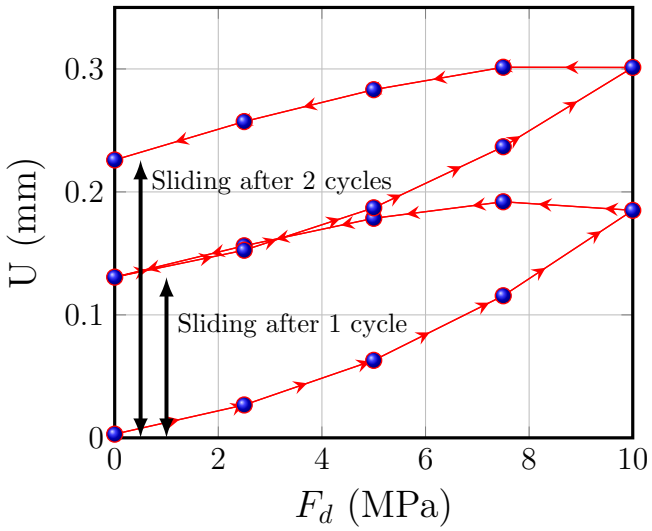


Figure 13: Mean longitudinal displacement at the end of the structure

5.2 Airbus flange

5.2.1 Initial substructuring

This case permits to illustrate the feasibility of a resubstructuring in order to balance the size of subdomains. The structure was provided by Airbus. It represents the joint between the wings and the fuselage of an aircraft. The whole structure is

composed of 47 subdomains and 89 interfaces, mostly frictionous. 10 bolts are considered. The structure is presented in Figure 14. It is clamped at the bottom and at the right. A traction load is imposed on the left. The screws and nuts are in titanium, whereas the rest of the structure is in aluminum. The load consists in one loading-unloading cycle discretized in 9 timesteps. All parameters are given in Table 2.

70 iterations are required to reach an error indicator around 10^{-6}. We present in Figure 15 the stress of Von Mises for some timesteps. Once more, an accumulated sliding can be observed.

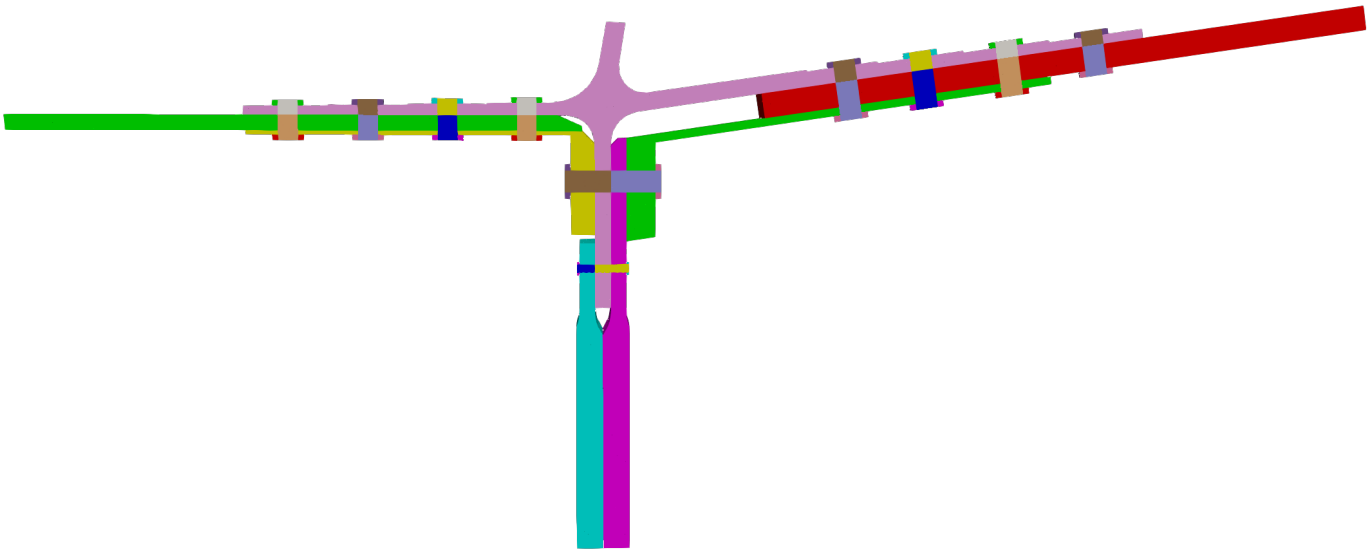


Figure 14: Description of subdomains

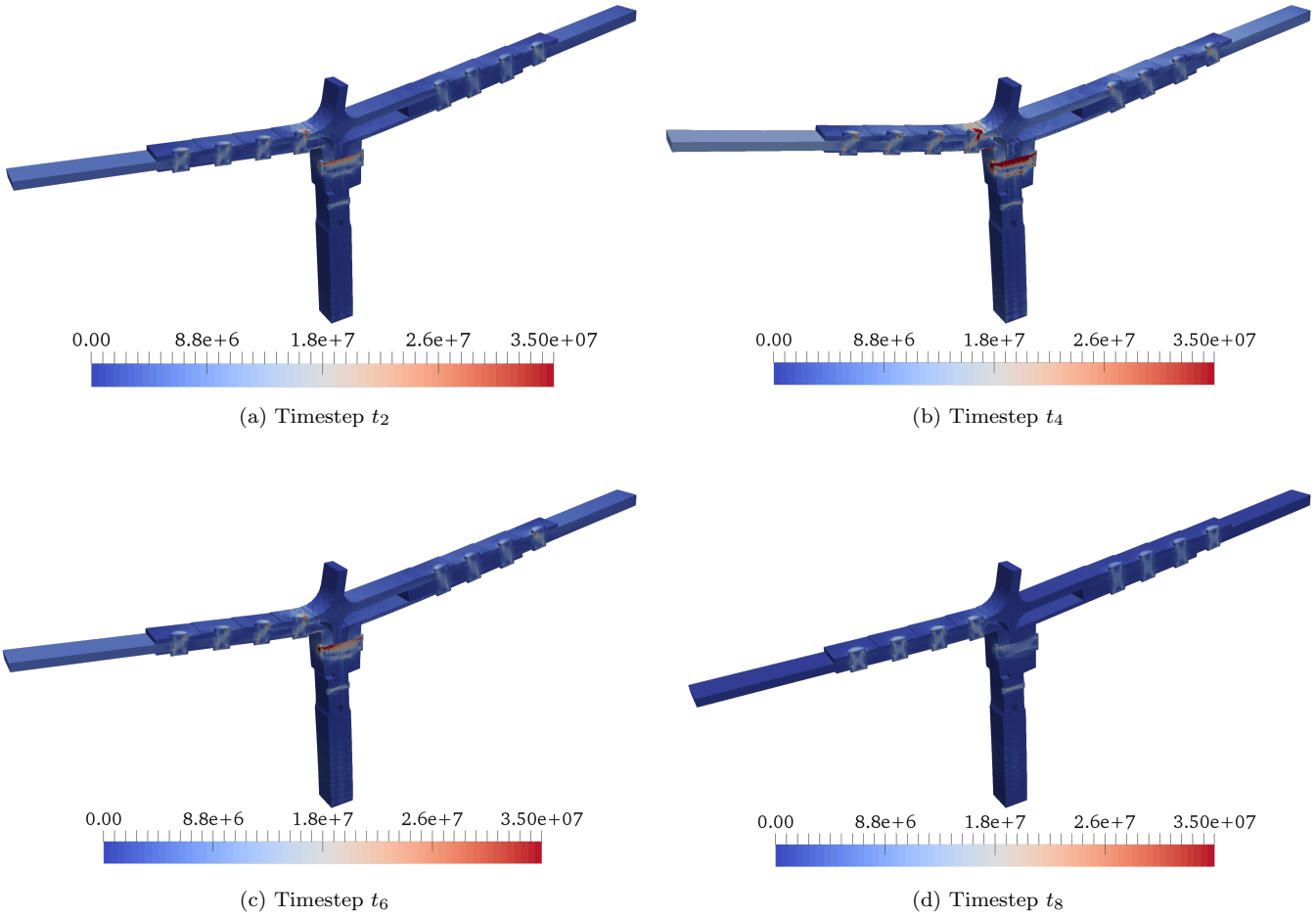
5.2.2 Re-substructuring of subdomains

Table 3 shows the number of nodes and the number of connected interfaces of the biggest subdomains. The biggest one represents almost a third of the whole structure and a quarter of the interfaces are connected to it. Thus the computation time is directly driven by this subdomain. For this computation, the total time is about 121.800 seconds. Therefore the objective is to re-decompose the large subdomains to reduce the number of degrees of freedom in each subdomain and also the number of interfaces per subdomain. The idea is to have new subdomains with as many dof as the smaller subdomains or to decompose the largest subdomains proportionally to their size in order to avoid the creation of too many subdomains. Thus the re-substructuring is limited by a user parameter M: the biggest subdomain is decomposed in at most M new ones, other subdomains are decomposed in proportion. This decomposition is realized through an existing function in code_aster based on *SCOTCH* [36] whose only parameter is the number of subdomains.

In our case, we choose $M = 10$. The new structure is composed of 69 subdomains (Figure 16). With this new decomposition, we also ran 70 iterations and compared the CPU time. It permitted to reduce the CPU time to 18.450 seconds contrary to 121.800 seconds before, representing a gain of 6.6 from the initial substructuring. Figure 17 shows the

Parameters	Values
E_{alu}	70 GPa
ν_{alu}	0.3
E_{titane}	110 GPa
ν_{titane}	0.3
F_d^{max}	5 MPa
Preload	0.005 mm
Initial gap	0.005 mm
Δt	1s
dofs	300,000

Table 2: Parameters



(a) Timestep t_2

(b) Timestep t_4

(c) Timestep t_6

(d) Timestep t_8

Figure 15: Stress of Von Mises (Pa) and deformed shape ($\times 20$)

comparison of the error indicator between the two substructuring. The convergence of the first iterations are relatively similar. The multiscale approach grants scalability to the method. However the effect of the multiscale extension is decreased by the numerous non-linearities, and differences appear in term of convergence. The new substructuring seems

Biggest subdomains	Number of nodes	Connected interfaces
1^{st}	29,107	23
2^{nd}	17,789	7
3^{rd}	14,530	6
4^{th}	11,030	10
5^{th}	10,322	12
6^{th}	6,861	6
7^{th}	2,072	4

Table 3: Size of the biggest subdomains

to reach a plateau after 20 iterations. This is simply explained by the particular substructuring provided automatically by *SCOTCH*. For example in Figure 18, we show the new decomposition of the subdomain S_0 into 10 new substructures. One can remark that some subdomains are non-connected (composed of distinct parts).

The automatic weighting of the Young modulus cannot be efficient on such subdomains as the bounding box is not representative of the subdomain.

Therefore, we decompose these problematic subdomains into new subdomains. With this, we obtain a decomposition of the whole structure into 72 subdomains. Such a decomposition is much more adapted to the automatic weighting of the search direction and we expect a better convergence. Therefore, the error indicator no longer reaches a plateau and the convergence follows the initial one but 6.6 times faster (Figure 17)

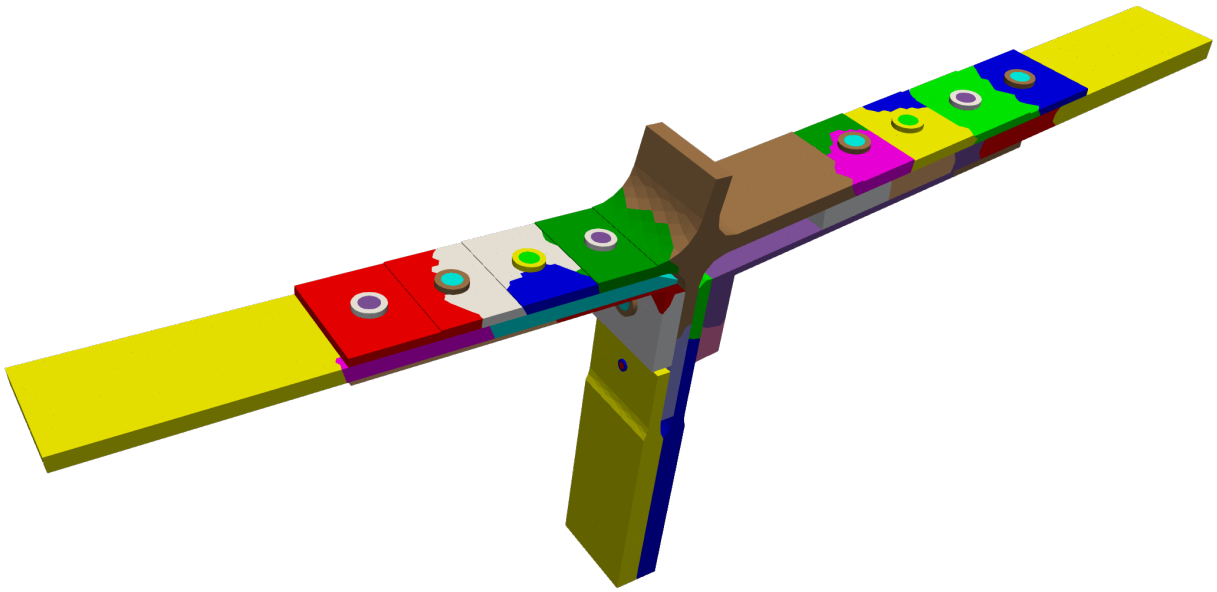


Figure 16: Re-decomposition of the structure

6 Conclusion

In this paper we presented a multiscale non-invasive implementation of a mixed domain decomposition method. This implementation has been confronted to industrial structures with quasi-static and small strain assumptions, with many frictional contact interfaces.

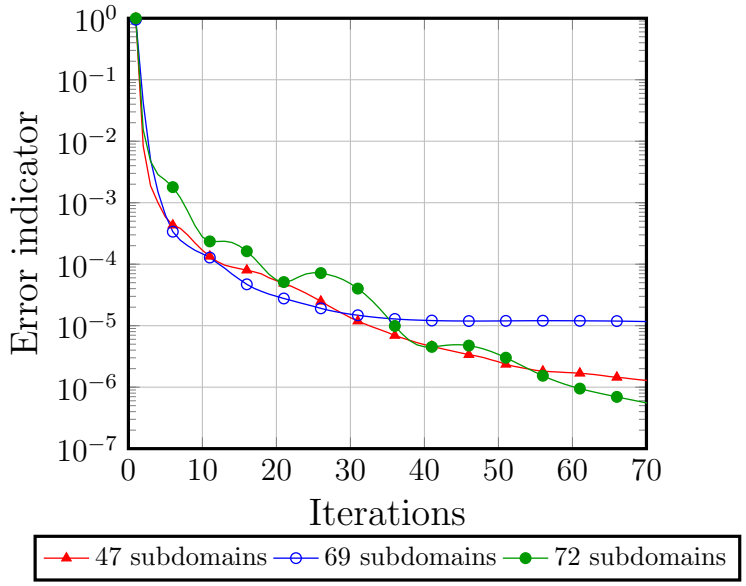


Figure 17: Comparison of the error indicator

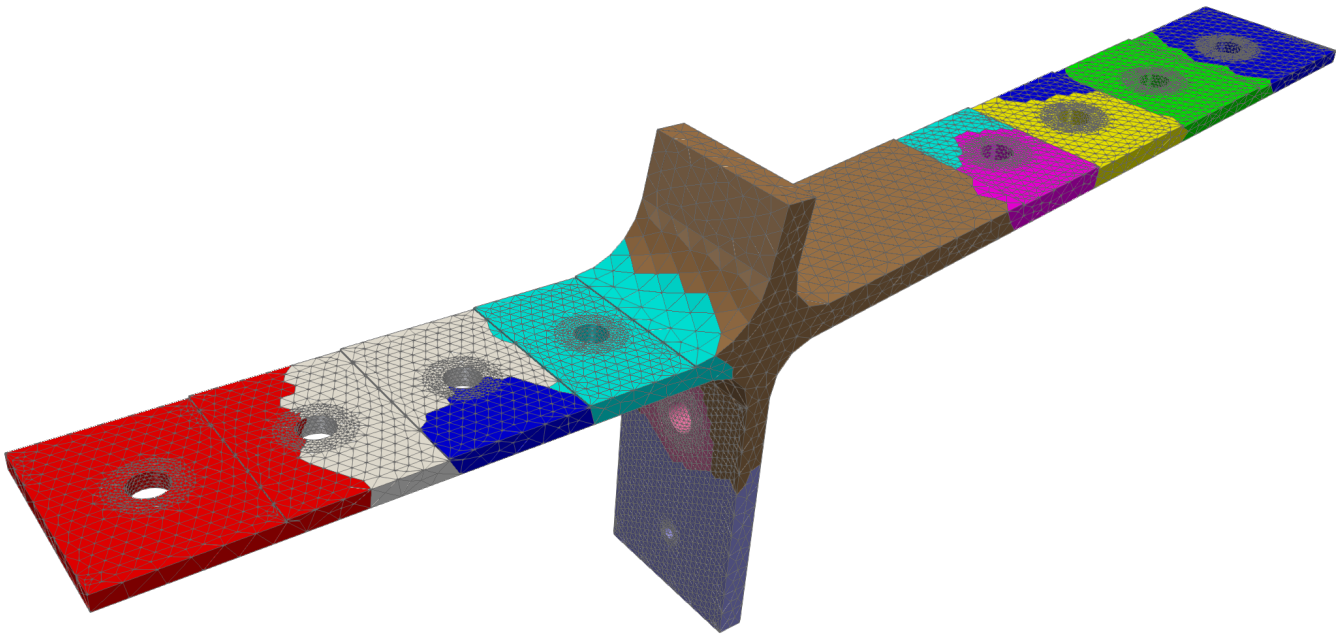


Figure 18: Decomposition of the subdomain S_0

The non-invasive implementation of the mixed conditions is realized by adding of a layer of elements at the interface. We briefly reminded the non-invasive multiscale approach detailed in [24] and the treatment of frictional contact [28, 23].

Moreover we presented an automatic setting of the Robin parameter which is suited for non warped and homogeneous subdomains. Although this permits to avoid a manual parametrization of all search directions, we exhibited some limita-

tions for certain shapes of subdomain. We also confronted the method to semi-industrial cases involving bolted assemblies with many frictional contact interfaces.

The next step will be to investigate the weighting of search direction in the case of heterogeneous materials in subdomains. The choice of only one Young modulus may not be efficient enough to ensure robust convergence. This needs also to be coupled with spectral coarse spaces such as proposed in GenEO approach [1].

Acknowledgments

We thank Airbus and EDF for their financial supports and technical help with *code-aster*. We also thank the project ECOS-CONICYT C17E04 for their financial support.

References

[1] N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, R. Scheichl, Abstract robust coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps, Numer. Math. 126 (4) (2014) 741–770. doi:10.1007/s00211-013-0576-y.

[2] N. Spillane, D. J. Rixen, Automatic spectral coarse spaces for robust FETI and BDD algorithms, International Journal for Numerical Methods in Engineering 95 (11) (2013) 953–990. doi:10.1002/nme.4534.

[3] A. Klawonn, P. Radtke, O. Rheinbach, Adaptive Coarse Spaces for BDDC with a Transformation of Basis, in: Domain Decomposition Methods in Science and Engineering XXII, Vol. 104 of Lecture Notes in Computational Science and Engineering, Springer International Publishing, Cham, 2016, pp. 301–309. doi:10.1007/978-3-319-18827-0_29.

[4] A. Klawonn, M. Kühn, O. Rheinbach, Adaptive coarse spaces for FETI-DP in three dimensions, SIAM J. Sci. Comput. 38 (5) (2016) A2880–A2911. doi:10.1137/15M1049610.
URL http://dx.doi.org/10.1137/15M1049610

[5] L. Beirão da Veiga, L. F. Pavarino, S. Scacchi, O. B. Widlund, S. Zampini, Isogeometric BDDC preconditioners with deluxe scaling, SIAM J. Sci. Comput. 36 (3) (2014) A1118–A1139. doi:10.1137/130917399.
URL http://dx.doi.org/10.1137/130917399

[6] C. Bovet, A. Parret-Fréaud, N. Spillane, P. Gosselet, Adaptive multipreconditioned FETI: scalability results and robustness assessment, Computers & Structures 193 (2017) 1–20. doi:10.1016/j.compstruc.2017.07.010.

[7] C. Bovet, P. Gosselet, N. Spillane, Multipreconditioning for nonsymmetric problems: the case of orthomin and biCG, Comptes rendus de l'académie des sciences (math.) 355 (3) (2017) 354–358. doi:10.1016/j.crma.2017.01.010.

[8] L. Badea, F. Lebon, Schwarz method for dual contact problems, Computational and Applied Mathematics 36 (1) (2017) 719–731. doi:10.1007/s40314-015-0255-y.

[9] M. Barboteu, P. Alart, M. Vidrascu, A domain decomposition strategy for nonclassical frictional multi-contact problems, Computer Methods in Applied Mechanics and Engineering 190 (2001) 4785–4803.

[10] Z. Dostál, D. Horák, Scalability and FETI based algorithm for large discretized variational inequalities, Mathematics and Computers in Simulation 61 (3) (2003) 347–357. doi:10.1016/S0378-4754(02)00088-5.

[11] Z. Dostál, D. Horák, Scalable FETI with optimal dual penalty for a variational inequality, Numerical linear algebra with applications 11 (56) (2004) 455–472.

[12] Z. Dostál, D. Horák, R. Kučera, V. Vondrák, J. Haslinger, J. Dobiáš, S. Pták, FETI based algorithms for contact problems: scalability, large displacements and 3D Coulomb friction, Computer Methods in Applied Mechanics and Engineering 194 (2-5) (2005) 395–409. doi:10.1016/j.cma.2004.05.015.

[13] Z. Dostál, T. Kozubek, T. Brzobohaty, A. Markopoulos, O. Vlach, Scalable TFETI with optional preconditioning by conjugate projector for transient frictionless contact problems of elasticity, Computer Methods in Applied Mechanics and Engineering 247-248 (2012) 37–50. doi:10.1016/j.cma.2012.08.003.

[14] P. Ladevèze, Nonlinear computational structural mechanics: new approaches and non-incremental methods of calculation, Springer-Verlag, New-York, 1999. doi:10.1007/978-1-4612-1432-8.

[15] P.-L. Lions, On the Schwarz alternating method. III: a variant for nonoverlapping subdomains, in: Third international symposium on domain decomposition methods for partial differential equations, Vol. 6, SIAM Philadelphia, PA, 1990, pp. 202–223.

[16] M. J. Gander, Optimized Schwarz Methods, SIAM Review 44 (2) (2006) 699–731.

[17] L. Gendre, O. Allix, P. Gosselet, Non-intrusive and exact global/local techniques for structural problems with local plasticity, Computational Mechanics 44 (2009) 233–245. doi:10.1007/s00466-009-0372-9.

[18] J.-C. Passieux, J. Réthoré, A. Gravouil, M.-C. Baietto, Local/global non-intrusive crack propagation simulation using a multigrid x-fem solver, Computational Mechanics 52 (6) (2013) 1381–1393.

[19] G. Guguin, O. Allix, P. Gosselet, S. Guinard, Nonintrusive coupling of 3D and 2D laminated composite models based on finite element 3D recovery, International Journal for Numerical Methods in Engineering 98 (5) (2014) 324–343. doi:10.1002/nme.

[20] G. Guguin, O. Allix, P. Gosselet, S. Guinard, On the computation of plate assemblies using realistic 3D joint model: a non-intrusive approach, Advanced Modeling and Simulation in Engineering Sciences 3 (1) (2016) 16. doi:10.1186/s40323-016-0069-5.

[21] M. Duval, J. C. Passieux, M. Salaün, S. Guinard, Non-intrusive Coupling: Recent Advances and Scalable Nonlinear Domain Decomposition, Archives of Computational Methods in Engineering 23 (1) (2016) 17–38. doi:10.1007/s11831-014-9132-x.

[22] P. Gosselet, M. Blanchard, O. Allix, G. Guguin, Non-invasive global-local coupling as a Schwarz domain decomposition method: acceleration and generalization, Advanced Modeling and Simulation in Engineering Sciences 5 (4). doi:10.1186/s40323-018-0097-4.

[23] P. Oumaziz, P. Gosselet, P.-A. Boucard, S. Guinard, A non-invasive implementation of a mixed domain decomposition method for frictional contact problems, Computational Mechanics 60 (5) (2017) 797–812. doi:10.1007/s00466-017-1444-x.

[24] P. Oumaziz, P. Gosselet, P. A. Boucard, M. Abbas, A parallel noninvasive multiscale strategy for a mixed domain decomposition method with frictional contact, Int. J. Numer. Methods Eng. 115 (8) (2018) 893–912. doi:10.1002/nme.5830.

[25] P. Ladevèze, A. Nouy, On a multiscale computational strategy with time and space homogenization for structural mechanics, Computer Methods in Applied Mechanics and Engineering 192 (28-30) (2003) 3061–3087. doi:10.1016/S0045-7825(03)00341-4.

[26] C. Bernardi, Y. Maday, A. T. 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

[27] 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) 720–738. doi:10.1007/BF02510249.
URL link.springer.com/article/10.1007/BF02510249https://doi.org/10.1007/BF02510249

[28] C. Blanze, L. Champaney, J.-Y. Cognard, P. Ladevèze, A modular approach to structure assembly computations: application to contact problems, Engineering Computations 13 (1) (1996) 15–32. doi:10.1108/02644409610110976.

[29] P. Kerfriden, O. Allix, P. Gosselet, A three-scale domain decomposition method for the 3D analysis of debonding in laminates, Computational Mechanics 44 (2009) 343–362.

[30] K. Saavedra, O. Allix, P. Gosselet, On a multiscale strategy and its optimization for the simulation of combined delamination and buckling, Int. J. Numer. Methods Eng. 91 (7) (2012) 772–798. doi:10.1002/nme.4305.
URL http://doi.wiley.com/10.1002/nme.4305

[31] D. Dureisseix, Une Approche Multi-échelles pour des Calculs de Structures sur Ordinateurs à Architecture Parallèle, Ph.D. thesis, Ecole Normale Supérieure de Cachan (1997).

[32] P. Ladevèze, O. Loiseau, D. Dureisseix, A micro–macro and parallel computational strategy for highly heterogeneous structures, International Journal for Numerical Methods in Engineering 52 (12) (2001) 121–138.

[33] EDF R&D, Code-aster, www.code-aster.org.

[34] P. Oumaziz, Une méthode de décomposition de domaine mixte non-intrusive pour le calcul parallèle d'assemblages, Ph.D. thesis, Université Paris-Saclay - ENS Paris-Saclay (2017).
URL https://www.theses.fr/2017SACLN030

[35] L. Dalcín, R. Paz, M. Storti, J. D'Elía, MPI for Python: Performance improvements and MPI-2 extensions, J. Parallel Distrib. Comput. 68 (5) (2008) 655–662. doi:10.1016/j.jpdc.2007.09.005.

[36] F. Pellegrini, J. Roman, Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs, in: H. Liddell, A. Colbrook, B. Hertzberger, P. Sloot (Eds.), High-Performance Computing and Networking, Springer Berlin Heidelberg, Berlin, Heidelberg, 1996, pp. 493–498.