Parallel preprocessing with subdomain's shape control for domain decomposition methods

Yannis El Gharbi, Pierre Gosselet, Augustin Parret-Fréaud, Christophe Bovet

© Civil-Comp Press, 2019
Proceedings of the Sixth International Conference on
Parallel, Distributed, GPU and Cloud Computing for Engineering,
P. Iványi and B.H.V Topping (Editors)
Civil-Comp Press, Stirlingshire, Scotland

Paper 21

Parallel preprocessing with subdomain's shape control for domain decomposition methods

Y. EL GHARBI[1,2]**, P. GOSSELET**[1]**, A. PARRET-FRÉAUD**[2]**, C. BOVET**[3]
[1]LMT, École Normale Supérieure Paris-Saclay, France
[2]Safran Tech, Modeling & Simulation, Châteaufort, France
[3]Onera - The French Aerospace Lab, Châtillon, France

Abstract

Solving highly heterogeneous structural mechanic problems with a large number of degrees of freedom (HPC simulations) is a real issue in engineering work, because of the required time and memory. Non overlapping domain decomposition methods such as the Finite Element Tearing and Interconnecting (FETI) or Balanced Domain Decomposition (BDD) methods have been developed in order to allocate the problems on distributed memory clusters with a large number of processors and to make mechanical calculations parallel.

Two difficulties are encountered when applying domain decomposition methods. First, the mesh generation is most often a sequential process applied to the full domain. Second, the linear system resulting from the partitioning of the mesh may be poorly conditioned, leading to slow convergence. Recently developed techniques such as adapted coarse spaces (e.g. FETI-GenEO) or multipreconditioning (e.g. AMPFETI) enable to restore good convergence rate, at the cost of extra computations.

In this study, we try to mitigate these two difficulties by proposing a new hierarchical substructuring method which aims at making the mesh preprocessing step parallel and at improving the condition number of the linear system to be solved by generating regular interfaces adapted to the heterogeneity.

Keywords: domain decomposition methods, hierarchical substructuring, parallel computing, mesh deformation, FETI

1 Introduction

Non overlapping domain decomposition methods such as FETI (Finite Elements Tearing and Interconnecting) [1] or BDD (Balanced Domain Decomposition) [2] provide an advantageous framework to define iterative solvers for the resolution of linear and non-linear problems - derived from the discretisation of partial differential equations - adapted to the massively parallel architecture of recent supercomputers. They have been successfully applied to solve several challenging mechanical heterogenous problems such as homogeneous arterial walls

[3] or strongly heterogenous wind model [4]. An overview of these Schur complement based methods is available for instance in [5].

However, as any iterative solver, they are sensitive to the condition number of the linear system to be solved which has hindered their adoption in an industrial codes and even their use in industrial context.

Classically, solving a finite element problem with a domain decomposition approach can be sum up as follows (upper branch of Figure 2):

1. Mesh generation on the global structure, possibly of large size, with a sequential mesher (or possibly with its hard to implement parallel counterpart [6]);

2. Substructuring with a partitioning software of the associated connectivity graph (*e.g.* Metis [7], Scotch [8], Chaco [9]...) or with a Binary Space Partitioning tree of the position of elements (KD-tree [10], RP-tree, MM-tree...).

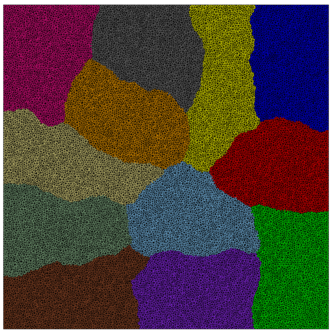


Figure 1: Substructuring of a 2D mesh coming from Metis [7] graph partitioning software.

For heterogeneous problems, frequently encountered in industrial framework, efficient preconditioning methods have been developed in the early 2000s [4, 11], in order to obtain good convergence properties if the heterogeneities do not cross the frontier between subdomains. Nevertheless, the use of automatic mesh partitioning software—for the load balancing and the minimization of cross-processes communication—leads to subdomains with slender and irregular shape and misplaced subdomains with respect to heterogeneities (see Figure 1).

Recent advances, such as algorithms of the family of AMPFETI (Adaptive Multipreconditioned-FETI) based on multipreconditioning [12, 13], or block Krylov based algorith (Block FETI [13]), or even preliminarily solved generalized eigenvalues problem (GenEO (Generalized Eigenproblems in the Overlaps) [14]), lead to an improved robustness of these solvers on very heterogeneous and ill-conditioned linear systems but their use induce higher memory consumption (Block FETI, AMPFETI) or extra computation (GenEO) remains expensive in terms of memory consumption.

On the one hand, if it seems possible to produce homogeneous subdomains with an optimization of the position of interfaces with respect to heterogeneities by a weighting of graph edges. It appears impossible to guarantee a regular geometric aspect of these interfaces because the graph partitioner software does not take into account any geometrical features. A BSP-tree (such as a KD-tree) could generate spatially localized subdomains but it could not generate regular interfaces because the mesh is not regular at the location of the interfaces.

On the other hand, this traditional approach has another weakness: the mesh and subdomains generation stages are widely driven by sequential steps which may constitute significant part of the total computational time. Specifically, the conformal mesh generation of complex 3D structures from geometrical data is a process which can become very expensive and would benefit from a parallelisation. Parallel meshing techniques are not satisfying because the refinement of interface imposes either multiple exchanges across subdomains [15] or a re-partitioning of the mesh (which creates jagged interfaces) [15, 16]. Other parallel meshing methods can create regular interfaces between subdomains thanks to an initial coarse mesh generated sequentially (this kind of methods is similar to the one developed in this paper). These methods ensure the conformal of the interface by communications between subdomains [17, 18] or thanks to a hierarchical refining [19] which ensures *a priori* this conformity. However, the shape of outside boundary surfaces is not perfectly recovered because a quadratic shape functions interpolation, which does not take into account the underlying geometry, is performed. Moreover, this kind of parallel meshers seems not to respect the components of the structure and can not produce homogeneous subdomains.

The aim of the presented method is to avoid previously mentioned difficulties by obtaining automatically, and in the most parallel way as possible, a substructuration that presents regular interfaces and that is adapted to structural heterogeneities.

The condition number of the linear systems to be solve will be improved making it possible to avoid the use of the advanced and expensive techniques.

The proposed method—which appears to be completely original in solid computational mechanics—is schematically based on a reversal of meshing and substructuring steps associated with a possibly hierarchical discretisation.

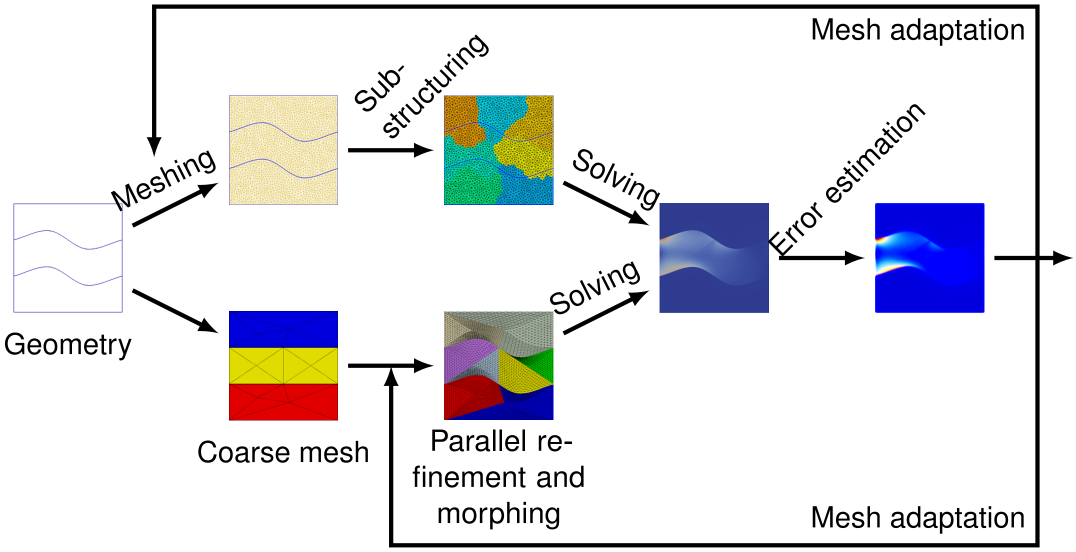


Figure 2: Domain decomposition computation approach: - Upper branch: classical method - Lower branch: developed hierarchical method.

As described on the lower branch of Figure 2, from a discrete geometry (*i.e.* CAD or discrete CAD), a first coarse mesh is generated. This coarse mesh does not represent the underlying geometry well but allow to define a partitioning which presents a compact shape because the subdomains are defined as sets of coarse elements. Then, the subdomains are distributed on the calculation cores with the underlying subdomain CAD information. The fine mesh of the structure is then carried out in parallel while keeping the interface compatibility. Finally, the application of mesh morphing techniques allows to ensure that these fine meshes respect the underlying geometry as well as possible. The developments are performed using the Python interface of the finite element software Z-set [20, 21], in which the FETI algorithm is implemented.

This article is organized as follows: Section 2 gives a reminder about the FETI method for solving substructured problems resulting from the method described in the Section 3; then numerical results on heterogeneous case are provided in Section 4, Section 5 concludes the paper.

2 FETI in a nutshell

To begin, readers are reminded of the FETI method used within the context of this work (for further details about FETI, see for instance [1, 5]). Let us consider a linear mechanical problem set on domain Ω discretised by means of the finite element method. The resulting symmetric positive definite problem $Ku = f$ can be partitioned into N subdomains $\Omega^{(s)}$ conforming to the domain Ω such that the associated problem writes $\forall s \in [\![1; N]\!]$,

$$K^{(s)}u^{(s)} = f^{(s)} + {t^{(s)}}^T {B^{(s)}}^T \lambda, \tag{1}$$

$$\sum_s B^{(s)} t^{(s)} u^{(s)} = 0, \tag{2}$$

where $t^{(s)}$ are trace operators, $B^{(s)}$ are signed Boolean assembly operators and λ a set of Lagrange multipliers describing the traction field that connects subdomains.

Let us classically define the local primal Schur complement defined on $\Omega^{(s)}$, $S_p^{(s)} = K_{bb}^{(s)} - K_{bi}^{(s)} {K_{ii}^{(s)}}^{-1} K_{ib}^{(s)}$ (i stands for internal degrees of freedom and b for boundary degrees of freedom), the local dual Schur complement $S_d^{(s)} = {S_p^{(s)}}^+ = t^{(s)} {K^{(s)}}^+ {t^{(s)}}^T$ (${K^{(s)}}^+$ is a generalized inverse of $K^{(s)}$) and the basis of rigid body motions $R^{(s)} = \text{Ker}\left(K^{(s)}\right)$. We also write:

$$e = \begin{bmatrix} \dots & {f^{(s)}}^T R^{(s)} & \dots \end{bmatrix}^T, \qquad G = \begin{bmatrix} \dots & B^{(s)} t^{(s)} R^{(s)} & \dots \end{bmatrix},$$

$$S_d = \sum_s B^{(s)} S_d^{(s)} {B^{(s)}}^T, \qquad b_d = \sum_s B^{(s)} t^{(s)} {K^{(s)}}^+ f^{(s)}.$$

The system (2) leads to the classical FETI system :

$$\begin{bmatrix} S_d & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \alpha \end{bmatrix} = \begin{bmatrix} -b_d \\ -e \end{bmatrix}. \tag{3}$$

The $G^T\lambda = -e$ constraint (this constraint is also called the coarse problem: it enables global communication between non-neighbouring subdomains) is handled by the introduction of the initial estimates λ_0 observing this condition ($G^T\lambda_0 = -e$) and a projector P on the

Algorithm 1: FETI Conjugate Gradient

```
Initialisation;
computation of the projector $P = I - QG(G^TQG)^{-1}G^T$;
solution initialisation for projection $\lambda_0 = -QG(G^TQG)^{-1}e$;
residual initialisation $r_0 = -P^T(b_d + S_d\lambda_0)$;
preconditioning of the problem $z_0 = P\tilde{S}_p r_0$ et $w_0 = z_0$;
while $\sqrt{{r_i}^T z_i} < \varepsilon$ do
    $p_i = P^T S_d w_i$;
    optimal coefficient computation $\alpha_i = \frac{{r_i}^T z_i}{{w_i}^T p_i}$;
    solution at iteration $i+1$ : $\tilde{\lambda}_{i+1} = \tilde{\lambda}_i + \alpha_i w_i$;
    new residual computation $r_{i+1} = r_i - \alpha_i p_i$;
    preconditioned residual computation $z_{i+1} = P\tilde{S}_p r_{i+1}$;
    for $0 \leqslant j \leqslant i$; do  search basis reorthogonalisation
        $\beta_i^j = -\frac{{z_{i+1}}^T p_j}{{w_i}^T p_i}$;
    end
    New direction computation: $w_{i+1} = z_{i+1} + \sum_{j=1}^{i} \beta_i^j w_j$;
end
Global solution computation;
$\lambda = \lambda_0 + P\tilde{\lambda}_i$;
$\alpha = (G^TQG)^{-1}G^TQ(-b_d - S_d\lambda)$;
```

orthogonal space to G ($G^TP = 0$). The λ unknown is searched as $\lambda = \lambda_0 + P\tilde{\lambda}$ with:

$$\begin{aligned} \lambda_0 &= -QG\left(G^TQG\right)^{-1}e, \\ P &= I - QG\left(G^TQG\right)^{-1}G^T, \end{aligned} \tag{4}$$

Then, $\tilde{\lambda}$ is the solution of the following system:

$$P^TS_dP\tilde{\lambda} = P^T(-b_d - S_d\lambda_0). \tag{5}$$

The Q matrix in (4) is symmetric positive definite and can be chosen as the $\tilde{S}_p$ preconditioner, the identity matrix or a scaling matrix [22].

The system (5), which is an equivalent of the system (3), is solved by an iterative preconditioned Conjugate Gradient method (Algorithm 1) for which the preconditioner $\tilde{S}_p$ is an assembly of primal quantities, such as:

$$\tilde{S}_p = \sum_s \tilde{B}^{(s)}\tilde{S}_p^{(s)}{\tilde{B}^{(s)}}^T, \tag{6}$$

where $\tilde{B}^{(s)}$ are scaling assembly operators such as $\sum_s B^{(s)}{\tilde{B}^{(s)}}^T = I$ and where $\tilde{S}_p^{(s)}$ can be local primal Schur complements $S_p^{(s)}$or their approximations ($K_{bb}^{(s)}$ or diag($K_{bb}^{(s)}$)). The chosen scaling in $\tilde{B}^{(s)}$ can be based on the multiplicity of interface degrees of freedom for homogeneous problems or on the diagonal coefficients of $K_{bb}^{(s)}$ for heterogeneous ones [4].

The FETI method attached with its coarse problem and the optimal preconditioner from the literature is theoretically scalable [23, 11]. The condition number κ—which is strongly related to the convergence rate—reads:

$$\kappa \simeq C\left(1+\log\left(\frac{H}{h}\right)\right)^2, \tag{7}$$

where H is the characteristic dimension of subdomains, h the size of the mesh elements and C is a constant only depending on the shape and the homogeneity of subdomains.

3 The hierarchical substructuring method

In this section, the developed substructuring strategy adapted to non overlapping domain decomposition methods is described. The aim is to produce a substructuration with the following proprieties: :

- Each subdomain must be made of a single material (in case of complex geometry, this condition can be loosen).
- The interface between subdomains must be as regular as possible (it must follows the curvature of the interface between materials or must be large pieces of planar surfaces);

From a geometry (CAD/discrete CAD), a fine surface mesh and a very coarse volumetric mesh are extracted. The fine surface mesh is a shell mesh that has a similar description to that of STL format (Stereo-Lithography); it describes an object by its outside surface faceted by triangular elements. The coarse mesh—which does not represents the actual geometry—enables to define a substructuring with an interface made of large pieces of plane surfaces.

Let ”physical edges“ denote all the edges defined on the CAD geometry, and ”physical surfaces“ denote all the surfaces from CAD delimited by a closed lined formed by physical edges. Moreover, let sets of elements composed of the same material be called physical sets.

The whole process of the hierarchical decomposition can be sum up in three steps:

1. Coarse mesh generation representing physical entities (materials);
2. Parallel mesh refinement of subdomains keeping interfaces conformity;
3. Parallel mesh morphing of subdomains in order to regain the structure geometry.

3.1 First coarse and homogeneous decomposition

From the initial coarse mesh, physical sets of elements are extracted in independent meshes. Once they are isolated, a graph partitioning software is used for each of them. At this step, each subdomain is homogeneous and is an union of coarse elements.

From this point, each subdomain can be treated independently in a parallel way. Since the resulting meshes are too coarse for an acceptable finite element computation, it is necessary to generate finer meshes of subdomains. In a parallel context, the mesh generation is not a trivial task. Indeed, the conformity at the interface between subdomain is not guaranteed on both side of the interfaces of subdomains during a remeshing step. The hierarchical remeshing strategy set up here enables to a priori guarantee this conformity with a parallel remeshing without any communications between subdomains. For this purpose, a hierarchical h-refinement—which consists in dividing existing elements into smaller ones— is performed in each subdomain in parallel to maintain automatically the conformity at interface. By construction, the new created elements are similar to the initial coarse element.

3.2 Subdomains deformation

At this stage, the resulting meshes do not respect the underlying geometry. Indeed, the boundary nodes are located on coarse elements faces and are not located on the entities they are supposed to represent. A mesh deformation step is necessary to ensure the mesh to be on the actual geometry.

Classically, the geometry recovery process is not performed exactly. Indeed, several approximate methods have been proposed. For instance, the new created nodes are repositioned on a very fine mesh describing the geometry using a KD-tree search algorithm [24]. Other methods [25, 26] based on a quadratic surface recovery technique ignore the underlying geometry to interpolate the new created nodes positions. This surface recovery methods are not satisfying because they do not keep sharp edges or they are approximative.

The deformation step is made of the two following main stages:

1. Projection of physical surfaces on the fine skin mesh;

2. Internal volume deformation using previous projections as ”boundary conditions”.

The subdomain’s external surface is projected on the fine geometry with the Möller-Trumbore’s ray-triangle intersection algorithm [27] which consists in finding—for all nodes in the subdomain boundary—the triangle of the fine skin mesh which intersects the line passing through this node in a given direction. The physical edges (those which represent actual geometrical edges) are handled differently. Indeed, the nodes on the physical edge of subdomain are sent at the location of their curvilinear abscissa on the geometric edges of the fine surface mesh.

Then, the mesh deformation is performed with interpolation techniques built on Radial Basis Function (see RBF method in [28]). Each displacement field component u_i (i can be x,y or z, one of the three space directions) at position $\underline{X}$ of the mesh nodes can be derived independently of other directions in terms of the $\underline{X}_{b_j}$ position of the n_b boundary nodes by the expression:

$$u_i(\underline{X}) = \sum_{j=1}^{n_b} \alpha_j \phi\left(\left\|\underline{X} - \underline{X}_{b_j}\right\|\right) + p(\underline{X}), \forall i \in [\![1;3]\!], \tag{8}$$

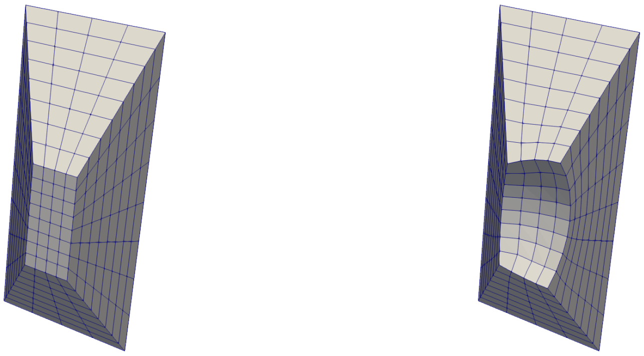
with p a scalar polynomial function defined by its (β_i) coefficients, $(\alpha_j)_{0\leqslant i\leqslant n_b)}$ unknown real coefficients and ϕ a scalar radial basis function. The ϕ functions are $\mathcal{C}^1$ bell shape functions such as $\phi(0) = 1$ and $\lim_{\xi\to\infty} \phi(\xi) = 0$. The p polynomial is a degree-1 polynomial which enables to exactly recover rigid body motions. With the interpolation condition $u_i(\underline{X}_{b_j}) = \underline{u}_{b_j}.\underline{x}_i$, $\forall b_j \in [\![1, n_b]\!]$, we can derive the following linear system (valid for each space directions):

$$\begin{bmatrix} M & P_b \\ P_b^T & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} u_b \\ 0 \end{bmatrix}, \tag{9}$$

where M is a $n_b \times n_b$ symmetric positive definite matrix (provided that ϕ is positive definite) defined by

$$M_{ij} = \phi\left(\left\|\underline{X}_{b_i} - \underline{X}_{b_j}\right\|\right), \tag{10}$$

P_b is a $n_b \times 4$ matrix such as $P_{b_i} = \begin{bmatrix} 1 & X_{b_i} \end{bmatrix}$ and X_b is a $n_b \times 3$ matrix with the coordinates list of prescribed displacements boundary nodes. The radial basis functions ϕ are described by [29]. Once Equation (9) is solved, the displacement field is built following Equation (8) for each space directions and each interior nodes. This transformation enables to change the subdomain's mesh (Figure 3a) into a deformed one with internal nodes which are spread out into the volume (shown on Figure 3b).



(a) A subdomain before volumic interpolation

(b) A subdomain after volumetric interpolation

Figure 3: Internal deformation of a subdomain (section view)

Since the decomposition is fully generated, the FETI strategy can be applied in order to solve the mechanical substructured problem over the newly created partitioned mesh in parallel.

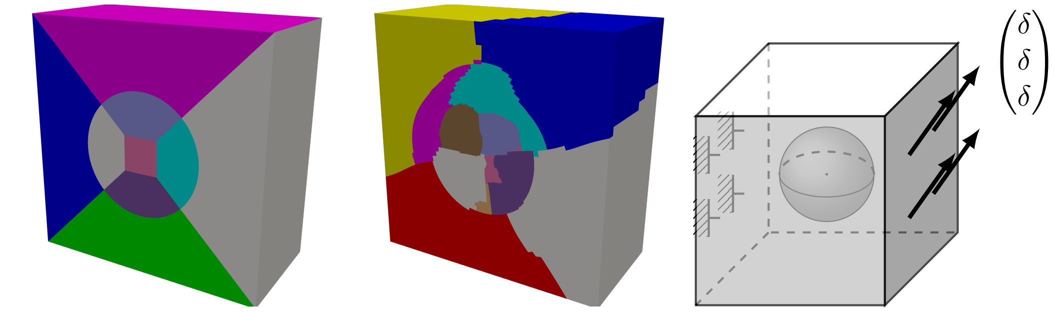
4 Numerical assessments

The potential of the developed method is studied on the academical test case of a cubical structure of $H = 10$ side length (see Figure 4c) with an isotropic linear elastic behaviour. A rigid spherical inclusion made of a stiff material whose elasticity modulus is $E_1 = 200$ GPa and its radius is $H/4$. This inclusion is surrounded by a E_2 modulus soft matrix, such as $E_1/E_2 = 10^5$. This structure is clamped on one side and traction and shear Dirichlet boundary conditions are applied on the opposite face. The initial coarse mesh is made of 13 hexahedral coarse linear elements (six are located in the matrix and seven in the inclusion).

The problem here is defined with a decomposition coming from the developed hierarchical approach presented in section 3 (see Figure 4a) after five hierarchical refinements and with 13 subdomains The size of the problem is about 1.2 million degrees of freedom.

This computation is compared to a classical decomposition of the same mesh obtained by the mean of the Scotch software (see Figure 4b). Keeping the mesh coming from the hierarchical decomposition leads to the same global problem with the same mesh characteristics. It enables to show only the influence of the decomposition on the convergence rate.

The convergence criterion being set as $\|r\|_{\tilde{S}_p} / \|r_0\|_{\tilde{S}_p} \leqslant 10^{-6}$, the evolution of this quantity over iterations is presented on Figure 5. The solver converges in only 34 iterations for



(a) hierarchical decomposition (b) classical decomposition with an automatic graph partitioner (c) traction and shear boundary conditions

Figure 4: Example of a cube with a spherical rigid inclusion and sectional view of the decompositions with 13 subdomains and five hierarchical refinements.

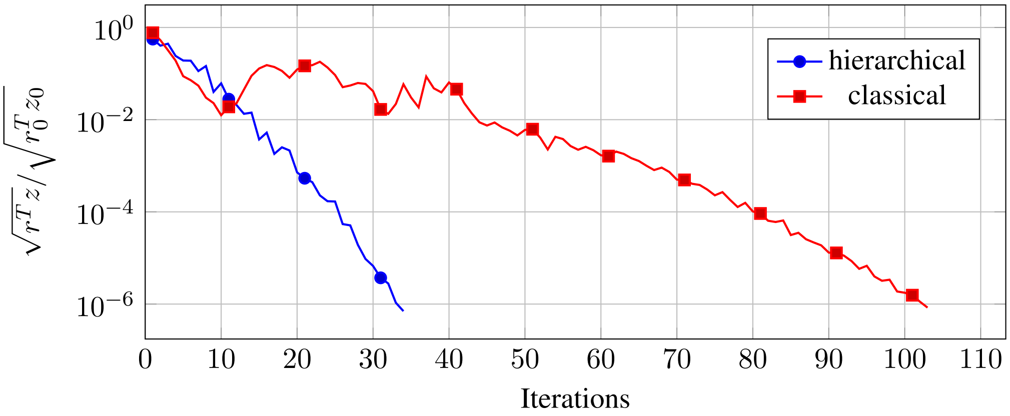


Figure 5: Evolution of the normalized preconditioned residual $\sqrt{r^T z}/\sqrt{r_0^T z_0}$ (notations of Algorithm 1) over FETI iterations (decreasing of initial residual by a 10^6 factor at convergence)

hierarchical decomposition compared to 103 for classical decomposition, or an iteration saving of 67 % on the considered case. Moreover, total CPU time to reach this criterion is 29.2 s for our approach compared with the 96.6 s of the classical approach with similar average CPU time per iteration (respectively 0.86 s and 0.94 s).

Other problems have been tested on the same case by varying from 1 to 5 the number of refinements and the number of subdomains from 2 to 13. Generally speaking, in most cases, the hierarchical decomposition gives iterations savings between 20 % et 60 % compared with classical decomposition computations. Furthermore, hierarchical decomposition leads to a convergence which is less dependent on the mesh size.

5 Conclusion

An original hierarchical substructuring strategy to obtain the most regular interfaces possible and homogeneous subdomains has been introduced and implemented using Python language interfaced with the Z-set finite element software. The entire process is achieved in a parallel framework using `mpi4py` Python module. This strategy consists firstly in generating homogeneous coarse subdomains which are then refined in parallel. Finally, parallel mesh morphing techniques enable to respect the underlying geometry while keeping *a priori* the interface conformity between subdomains. The results seem to be promising even if the method has only been applied on simple geometry.

Thereafter, the properties (particularly the regularity of interfaces) of the obtained decomposition, will be harnessed in order to define optimized solving strategies based on Hybrid of Dual-Primal domain decomposition methods.

References

[1] C. Farhat, F.X. Roux, "A method of finite element tearing and interconnecting and its parallel solution algorithm", *International Journal for Numerical Methods in Engineering*, 32(6): 1205–1227, Oct. 1991, ISSN 0029-5981, 1097-0207, URL `http://doi.wiley.com/10.1002/nme.1620320604`.

[2] J. Mandel, "Balancing domain decomposition", *Communications in Numerical Methods in Engineering*, 9(3): 233–241, Mar. 1993, ISSN 10698299, URL `http://doi.wiley.com/10.1002/cnm.1640090307`.

[3] D. Brands, A. Klawonn, O. Rheinbach, J. Schröder, "Modelling and convergence in arterial wall simulations using a parallel FETI solution strategy", *Computer Methods in Biomechanics and Biomedical Engineering*, 11(5): 569–583, Oct. 2008, ISSN 1025-5842, 1476-8259, URL `http://www.tandfonline.com/doi/abs/10.1080/10255840801949801`.

[4] D.J. Rixen, C. Farhat, "A simple and efficient extension of a class of substructure based preconditioners to heterogeneous structural mechanics problems", *International Journal for Numerical Methods in Engineering*, 44(4): 489–516, Feb. 1999.

[5] P. Gosselet, C. Rey, "Non-overlapping domain decomposition methods in structural mechanics", *Archives of Computational Methods in Engineering*, 13(4): 515–572, Dec. 2006, ISSN 1134-3060, 1886-1784, URL `http://link.springer.com/10.1007/BF02905857`.

[6] A. Loseille, V. Menier, F. Alauzet, "Parallel Generation of Large-size Adapted Meshes", *Procedia Engineering*, 124: 57–69, 2015, ISSN 18777058, URL `https://linkinghub.elsevier.com/retrieve/pii/S1877705815032233`.

[7] D. LaSalle, M.M.A. Patwary, N. Satish, N. Sundaram, P. Dubey, G. Karypis, "Improving graph partitioning for modern graphs and architectures", pages 1–4. ACM Press, 2015, ISBN 978-1-4503-4001-4, URL `http://dl.acm.org/citation.cfm?doid=2833179.2833188`.

[8] F. Pellegrini, J. Roman, "Scotch: A software package for static mapping by dual recursive bipartitioning of process and architecture graphs", in G. Goos, J. Hartmanis, J. van Leeuwen, H. Liddell, A. Colbrook, B. Hertzberger, P. Sloot (Editors), *High-Performance Computing and Networking*, Volume 1067, pages 493–498. Springer Berlin

Heidelberg, Berlin, Heidelberg, 1996, ISBN 978-3-540-61142-4 978-3-540-49955-8, URL http://link.springer.com/10.1007/3-540-61142-8_588.

[9] B. Hendrickson, “Chaco”, in D. Padua (Editor), *Encyclopedia of Parallel Computing*, pages 248–249. Springer US, Boston, MA, 2011, ISBN 978-0-387-09766-4, URL http://dx.doi.org/10.1007/978-0-387-09766-4_310.

[10] J.H. Friedman, J.L. Bentley, R.A. Finkel, “An Algorithm for Finding Best Matches in Logarithmic Expected Time”, *ACM Trans. Math. Softw.*, 3(3): 209–226, Sept. 1977, ISSN 0098-3500, URL http://doi.acm.org/10.1145/355744.355745.

[11] A. Klawonn, O. Widlund, “FETI and Neumann-Neumann iterative substructuring methods: Connections and new results”, *Communications on Pure and Applied Mathematics*, 54(1): 57–90, Jan. 2001, ISSN 0010-3640, 1097-0312, URL http://doi.wiley.com/10.1002/1097-0312%28200101%2954%3A1%3C57%3A%3AAID-CPA3%3E3.0.CO%3B2-D.

[12] C. Bovet, A. Parret-Fréaud, N. Spillane, P. Gosselet, “Adaptive multipreconditioned FETI: Scalability results and robustness assessment”, *Computers & Structures*, 193: 1–20, Dec. 2017, ISSN 00457949, URL https://linkinghub.elsevier.com/retrieve/pii/S004579491730367X.

[13] P. Gosselet, D.J. Rixen, F.X. Roux, N. Spillane, “Simultaneous FETI and block FETI: Robust domain decomposition with multiple search directions”, *International Journal for Numerical Methods in Engineering*, 104(10): 905–927, May 2015.

[14] N. Spillane, D.J. Rixen, “Automatic spectral coarse spaces for robust finite element tearing and interconnecting and balanced domain decomposition algorithms”, *International Journal for Numerical Methods in Engineering*, 95(11): 953–990, 2013, ISSN 1097-0207, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nme.4534.

[15] N. Chrisochoides, “Parallel Mesh Generation”, in A.M. Bruaset, A. Tveito (Editors), *Numerical Solution of Partial Differential Equations on Parallel Computers*, Lecture Notes in Computational Science and Engineering, pages 237–264. Springer Berlin Heidelberg, 2006, ISBN 978-3-540-31619-0.

[16] T. Coupez, H. Digonnet, R. Ducloux, “Parallel meshing and remeshing”, *Applied Mathematical Modelling*, 25(2): 153–175, Dec. 2000, ISSN 0307-904X, URL http://www.sciencedirect.com/science/article/pii/S0307904X00000457.

[17] Y. Ito, A.M. Shih, A.K. Erukala, B.K. Soni, A. Chernikov, N.P. Chrisochoides, K. Nakahashi, “Parallel unstructured mesh generation by an advancing front method”, *Mathematics and Computers in Simulation*, 75(5): 200–209, Sept. 2007, ISSN 0378-4754, URL http://www.sciencedirect.com/science/article/pii/S0378475406003351.

[18] J. Chen, Z. Xiao, Y. Zheng, J. Zou, D. Zhao, Y. Yao, “Scalable generation of large-scale unstructured meshes by a novel domain decomposition approach”, *Advances in Engineering Software*, 121: 131–146, July 2018, ISSN 0965-9978, URL http://www.sciencedirect.com/science/article/pii/S0965997816306378.

[19] X.q. Wang, X.l. Jin, D.z. Kou, J.h. Chen, “A Parallel Approach for the Generation of Unstructured Meshes with Billions of Elements on Distributed-Memory Supercomputers”, *International Journal of Parallel Programming*, 45(3): 680–710, June 2017, ISSN 1573-7640, URL `https://doi.org/10.1007/s10766-016-0452-3`.

[20] J. Besson, R. Foerch, “Large scale object-oriented finite element code design”, *Computer Methods in Applied Mechanics and Engineering*, 142(1-2): 165–187, Mar. 1997, ISSN 00457825, URL `http://linkinghub.elsevier.com/retrieve/pii/S0045782596011243`.

[21] F. Feyel, “Some new technics regarding the parallelisation of ZéBuLoN, an object oriented finite element code for structural mechanics”, *ESAIM: M2AN*, 36(5): 923–935, 2002, URL `https://doi.org/10.1051/m2an:2002040`.

[22] D.J. Rixen, C. Farhat, R. Tezaur, J. Mandel, “Theoretical comparison of the FETI and algebraically partitioned FETI methods, and performance comparisons with a direct sparse solver”, *International Journal for Numerical Methods in Engineering*, 46(4): 501–533, Oct. 1999, ISSN 0029-5981, 1097-0207, URL `http://doi.wiley.com/10.1002/%28SICI%291097-0207%2819991010%2946%3A4%3C501%3A%3AAID-NME685%3E3.0.CO%3B2-7`.

[23] C. Farhat, J. Mandel, “The two-level FETI method for static and dynamic plate problems Part I: An optimal iterative solver for biharmonic systems”, *Computer Methods in Applied Mechanics and Engineering*, 155(1): 129–151, 1998, ISSN 0045-7825, URL `http://www.sciencedirect.com/science/article/pii/S0045782597001461`.

[24] E. Yilmaz, S. Aliabadi, “Surface conformed linear mesh and data subdivision technique for large-scale flow simulation and visualization in Variable Intensity Computational Environment”, *Computers & Fluids*, 80: 388–402, July 2013, ISSN 0045-7930, URL `http://www.sciencedirect.com/science/article/pii/S0045793012000266`.

[25] Y. Ito, K. Nakahashi, “Direct surface triangulation using stereolithography data”, *AIAA Journal*, 40(3): 490–496, 2002.

[26] R. Löhner, “Regridding surface triangulations”, *Journal of Computational Physics*, 126 (1): 1–10, 1996.

[27] T. Möller, B. Trumbore, “Fast, Minimum Storage Ray-Triangle Intersection”, *Journal of Graphics Tools*, 2(1): 21–28, Jan. 1997, ISSN 1086-7651, URL `http://www.tandfonline.com/doi/abs/10.1080/10867651.1997.10487468`.

[28] A. De Boer, M.S. Van Der Schoot, H. Bijl, “Mesh deformation based on radial basis function interpolation”, *Computers & Structures*, 85(11-14): 784–795, June 2007, ISSN 00457949, URL `http://linkinghub.elsevier.com/retrieve/pii/S0045794907000223`.

[29] H. Wendland, *Konstruktion und Untersuchung radialer Basisfunktionen mit kompaktem Träger*, 1996, URL `https://books.google.fr/books?id=4AG5HAAACAAJ`.