A multiscale separated representation to compute the mechanical behavior of composites with periodic microstructure

Sondes Metoui, Etienne Pruliere, Amine Ammar, Frédéric Dau, Ivan Iordanoff

A multiscale separated representation to compute the mechanical behavior of composites with periodic microstructure

S. Metoui[a,b,*], E. Pruliere[a], A. Ammar[b], F. Dau[a], I. Iordanoff[a]

[a] *Arts et Métiers ParisTech, Centre de Bordeaux, I2M-DuMAS, Esplanade des Arts et Métiers, Talence 33405, France*
[b] *Arts et Métiers ParisTech, Centre d'Angers, LAMPA, 2 Boulevard de Ronceray, 49035 Angers Cedex 01, France*

Highlights

- Multiscale problems are generally difficult to solve due to the computational cost.
- The cost of mechanical problem in periodic domains can be reduced impressively using an adequate separated representation.
- A Proper Generalized Decomposition is successfully used to find a solution for multiscale model of composites.
- The gain in computational cost in comparison with the classical finite element increases exponentially when the number of periodic patterns increases.

Abstract

The requirements for advanced numerical computations are very high when studying the multiscale behavior of heterogeneous structures such as composites. For the description of local phenomena taking place on the microscopic scale, the computation must involve a fine discretization of the structure. This condition leads to problems with a high number of degrees of freedom that lead to prohibitive computational costs when using classical numerical methods such as the finite element method (FEM). To overcome these problems, this paper presents a new domain decomposition method based on the proper generalized decomposition (PGD) to predict the behavior of periodic composite structures. Several numerical tests are presented. The PGD results are compared with those obtained using the classical finite element method. A very good agreement is observed.

Keywords: Model reduction; Multiscale simulations; Proper Generalized Decomposition; Composite structures

1. Introduction

One of the main challenges in mechanics and engineering is to account for physical phenomena that occur at different scales. A coupling between scales is often observed, generating a real need for multiscale models in many

* Corresponding author at: Arts et Métiers ParisTech, Centre de Bordeaux, I2M-DuMAS, Esplanade des Arts et Métiers, Talence 33405, France.
E-mail addresses: sondes.metoui@ensam.eu (S. Metoui), Etienne.Pruliere@ensam.eu (E. Pruliere), amine.ammar@ensam.eu (A. Ammar), frederic.dau@ensam.eu (F. Dau), ivan.iordanoff@ensam.eu (I. Iordanoff).

http://dx.doi.org/10.1016/j.matcom.2017.07.010

applications. In composite materials, for example, there are at least two or three characteristic scales: the fiber scale, the ply scale and the laminate scale. A major difficulty related to multiscale modeling is the need of multiscale solvers that require a lot of computational resources. Therefore, there is a real need for computational methods able to reduce the cost of such simulations.

In this paper, we focus on problems related to structural mechanics of composite materials, that results in elliptic partial differential equations. The multiscale solvers available in literature can be divided into two main classes:

- The first class considers a set of microscopic volumes, generally representative volume element (RVE). For example, one RVE can be defined in each Gaussian point of the macroscopic finite element mesh. The size of the RVE is often much smaller than the size of the macroscopic finite elements i.e. the fine-scale mesh covers only a very small part of the macroscopic domain. Some well known methods are the Heterogeneous Multiscale Methods (HMM) [10] or the FE^2 methods [12] in engineering literature. This kind of methods is generally based on an extension of the homogenization theory for non linear materials.
- The second class take into account the microscale (solution) in each point of the macroscopic domain. The fine-scale mesh covers the whole macroscopic computational domain. In this case each fine-scale degree of freedom (DOF) has to be treated at least one time in the algorithm. Therefore, these methods are generally more expansive than the method belonging to the first class. In this class, we find in particular the multigrid methods and the Domain Decomposition Methods (DDM) [11,13,17–19,24].

The approaches belonging to the first classes are efficient in a computational point of view. However, they assume a well separated definition of scales that is not always relevant, in particular when considering the characteristic length of composite materials. Furthermore, this kind of methods requires to introduce some additional hypothesis to make the link between the microscale and the macroscale (generally boundary conditions applied to the microscopic domain [15]). For periodic microstructures, the natural choice is to assume periodic boundary conditions on the RVE but this lead to a loss of precision near the boundaries of the domain.

The approach presented in this paper belongs mainly to the second class in the sense that the whole problem is described. However, the objective is to reduce the number of degrees of freedom as in the methods belonging to the first class, by the use of some additional hypothesis.

The main hypothesis is that only materials with periodic structures are considered with an obvious application in composite materials. The periodicity of the microstructure enables to separate two scales: the scale of the periodic cell and a macroscopic scale. The idea is to use this hypothesis to build the full solution assuming a separated form in the context of the Proper Generalized Decomposition (PGD). With the PGD solver, the number of degrees of freedom to treat can theoretically be reduced by several orders of magnitude.

The PGD philosophy was originally proposed by Ladeveze with space–time problems in the context of the LATIN (LArge Time INcrement) method, which was called radial decomposition [16]. Ammar et al. have generalized this method for multi-dimensional problems in the context of the kinetic theory description of complex fluids [3,4]. This method enables the reduction in size of multidimensional and parametric problems [7,21,23]. The idea of the PGD is to build an approximation of the solution based on a separated representation. Assume that we are looking for a field $u : \Omega \to \mathbb{R}$ that depends on some coordinates x_j for $j = 1, \ldots, D$ where the domain $\Omega \subset \mathbb{R}^D$ is defined by: $\Omega = \Omega_{x_1} \times \Omega_{x_2} \times \cdots \times \Omega_{x_D}$ with $\Omega_{x_i} \subset \mathbb{R}$ for $i = 1 \ldots D$. A separated representation of u can be defined as a sum of N products of some functions $F_i^j(x_j) : \Omega_{x_j} \to \mathbb{R}$ such as:

$$u(x_1, \ldots, x_D) \approx \sum_{i=1}^{N} F_i^1(x_1) \times \cdots \times F_i^D(x_D). \tag{1}$$

All these functions F_i^j are a priori unknown. This representation is injected into the weak formulation of the partial differential equations depending on the considered problem. This leads to non-linear equations that are solved iteratively. Separated representations have been used for a wide range of problems. In particular, the PGD has been applied to compute full 3D solutions using an in-plane/out-of-plane separated representation in composite laminates [6] and for the modeling of laminated and sandwich composite plates [26,27]. In our previous works, we have employed this method to simulate delamination of composite laminates [20].

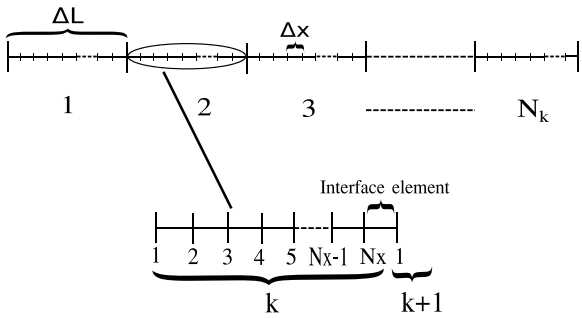


Fig. 1. 1D mesh and decomposition in elementary parts.

2. A multiscale separated representation

2.1. Domain decomposition method for periodic 1D domains

2.1.1. Separated description of the 1D problem

For the sake of simplicity, the method will be described firstly for a simple problem defined in a 1D domain $\Omega = [0, L]$. The 2D and 3D cases will be addressed in the next section.

Then, an 1D problem with a periodic geometry and periodic material properties is considered in this section. The weak formulation of the static equilibrium equation for a beam in traction/compression with an elastic material is:

$$\int_0^L AE \frac{\mathrm{d}U^\star(X)}{\mathrm{d}X} \frac{\mathrm{d}U(X)}{\mathrm{d}X} \,\mathrm{d}X = \left[AE\, U^\star(X) \frac{\mathrm{d}U(X)}{\mathrm{d}X} \right]_0^L \quad \forall U^\star \in \mathcal{V} \tag{2}$$

E is the elastic modulus, A is the area of the beam, X is the coordinate along the beam axis, $U \in H^1(\Omega)$ is the longitudinal displacement, and $U^\star \in \mathcal{V} = \{f \in H^1(\Omega)\}$ is a test function. L denotes the length of the beam. We also define: $[f(X)]_0^L = f(L) - f(0)$ for any function f.

The right hand term is related to the boundary conditions on the left ($X = 0$) and on the right ($X = L$) of the domain. To enforce the boundary conditions, a penalty method will be used. This method will be described in Section 2.1.4. For now, only the left hand term of the equation will be considered.

In the finite element method, this weak form is approximated using some shape functions over each element of a mesh. The proposed strategy takes advantage of the periodicity of the geometry and material properties to build a periodic mesh. The periodicity assumption, in this context is that the domain may be sliced into identical elementary parts with: $E(X + \Delta L) = E(X)$ and $A(X + \Delta L) = A(X)$. ΔL is the length of each part. If loads are added to this problem, the periodicity is not required.

Then, the mesh can be built as the sum of identical mesh with common nodes on the edge as depicted in Fig. 1 for the 1D mesh. Each occurrence of the periodic cell is associated to an integer denoted k as shown in Fig. 1. k is an integer such as $k \in \{1, \ldots, N_k\}$, and $N_k = \frac{L}{\Delta L}$ denotes the number of cells. N_x denotes the number of nodes in the periodic mesh.

The coordinate X defining the horizontal position in the beam is written using k:

$$X = (k - 1)\, \Delta L + x \tag{3}$$

$x \in [0, \Delta L]$ is the position in each part.

The continuous form of the displacement field found using the separated approximation of the solution is given by:

$$U(X) = U((k - 1)\, \Delta L + x) = \sum_{i=1}^{n} F_i(x) G_i(k) \tag{4}$$

where $F_i(x) : [0, \Delta L] \to \mathbb{R}$ and $G_i(k) : \{1, \ldots, N_k\} \to \mathbb{R}$ are some unknown functions. In the following, the notation $U(k, x) = U((k - 1)\, \Delta L + x)$ will be used.

Using k and x instead of X leads to a double definition of the position at the edge of the elementary parts: $U(k, \Delta L) = U(k+1, 0)$. This is problematic from a numerical point of view because it leads to a multiplication of degrees of freedom on these points. There are two possibilities to treat the edge of the elementary part:

1. Solving the problem using more degrees of freedom than necessary and using boundary conditions to enforce the continuity: $U(k, \Delta L) = U(k+1, 0)$.
2. Deleting some nodes when discretizing the domain in order to suppress the non-necessary degrees of freedom.

The first method requires to enforce some continuity conditions at the interface of each cell. With the separated representation of the solution Eq. (4), the continuity conditions give:

$$\sum_{i=1}^{n} F_i(\Delta L) G_i(k) = \sum_{i=1}^{n} F_i(0) G_i(k+1). \tag{5}$$

As the separated representation is generally built term by term, the continuity must be satisfied for each term of the sum. This leads to:

$$F_i(\Delta L) G_i(k) = F_i(0) G_i(k+1) \quad \forall (i,k) \in \{1, \dots, N_k\} \times \{1, \dots, n\}. \tag{6}$$

Thus, if $F_i(0) \neq 0$ the function G_i is completely defined from the values of $F_i(0)$, $F_i(\Delta L)$ and $G_i(1)$:

$$G_i(k) = \left(\frac{F_i(\Delta L)}{F_i(0)} \right)^{k-1} G_i(1).$$

This constraint on G_i decreases dramatically the convergence rate of the PGD. To circumvent this problem, a possibility is to compute many terms of the separated approximation at the same time instead of a classical term by term algorithm.

In the following, only the second strategy is described.

Let us consider a finite element discretization of the domain with equally distributed nodes as shown in Fig. 1. This distribution of nodes is only for sake of clarity without loss of generality. The size of an element is denoted Δx. Then, the weak formulation Eq. (2) can be rewritten:

$$\begin{aligned} &\sum_{k=1}^{N_k} \left(\int_{(k-1)\Delta L}^{k\Delta L - \Delta x} AE \frac{\mathrm{d}U^\star(X)}{\mathrm{d}X} \frac{\mathrm{d}U(X)}{\mathrm{d}X} \,\mathrm{d}X \right) + \sum_{k=1}^{N_k - 1} \left(\int_{k\Delta L - \Delta x}^{k\Delta L} AE \frac{\mathrm{d}U^\star(X)}{\mathrm{d}X} \frac{\mathrm{d}U(X)}{\mathrm{d}X} \,\mathrm{d}X \right) \\ &\quad + \int_{N_k \Delta L - \Delta x}^{N_k \Delta L} AE \frac{\mathrm{d}U^\star(X)}{\mathrm{d}X} \frac{\mathrm{d}U(X)}{\mathrm{d}X} \,\mathrm{d}X = \left[AE\, U^\star(X) \frac{\mathrm{d}U(X)}{\mathrm{d}X} \right]_0^L \quad \forall U^\star \in \mathcal{V}. \end{aligned} \tag{7}$$

The first sum $\sum_{k=1}^{N_k} \left(\int_{(k-1)\Delta L}^{k\Delta L - \Delta x} AE \frac{\mathrm{d}U^\star(X)}{\mathrm{d}X} \frac{\mathrm{d}U(X)}{\mathrm{d}X} \,\mathrm{d}X \right)$ contains integrals defined on the elements that are inside the elementary parts. The second sum $\sum_{k=1}^{N_k - 1} \left(\int_{k\Delta L - \Delta x}^{k\Delta L} AE \frac{\mathrm{d}U^\star(X)}{\mathrm{d}X} \frac{\mathrm{d}U(X)}{\mathrm{d}X} \,\mathrm{d}X \right)$ contains integrals defined over the interface elements. The element located on the right boundary needs a special treatment that will be detailed in Section 2.1.4.

With a change of variable between X and x it comes:

$$\begin{aligned} &\overbrace{\sum_{k=1}^{N_k} \left(\int_{0}^{\Delta L - \Delta x} AE \frac{\mathrm{d}U^\star(k,x)}{\mathrm{d}x} \frac{\mathrm{d}U(k,x)}{\mathrm{d}x} \,\mathrm{d}x \right)}^{I_1} + \overbrace{\sum_{k=1}^{N_k - 1} \left(\int_{\Delta L - \Delta x}^{\Delta L} AE \frac{\mathrm{d}U^\star(k,x)}{\mathrm{d}x} \frac{\mathrm{d}U(k,x)}{\mathrm{d}x} \,\mathrm{d}x \right)}^{I_2} \\ &\quad + \int_{N_k \Delta L - \Delta x}^{N_k \Delta L} AE \frac{\mathrm{d}U^\star(X)}{\mathrm{d}X} \frac{\mathrm{d}U(X)}{\mathrm{d}X} \,\mathrm{d}X = \left[AE\, U^\star(X) \frac{\mathrm{d}U(X)}{\mathrm{d}X} \right]_0^L \quad \forall U^\star \in \mathcal{V}. \end{aligned} \tag{8}$$

2.1.2. First iteration

The approximation defined by Eq. (4) is built term by term. For now, we focus on the first iteration. The displacement is then approximated by:

$$U(X) = F_1(x) G_1(k) = R(x) S(k). \tag{9}$$

To avoid the redundant use of subscripts, we denote $R = F_1$ and $S = G_1$. The determination of R and S involves a non-linear problem that is solved using the classical alternate direction strategy [3]. At the beginning, R is computed assuming a random value for S. Then S is computed knowing R. And again R is computed knowing S and so on until R and S have converged.

Then, two different problems must be treated:

1. Computing R knowing S
2. Computing S knowing R.

The test function is $U^\star(k, x) = R^\star(x)S(k)$ for the first problem and $U^\star(k, x) = R(x)S^\star(k)$ for the second problem where $R^\star(x) \in H^1([0, \Delta L]) = \mathcal{V}_R$ and $S^\star(k) : \{1, \ldots, N_k\} \to \mathbb{R}$ are some associated test functions.

The first problem is considered in the following. The first integral of Eq. (8) can be simply rewritten using the separated approximation:

$$\begin{aligned} I_1 &= \sum_{k=1}^{N_k} \int_0^{\Delta L - \Delta x} AE \frac{\mathrm{d}R^\star(x)}{\mathrm{d}x} \frac{\mathrm{d}R(x)}{\mathrm{d}x} S^2(k)\,\mathrm{d}x \\ &= \left(\int_0^{\Delta L - \Delta x} AE \frac{\mathrm{d}R^\star(x)}{\mathrm{d}x} \frac{\mathrm{d}R(x)}{\mathrm{d}x}\,\mathrm{d}x \right) \sum_{k=1}^{N_k} S^2(k). \end{aligned} \tag{10}$$

After a finite element discretization it remains:

$$I_1 = \left(\mathbf{R^{\star T}} \mathbf{K} \mathbf{R}\right) \times \left(\mathbf{S^T} \mathbf{I_{N_k}} \mathbf{S}\right) \tag{11}$$

$\mathbf{R}$ is the column vector containing the nodal values of R:

$$\mathbf{R} = \begin{bmatrix} R_1 & R_2 & \cdots & R_{N_x} \end{bmatrix}^T \tag{12}$$

$\mathbf{S}$ is the column vector containing the values of S:

$$\mathbf{S} = \begin{bmatrix} S_1 & S_2 & \cdots & S_{N_k} \end{bmatrix}^T \tag{13}$$

$\mathbf{R}^*$ is a vector containing the nodal values of the test function R^*.

$\mathbf{K}$ is the stiffness matrix related to the periodic cell and $\mathbf{I}_{N_k}$ is the $N_k \times N_k$ identity matrix.

The second integral in Eq. (8) requires a little more development since the degree of freedom corresponding to $U_k(\Delta L) = U_{k+1}(0)$ is defined only on the part $k + 1$. This integral is defined on the interface elements between two parts. For 1D linear elements the shape function vector over the interface element is: $\mathbf{\Phi}(x) = \begin{bmatrix} \phi_{N_x}(x) & \phi_1(x) \end{bmatrix}$(see Fig. 1) where $\phi_i(x)$ denote the shape function associated to the node i. For 3 nodes quadratic elements the shape function vector becomes: $\mathbf{\Phi}(x) = \begin{bmatrix} \phi_{N_x-1}(x) & \phi_{N_x}(x) & \phi_1(x) \end{bmatrix}$.

The matrix of DOF is for linear elements:

$$\mathbf{Q}_k = \begin{bmatrix} R_{N_x} S_k \\ R_1 S_{k+1} \end{bmatrix} \tag{14}$$

or for quadratic elements:

$$\mathbf{Q}_k = \begin{bmatrix} R_{N_x-1} S_k \\ R_{N_x} S_k \\ R_1 S_{k+1} \end{bmatrix}. \tag{15}$$

Using this finite element approximation, I_2 can be written as:

$$I_2 = \sum_{k=1}^{N_k-1} \int_{\Delta L - \Delta x}^{\Delta L} AE \frac{\mathrm{d}U^\star(k, x)}{\mathrm{d}x} \frac{\mathrm{d}U(k, x)}{\mathrm{d}x}\,\mathrm{d}x = \sum_{k=1}^{N_k-1} \mathbf{Q}^{\star T} \int_{\Delta L - \Delta x}^{\Delta L} AE \frac{\mathrm{d}\mathbf{\Phi}(x)}{\mathrm{d}x}^T \frac{\mathrm{d}\mathbf{\Phi}(x)}{\mathrm{d}x}\,\mathrm{d}x \mathbf{Q}. \tag{16}$$

Defining $\alpha_{ij} = \int_{\Delta L - \Delta x}^{\Delta L} AE \frac{\mathrm{d}\phi_i}{\mathrm{d}x} \frac{\mathrm{d}\phi_j}{\mathrm{d}x}\,\mathrm{d}x$ and developing the previous equation in the case of linear elements, I_2 becomes:

$$I_2 = R^\star_{N_x} \alpha_{N_x N_x} R_{N_x} \sum_{k=1}^{N_k-1} S_k^2 + R^\star_{N_x} \alpha_{1 N_x} R_1 \sum_{k=1}^{N_k-1} S_k S_{k+1} + R^\star_1 \alpha_{1 N_x} R_{N_x} \sum_{k=1}^{N_k-1} S_{k+1} S_k + R^\star_1 \alpha_{11} R_1 \sum_{k=1}^{N_k-1} S_{k+1}^2. \tag{17}$$

In general, four operators may be defined as follows:

$$I_2 = \left(\mathbf{R}^{\star\mathbf{T}}\mathbf{M}_{N_x,N_x}\mathbf{R}\right) \times \left(\mathbf{S}^{\mathbf{T}}\mathbf{D}_{k,k}\mathbf{S}\right) + \left(\mathbf{R}^{\star\mathbf{T}}\mathbf{M}_{N_x1}\mathbf{R}\right) \times \left(\mathbf{S}^{\mathbf{T}}\mathbf{D}_{k,k+1}\mathbf{S}\right) + \left(\mathbf{R}^{\star\mathbf{T}}\mathbf{M}_{1,N_x}\mathbf{R}\right) \times \left(\mathbf{S}^{\mathbf{T}}\mathbf{D}_{k+1,k}\mathbf{S}\right) + \left(\mathbf{R}^{\star\mathbf{T}}\mathbf{M}_{1,1}\mathbf{R}\right) \times \left(\mathbf{S}^{\mathbf{T}}\mathbf{D}_{k+1,k+1}\mathbf{S}\right) \tag{18}$$

where $\mathbf{D}_{k,k}$, $\mathbf{D}_{k+1,k+1}$, $\mathbf{D}_{k+1,k}$ and $\mathbf{D}_{k,k+1}$ are the following $N_k \times N_k$ square matrices:

$$\mathbf{D}_{k,k} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{D}_{k+1,k+1} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \tag{19}$$

$$\mathbf{D}_{k,k+1} = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 0 & 1 \\ 0 & \cdots & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{D}_{k+1,k} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix} \tag{20}$$

and $\mathbf{M}_{N_x,N_x}$, $\mathbf{M}_{1,1}$, $\mathbf{M}_{N_x,1}$ and $\mathbf{M}_{1,N_x}$ are some $N_x \times N_x$ square matrices coming from the development of Eq. (16). In the case of linear elements, these matrices are identified from Eq. (17):

$$\mathbf{M}_{N_x,N_x} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & \alpha_{N_xN_x} \end{bmatrix} \quad \mathbf{M}_{1,1} = \begin{bmatrix} \alpha_{11} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 \end{bmatrix} \tag{21}$$

$$\mathbf{M}_{N_x,1} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \iddots & \vdots \\ 0 & 0 & \cdots & 0 \\ \alpha_{1N_x} & 0 & \cdots & 0 \end{bmatrix} \quad \mathbf{M}_{1,N_x} = \begin{bmatrix} 0 & \cdots & 0 & \alpha_{1N_x} \\ 0 & \cdots & 0 & 0 \\ \vdots & \iddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \end{bmatrix}. \tag{22}$$

In the separated formulation some operators are non symmetric. However it is interesting to notice that the global problem remains symmetric because: $\mathbf{D}_{k+1,k} = \mathbf{D}_{k,k+1}^T$ and $\mathbf{M}_{1,N_x} = \mathbf{M}_{N_x,1}^T$. The stiffness matrix $\mathbf{K}$ defined in Eq. (11) is also symmetric.

The weak formulation Eq. (8) without boundary gives eventually after eliminating $\mathbf{R}^*$:

$$(\mathbf{KR}) \times \left(\mathbf{S}^{\mathbf{T}}\mathbf{I}_{N_k}\mathbf{S}\right) + \left(\mathbf{M}_{N_x,N_x}\mathbf{R}\right) \times \left(\mathbf{S}^{\mathbf{T}}\mathbf{D}_{k,k}\mathbf{S}\right) + \left(\mathbf{M}_{N_x1}\mathbf{R}\right) \times \left(\mathbf{S}^{\mathbf{T}}\mathbf{D}_{k,k+1}\mathbf{S}\right) + \left(\mathbf{M}_{1,N_x}\mathbf{R}\right) \times \left(\mathbf{S}^{\mathbf{T}}\mathbf{D}_{k+1,k}\mathbf{S}\right) + \left(\mathbf{M}_{1,1}\mathbf{R}\right) \times \left(\mathbf{S}^{\mathbf{T}}\mathbf{D}_{k+1,k+1}\mathbf{S}\right) = \mathbf{B_1}(\mathbf{R},\mathbf{S}). \tag{23}$$

And the system for the second problem, that is to compute S knowing R writes:

$$\left(\mathbf{R}^{\mathbf{T}}\mathbf{KR}\right) \times \left(\mathbf{I}_{N_k}\mathbf{S}\right) + \left(\mathbf{R}^{\mathbf{T}}\mathbf{M}_{N_x,N_x}\mathbf{R}\right) \times \left(\mathbf{D}_{k,k}\mathbf{S}\right) + \left(g\mathbf{R}^{\mathbf{T}}\mathbf{M}_{N_x1}\mathbf{R}\right) \times \left(\mathbf{D}_{k,k+1}\mathbf{S}\right) + \left(\mathbf{R}^{\mathbf{T}}\mathbf{M}_{1,N_x}\mathbf{R}\right) \times \left(\mathbf{D}_{k+1,k}\mathbf{S}\right) + \left(\mathbf{R}^{\mathbf{T}}\mathbf{M}_{1,1}\mathbf{R}\right) \times \left(\mathbf{D}_{k+1,k+1}\mathbf{S}\right) = \mathbf{B_2}(\mathbf{R},\mathbf{S}) \tag{24}$$

$\mathbf{B_1}(\mathbf{R},\mathbf{S})$ and $\mathbf{B_2}(\mathbf{R},\mathbf{S})$ are vectors associated to boundary conditions that are defined in Section 2.1.4.

The linear systems Eqs. (23) and (24) are some reduced version of a more global system (i.e, the full FEM system). The nodal values of the global solution $U(X) = R(x)S(k)$ can be built a posteriori using a simple tensor product $\mathbf{U} = \mathbf{R} \otimes \mathbf{S}$. These two systems are then well-posed if the equivalent global problem is also well-posed and if the global solution $\mathbf{U}$ computed from $\mathbf{S}$ and $\mathbf{R}$ satisfies the boundary conditions.

Algorithm 1 General PGD algorithm used to make the link between the elementary cell and the global mesh

1: n = 0
2: initialize F_{n+1} and G_{n+1} to random values
3: $F_{n+1}^{old} = F_{n+1}$; $G_{n+1}^{old} = G_{n+1}$
4: Compute F_{n+1} knowing G_{n+1} (update the function related to the elementary cell)
5: Compute G_{n+1} knowing F_{n+1} (update the function related to the global mesh)
6: if $max(\left\| F_{n+1} - F_{n+1}^{old} \right\|_2, \left\| G_{n+1} - G_{n+1}^{old} \right\|_2) > \epsilon_1 max(\|F_{n+1}\|_2, \|G_{n+1}\|_2)$ then go to 3
7: n = n+1
8: $U(x,k) = \sum_{i=1}^{n} F_i(x)G_i(k)$ (no need to compute U explicitly)
9: If the normalized Euclidean norm of the residual of the system (Eq. 26) $> \epsilon_2$ then go to 2

2.1.3. Other iterations

For other iterations, F_i and G_i are assumed known for $i = 1, \ldots, n$. Now, we are looking for $R = F_{n+1}$ and $S = G_{n+1}$ such as:

$$U(x,k) = \sum_{i=1}^{n} F_i(x)G_i(k) + R(x)S(k). \tag{25}$$

The test function is the same as in the first iteration:

- $U^\star(k,x) = R^\star(x)\, S(k)$ if the unknown is R.
- $U^\star(k,x) = R(x)\, S^\star(k)$ if the unknown is S.

We focus on the case where the unknown is $\mathbf{R}$. Introducing this expression into the weak form, and using a finite elements discretization as described for the first iteration we get:

$$\begin{aligned}
&(\mathbf{KR}) \times \left(\mathbf{S^T I}_m \mathbf{S}\right) + \left(\mathbf{M}_{N_x,N_x}\mathbf{R}\right) \times \left(\mathbf{S^T D}_{k,k}\mathbf{S}\right) + \left(\mathbf{M}_{N_x 1}\mathbf{R}\right) \times \left(\mathbf{S^T D}_{k,k+1}\mathbf{S}\right) \\
&\quad + \left(\mathbf{M}_{1,N_x}\mathbf{R}\right) \times \left(\mathbf{S^T D}_{k+1,k}\mathbf{S}\right) + \left(\mathbf{M}_{1,1}\mathbf{R}\right) \times \left(\mathbf{S^T D}_{k+1,k+1}\mathbf{S}\right) = \mathbf{B_1}(\mathbf{R},\mathbf{S}) \\
&\quad - \sum_{i=1}^{n} \left[(\mathbf{KF}_i) \times \left(\mathbf{S^T I}_m \mathbf{G}_i\right) + \left(\mathbf{M}_{N_x,N_x}\mathbf{F}_i\right) \times \left(\mathbf{S^T D}_{k,k}\mathbf{G}_i\right) + \left(\mathbf{M}_{N_x 1}\mathbf{F}_i\right) \times \left(\mathbf{S^T D}_{k,k+1}\mathbf{G}_i\right)\right. \\
&\quad \left. + \left(\mathbf{M}_{1,N_x}\mathbf{F}_i\right) \times \left(\mathbf{S^T D}_{k+1,k}\mathbf{G}_i\right) + \left(\mathbf{M}_{1,1}\mathbf{F}_i\right) \times \left(\mathbf{S^T D}_{k+1,k+1}\mathbf{G}_i\right)\right]
\end{aligned} \tag{26}$$

where $\mathbf{F}_i$ and $\mathbf{G}_i$ contain the nodal values of $F_i(x)$ and the values of $G_i(k)$.

A very similar system can be easily written when the unknown is S.

The global PGD algorithm is summed up in algorithm 1. In this algorithm ϵ_1 and ϵ_2 are error tolerances related to the convergence criteria. The convergence of this algorithm has been discussed in [2].

2.1.4. Boundary conditions

For now, the method has been described without accounting for boundary conditions. As the method is based on a finite element discretization, stress conditions can naturally be introduced by adding a vector of generalized force to the system to be solved. The only difference with the classical finite element method is that it must be written on a separated form. The direct application of Dirichlet conditions on the separated representation is generally not possible. For instance, if the beam is clamped in $x = 0$ the boundary condition is:

$$U(k = 1, x = 0) = 0. \tag{27}$$

As the solution is built iteratively, the boundary conditions must be satisfied at each iteration, i.e.:

$$\sum_{i=1}^{n} F_i(0)G_i(1) = 0. \tag{28}$$

It is easy to prove that this leads to:

$$F_i(0) = 0 \text{ or } G_i(1) = 0 \quad \forall i \in \{1, \ldots, n\}. \tag{29}$$

If we enforce $G_i(1) = 0\ \forall i \in \{1, \ldots, n\}$, the solution is zero over the first cell and we can get the true solution only in the very restrictive case where there is actually no displacement over this cell. In other hand, if we enforce $F_i(0) = 0\ \forall i \in \{1, \ldots, n\}$, that leads to:

$$U(k, 0) = U((k-1)\,\Delta L) = 0 \quad \forall k \in \{1, \ldots, N_k\}. \tag{30}$$

In that case, the displacement vanishes on the left of each cell. In this case again, we cannot get the real displacement or only in some very restrictive cases.

Then, the best solution to apply Dirichlet conditions is to use a penalty method. It consists in adding some new operators that describe the boundary conditions. For instance, to enforce a unitary displacement on the left, the operators related to boundary conditions are:

$$\mathbf{M_{bc}} = \beta \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 \end{bmatrix} \qquad \mathbf{D_{bc}} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 \end{bmatrix}$$

$$\mathbf{B^x_{bc}} = \beta \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \qquad \mathbf{B^k_{bc}} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

where $\beta \in \mathbb{R}^+$ is the purely numerical penalty parameter. $\mathbf{M_{bc}}$ is a $N_x \times N_x$ matrix, $\mathbf{D_{bc}}$ is a $N_k \times N_k$ matrix, $\mathbf{B^x_{bc}}$ is a $N_x \times 1$ matrix and $\mathbf{B^k_{bc}}$ is a $N_k \times 1$ matrix.

The boundary conditions are specified by defining $\mathbf{B_1}$ and $\mathbf{B_2}$ in Eqs. (23) and (24):

$$\mathbf{B_1}(\mathbf{R}, \mathbf{S}) = \mathbf{B^x_{bc}} \times \left(\mathbf{S^T B_{bc}}^k\right) - (\mathbf{M_{bc}R}) \times \left(\mathbf{S^T D_{bc} S}\right)$$
$$\mathbf{B_2}(\mathbf{R}, \mathbf{S}) = \left(\mathbf{R^T B_{bc}}^x\right) \times \mathbf{B^k_{bc}} - \left(\mathbf{R^T M_{bc} R}\right) \times (\mathbf{D_{bc} S})\,.$$

The system given in Eq. (23) becomes with this boundary condition:

$$\begin{aligned} &(\mathbf{KR}) \times \left(\mathbf{S^T I}_m \mathbf{S}\right) + \left(\mathbf{M}_{N_x,N_x}\mathbf{R}\right) \times \left(\mathbf{S^T D}_{k,k}\mathbf{S}\right) + \left(\mathbf{M}_{N_x 1}\mathbf{R}\right) \times \left(\mathbf{S^T D}_{k,k+1}\mathbf{S}\right) \\ &\quad + \left(\mathbf{M}_{1,N_x}\mathbf{R}\right) \times \left(\mathbf{S^T D}_{k+1,k}\mathbf{S}\right) + \left(\mathbf{M}_{1,1}\mathbf{R}\right) \times \left(\mathbf{S^T D}_{k+1,k+1}\mathbf{S}\right) + (\mathbf{M_{bc}R}) \times \left(\mathbf{S^T D_{bc}S}\right) \\ &= \mathbf{B^x_{bc}} \times \left(\mathbf{S^T B^k_{bc}}\right). \end{aligned} \tag{31}$$

This define a linear system whose unknowns are the component of $\mathbf{R}$. This system may be solved by the conjugate gradient method or any other classical method.

Another problem has to be discussed relating to boundary conditions. On the right side, there is obviously no interface element. In comparison to other elementary parts, the right part has one node less on the right. This may be a problem. A simple solution to overcome this difficulty is to add a "virtual" elementary part on the right that is only used for one node. The other nodes are only virtual and the element inside this virtual part have to be ignored. In operators $\mathbf{D}_{k,k}$, $\mathbf{D}_{k,k+1}$, $\mathbf{D}_{k+1,k}$ and $\mathbf{D}_{k+1,k+1}$, the row corresponding to the virtual part (the row N_k) must be filled with 0 in order to prevent the effect of virtual element. There is no undesirable numerical error related to the presence of virtual element even if there is no unique solution (the solution inside the virtual element can be anything). As detailed previously, the enrichment of the separated representation is performed using an alternate direction strategy that is a type of fixed point method. The solution will converge to one among the possible solutions according to the initial values of the vectors $\mathbf{R}$ and $\mathbf{S}$ (random values in general) [2].

2.2. Domain decomposition method for periodic 2D and 3D domains

2.2.1. Mechanical model

In this section, a 2D or 3D static problem is considered.

$\underline{U}$ denotes the displacement vector expressed in the canonical basis:

$$\underline{U}(\underline{X}) = \begin{bmatrix} U_x(\underline{X}) \\ U_y(\underline{X}) \\ U_z(\underline{X}) \end{bmatrix} \tag{32}$$

where $\underline{X} = (X, Y, Z)^T$ is the coordinate vector. $\boldsymbol{\varepsilon}$ denotes the strain tensor using Voigt notation:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ 2\varepsilon_{yz} \\ 2\varepsilon_{xz} \\ 2\varepsilon_{xy} \end{bmatrix}. \tag{33}$$

The stress tensor using the Voigt notation is written as:

$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{zz} \\ \sigma_{yz} \\ \sigma_{xz} \\ \sigma_{xy} \end{bmatrix} = \mathbf{H} \begin{bmatrix} \varepsilon_{xx} \\ \varepsilon_{yy} \\ \varepsilon_{zz} \\ 2\varepsilon_{yz} \\ 2\varepsilon_{xz} \\ 2\varepsilon_{xy} \end{bmatrix} = \mathbf{H}\boldsymbol{\varepsilon} \tag{34}$$

$\mathbf{H}$ is the matrix describing the constitutive equation. In small displacements, $\boldsymbol{\varepsilon}$ is the symmetric gradient of the displacement. With these notations, the weak formulation of the equilibrium equation without dynamic effect and volume forces is expressed as:

$$\int_{\Omega} \boldsymbol{\varepsilon}(\underline{U}^\star)^T (\mathbf{H}\varepsilon(\underline{U}))\, \mathrm{d}\Omega = \int_{\Gamma} \underline{U}^\star.(\sigma.\underline{n})\, \mathrm{d}\Gamma \qquad \forall \underline{U}^\star \in \left(H^1(\Omega)\right)^2 : \tag{35}$$

Ω is the domain taken by the structure, Γ is the boundary of the domain, $U^\star \in H^1(\Omega)$ is a test function and $\underline{n}$ is the unit vector normal to Γ directed toward the exterior of Ω. The right part of this weak formulation is only used for boundary conditions. It is not included in the following for the sake of readability.

2.2.2. *Separated description of 2D problems*

As for the 1D case, the domain Ω is decomposed into N_k periodic elementary cells Ω_k. Ω_i denotes the set of elements inside the elementary cell and Ω_e is the set of interface elements. The integral over Ω in Eq. (35) can be decomposed as a sum of integrals over $\Omega_k = \Omega_i \cup \Omega_e(k)$. The weak form gives then for all $\underline{U}^\star \in \left(H^1(\Omega)\right)^2$:

$$\underbrace{\sum_{k=1}^{N_k} \int_{\Omega_i} \boldsymbol{\varepsilon}(\underline{U}^\star)^T (\mathbf{H}\boldsymbol{\varepsilon}(\underline{U}))\, \mathrm{d}\Omega}_{I_1} + \underbrace{\sum_{k=1}^{N_k} \int_{\Omega_e(k)} \boldsymbol{\varepsilon}(\underline{U}^\star)^T (\mathbf{H}\boldsymbol{\varepsilon}(\underline{U}))\, \mathrm{d}\Omega}_{I_2} = \int_{\Gamma} \underline{U}^\star.(\sigma.\underline{n})\, \mathrm{d}\Gamma. \tag{36}$$

To avoid the double definition of degrees of freedom at the interface between cells, some nodes need to be deleted at the interface. We choose to delete the nodes on the right and up faces for the 2D case as shown in Fig. 2.

The local coordinates in the elementary cells are noted $\underline{x}$. approximation associated to a 2D or a 3D problem is then:

$$\underline{U}(\underline{x}) = \sum_{i=1}^{n} \underline{F}_i(\underline{x}) G_i(k). \tag{37}$$

Here, G_i is a scalar function and $\underline{F}_i$ is a vector function:

$$\underline{F}_i(\underline{x}) = \begin{pmatrix} F_i^x(\underline{x}) \\ F_i^y(\underline{x}) \\ F_i^z(\underline{x}) \end{pmatrix}. \tag{38}$$

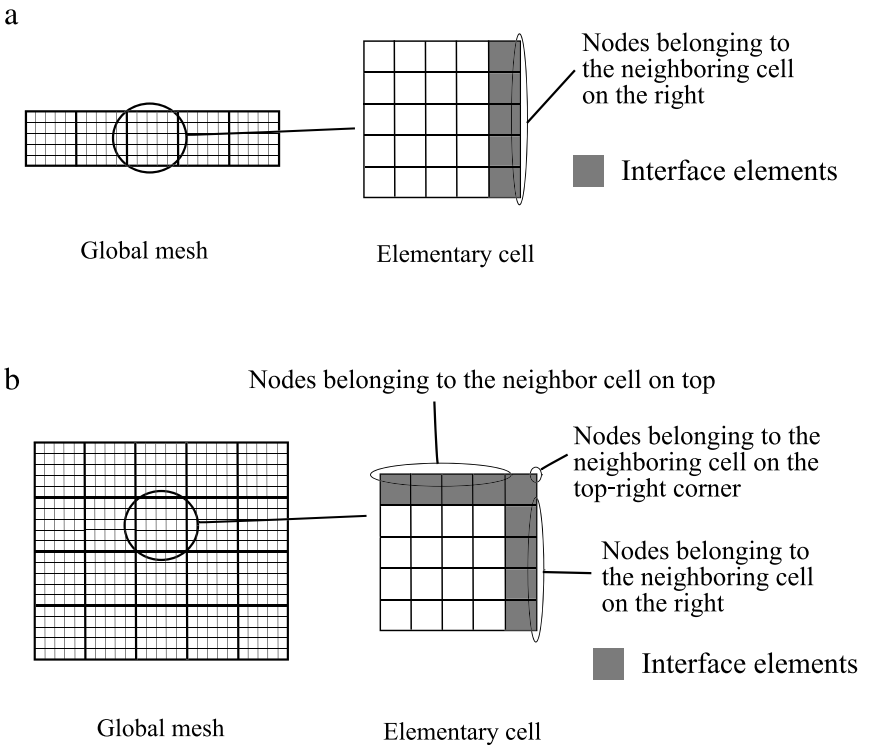


Fig. 2. Examples of 2D meshes and decomposition in elementary parts.

2.2.3. First iteration and operators assembly

The integral I_1 is treated as for the 1D case. For the first iteration with $\underline{U}(\underline{x}) = \underline{R}(\underline{x})S(k)$, I_1 reads:

$$I_1 = \left(\int_{\Omega_i} \boldsymbol{\varepsilon}(\underline{R}^\star)^T (\mathbf{H}\boldsymbol{\varepsilon}(\underline{R}))\, \mathrm{d}\Omega \right) \sum_{k=1}^{N_k} S(k)^2$$
$$\approx \left(\mathbf{R}^{\star\mathbf{T}}\mathbf{K}\mathbf{R}\right) \times \left(\mathbf{S}^{\mathbf{T}}\mathbf{I}_{\mathbf{N_k}}\mathbf{S}\right)$$

$\underline{R}^\star \in \left(H^1(\Omega_i)\right)^2$ is a test function, $\mathbf{K}$ is the stiffness matrix related to Ω_i and $\mathbf{I}_{N_k}$ is the $N_k \times N_k$ identity matrix. $\mathbf{R}$ is the column vector containing the nodal values of R^x, R^y and R^z and $\mathbf{S}$ is the column vector containing the values of S.

The second integral I_2 needs an assembly of new matrices that are representative of interactions between neighboring cells. The element stiffness matrices of interface elements must be assembled in some global operators written from a tensor product of matrix operators either associated to the nodal values of Ω_i or associated to the list periodic cells $\{1, \ldots, N_k\}$.

For instance, in the first 2D problem depicted in Fig. 2, I_2 can be written as:

$$I_2 = \sum_{m=1}^{4} \left(\mathbf{R}^{\star\mathbf{T}}\mathbf{M}_m^r\mathbf{R}\right) \times \left(\mathbf{S}^{\mathbf{T}}\mathbf{D}_m^r\mathbf{S}\right)$$

$\mathbf{D}_m^r$ are the matrices defined by:

$$\left(\mathbf{D}_1^r\right)_{i,j} = \begin{cases} 1 \text{ if } i = j \text{ and the cell } i \text{ has a right neighboring cell} \\ 0 \text{ otherwise} \end{cases}$$

$$\left(\mathbf{D}_2^r\right)_{i,j} = \begin{cases} 1 \text{ if } i = j \text{ and the cell } i \text{ has a left neighboring cell} \\ 0 \text{ otherwise} \end{cases}$$

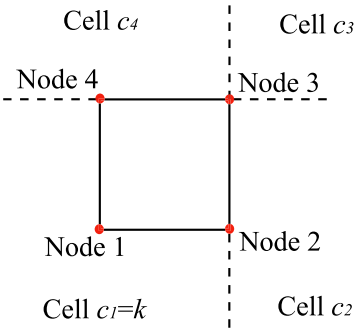


Fig. 3. Corner element: nodes and neighboring cells definition.

$$\left(\mathbf{D}_3^r\right)_{i,j} = \begin{cases} 1 \text{ if the cell } j \text{ is at the right of the cell } i \\ 0 \text{ otherwise} \end{cases}$$

$$\left(\mathbf{D}_4^r\right)_{i,j} = \begin{cases} 1 \text{ if the cell } i \text{ is at the right of the cell } j \\ 0 \text{ otherwise} \end{cases}$$

and the matrices $\mathbf{M}_m^r$ are matrices that result from the assembly. It is interesting to notice that $\mathbf{D}_2^r = \left(\mathbf{D}_1^r\right)^T$ and $\mathbf{M}_2^r = \left(\mathbf{M}_1^r\right)^T$.

The second problem depicted in Fig. 2 is more complicated. There are now 3 types of interface elements:

- elements between the cell and its right neighbor (subscript r),
- elements between the cell and its top neighbor (subscript t),
- an element in the top-right corner that is at the interface of 4 different cells (subscript c).

The assembly results in the following sum:

$$I_2 = \underbrace{\sum_{m=1}^{4} \left(\mathbf{R}^{\star\mathbf{T}}\mathbf{M}_m^r\mathbf{R}\right) \times \left(\mathbf{S}^{\mathbf{T}}\mathbf{D}_m^r\mathbf{S}\right)}_{\text{right interface elements}} + \underbrace{\sum_{m=1}^{4} \left(\mathbf{R}^{\star\mathbf{T}}\mathbf{M}_m^t\mathbf{R}\right) \times \left(\mathbf{S}^{\mathbf{T}}\mathbf{D}_m^t\mathbf{S}\right)}_{\text{top interface elements}} + \underbrace{\sum_{m=1}^{16} \left(\mathbf{R}^{\star\mathbf{T}}\mathbf{M}_m^c\mathbf{R}\right) \times \left(\mathbf{S}^{\mathbf{T}}\mathbf{D}_m^c\mathbf{S}\right)}_{\text{corner interface element}}.$$

The right interface elements bring the same operators as in the first case. The operators related to the top interface elements are also very similar.

The element on the corner is assembled with 16 operators because it is at the interface of 4 different cells. The assembly of the corner element is detailed as follows.

We assume for illustration that the corner element is the 4 nodes quadrilateral element as shown in Fig. 3.

The nodes belonging to the corner element of a cell $k \in \{1, \ldots, N_k\}$ are numbered from 1 to 4. Each one of these nodes $i \in 1, 2, 3, 4$ belong to a different cell $c_i(k)$.

Fig. 3 shows that $c_1 = k$, c_2 is the cell on the right of k, c_3 is on the top-right of k and c_4 is on the top of k. Using these notations, the matrix containing the degrees of freedom of the corner element is:

$$\mathbf{Q}_k^e = \begin{bmatrix} R_1^x S_{c_1} \\ R_2^x S_{c_2} \\ R_3^x S_{c_3} \\ R_4^x S_{c_4} \\ R_1^y S_{c_1} \\ R_2^y S_{c_2} \\ R_3^y S_{c_3} \\ R_4^y S_{c_4} \end{bmatrix}. \tag{39}$$

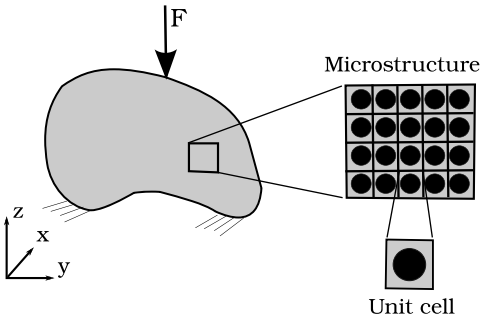


Fig. 4. Example of composite structure with periodic geometry.

Table 1
Material properties and geometric parameters of the virtual cell.

Material properties				
E_f (GPa)	E_m (GPa)	ν_f	ν_m	V_f (%)
390	4.5	0.35	0.40	34

The element stiffness matrix of the corner element is noted $\mathbf{K}^c$. The weak formulation restricted to the corner elements is: $\sum_{k\in\mathcal{C}} \left(\mathbf{Q}_k^{e\star T}\mathbf{K}^c\mathbf{Q}_k^e\right)$, where $\mathcal{C}$ denotes the set of cells having an interface element on the top-right corner.

This sum is then decomposed as follows:

$$\sum_{k\in\mathcal{C}} \left(\mathbf{Q}_k^{e\star T}\mathbf{K}^c\mathbf{Q}_k^e\right) = \sum_{i=1}^{4}\sum_{j=1}^{4}\left(\begin{bmatrix} R_i^{x\star} & R_i^{y\star}\end{bmatrix}\begin{bmatrix}\mathbf{K}^c_{i,i} & \mathbf{K}^c_{i,j+4}\\ \mathbf{K}^c_{i+4,j} & \mathbf{K}^c_{i+4,j+4}\end{bmatrix}\begin{bmatrix}R_j^x\\ R_j^y\end{bmatrix} \times \sum_{k\in\mathcal{C}}\left(S_{c_i(k)}S_{c_j(k)}\right)\right). \tag{40}$$

There are then 16 separated operators. It remains only to assemble these element operators into global operators in order to get $\mathbf{M}_m^c$ and $\mathbf{D}_m^c$ (with $m = 1, 2, \ldots, 16$).

Remark. This method requires a lot of separated operators. There are 25 operators for the 2D case treated here and 129 operators for the full 3D case. Though this is a high number, all the matrices are very sparse (some matrices have a few non-zero elements).

3. Results

In this section, the method described previously will be considered for solving some test cases in order to evaluate its performance in terms of results precision and computational cost.

3.1. Numerical results of 2d problem

Consider a composite structure composed of unidirectional fibers. We assume that this structure has a periodical geometry which can be described by an heterogeneous unit cell, as shown in Fig. 4.

Two modeling scales are then considered: the macroscopic scale which define the structure scale and the microscopic scale represented by the unit cell (a single fiber surrounded by resin).

The material properties and the geometric parameters are summarized in Table 1 [5]. The subscript f is used for the fiber properties and the subscript m for the matrix properties. V_f denotes the fiber volume fraction.

In this subsection, the numerical simulations performed using the PGD are compared with results of classical FEM implementation using the same set of parameters. The aim is to validate the PGD approach. The two methods are based on the same mesh of the unit cell and the error related to the finite element discretization is exactly the same in both cases. It is important to notice that the comparison between the PGD and the FEM will only show the error due to the separated approximation. In the best case, the PGD will give the same result as the finite element.

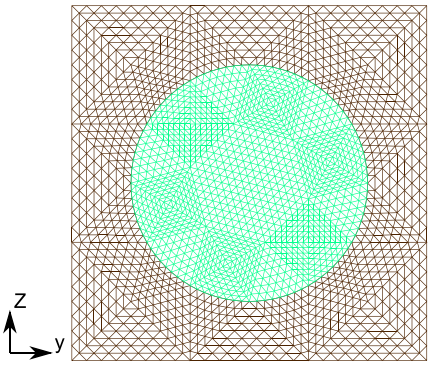


Fig. 5. The representative unit cell at the microscale.

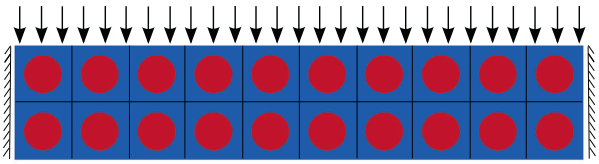


Fig. 6. Domain decomposed into subdomains and boundary conditions.

Therefore, the finite element method serves as a reference solution in order to assess the performance of the PGD approximation.

The problem was divided into 20 substructures, each substructure corresponding to one cell whose length is $\Delta L = 9\,\mu\text{m}$. The loads and boundary conditions applied to the structure are shown in Fig. 6. The linear force is set to -1 N mm^{-1} and the left and right edges are clamped. The problem is solved in 2D using a plane strain assumption with a unity thickness.

Three meshes are considered to study the mesh convergence. A coarse one composed of 149 nodes, a fine one with 2225 node and an intermediate one with 569 nodes are represented in Fig. 5. The three meshes have been generated by successive refinements. 2D triangular elements with 3-nodes are used for all meshes. The axis **x** is set to be along the fibers and is then perpendicular to the plane.

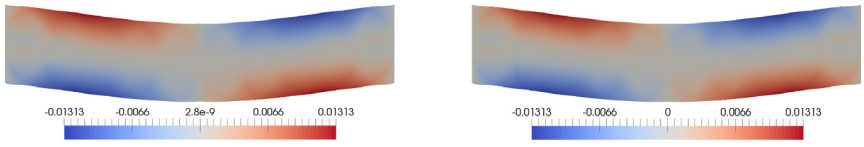
The finite element mesh was constructed by assembling the meshes of 20 unit cells. For the fine mesh, a total of 43809 nodes involving 87618 degrees of freedom (two DOF per node) are used.

The displacements of the structure are depicted for the intermediate mesh in Fig. 7. The PGD approach and the FEM give quite similar results for the displacements.

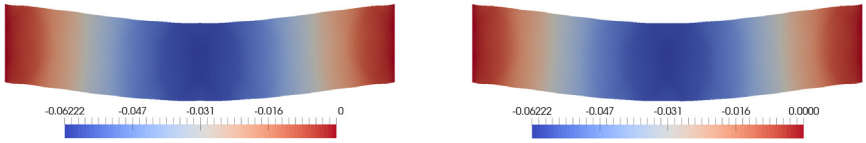
The stresses distributions are also plotted in Fig. 8. The results of the PGD are also in excellent agreement with the 2D FEM solution. The PGD approach is able to capture the stress distribution in the microstructure with almost the same precision as the FEM.

Fig. 9 gives the distribution of the stresses through the thickness (coordinate z) at different positions: the center for σ_{yy} and the left edge for σ_{yz}. These positions are chosen to be the most critical for each stress component. The difference between the two methods is also plotted. Referring to these figures, the PGD solution performs very well with respect to the FEM solution. However, we can see that the error increases slightly near the boundaries.

Table 2 summarizes the value obtained for some stress and displacement components at different positions and for the three meshes. The coordinates are given using the center of the geometry for origin and using the dimension of the unit cell denoted ΔL. We can see that the intermediate mesh gives some results very close to the ones of fine mesh. The coarse mesh is clearly less precise, especially for the shear stress σ_{yz}. The normal stress σ_{zz} on the top should logically converge to the external linear force -1 MPa. This is almost the case for the fine mesh. A Timoshenko beam model using a homogenized transverse Young modulus gives an approximate deflection of 0.0657 mm. This value is quite consistent with the deflection obtained at the center, especially for the two finer meshes.

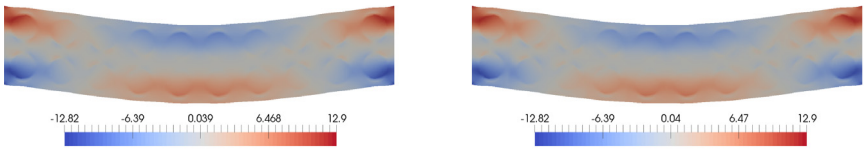


(a) u_y.

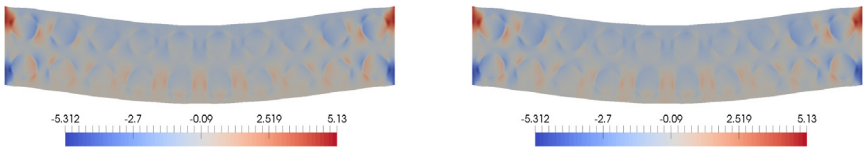


(b) u_z.

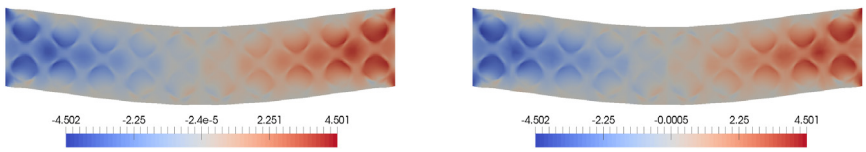
Fig. 7. Displacement fields (mm) : FEM solution (left)—PGD solution (right).



(a) σ_{yy}.



(b) σ_{zz}.



(c) σ_{yz}.

Fig. 8. Stresses distributions (MPa) : FEM solution (left)—PGD solution (right).

The values obtained by the PGD for around 250 iterations are in good agreement with the one obtained by the FEM. The 250 iterations lead to only $n \approx 30$ in the separated approximation because the solution is compressed using a SVD or a simple PGD algorithm with the identity operator. However, we can point out some significant differences. The displacement given by the PGD seems to be more precise than the stress. The value of σ_{zz} on the center/top is -0.748 MPa for the fine mesh, far from the -1 expected. This shows a locally non converged solution. The tolerance in the convergence criterion is difficult to choose correctly. This remains a difficulty in the PGD solver and the development of efficient error estimators is probably an important challenge for some future works.

For now, we have used the classical PGD solver which consist in trying to reach the convergence of the separated approximation Eq. (37) with respect to a defined criterion. Another possible approach is to set the number of terms n in Eq. (37) to a chosen value and to search the solution of the weak equation truncated to this n terms. For that purpose, instead of the Algorithm 1 we can compute the n functions $\underline{F}_i\ (\underline{x})$ (or alternately $G_i\ (k)$) for $i = 1 \ldots n$ at the same time. The interested reader can refer to [22] for more details.

Results of the FEM and PGD at different positions with 3 meshes: Stress components (MPa) and Displacement components (mm).

	σ_{yy}	σ_{yy}	σ_{yz}	σ_{zz}	u_y	u_z
(x, y)	$(0, \frac{\Delta L}{2})$	$(0, -\frac{\Delta L}{2})$	$(-4\Delta L, 0)$	$(0, \Delta L)$	$(-4\Delta L, \Delta L)$	$(0, 0)$
Classical finite element method						
Coarse mesh	−5.237	5.001	−3.449	−1.073	0.0101	−0.0608
Interm. mesh	−5.456	5.238	−3.762	−1.033	0.0104	−0.0621
Fine mesh	−5.526	5.311	−3.721	−1.012	0.0105	−0.0625
PGD—incremental strategy						
Coarse mesh	−5.225	5.026	−3.52	−1.097	0.0102	−0.0609
Interm. mesh	−5.373	5.07	−3.532	−0.989	0.0103	−0.062
Fine mesh	−5.407	5.114	−3.648	−0.748	0.0101	−0.0624

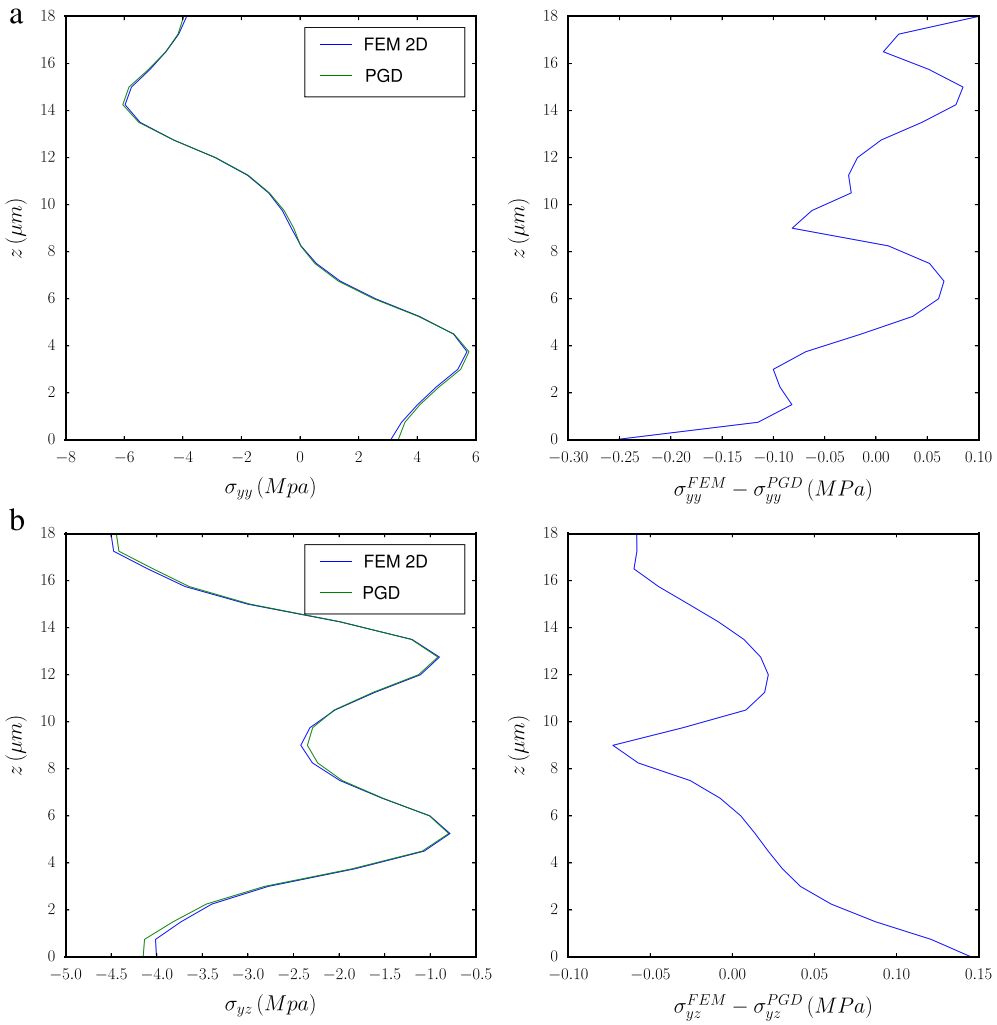


Fig. 9. Distribution of the stresses along the thickness: (a) σ_{yy} at the center, (b) σ_{yz} along the left edge.

Table 3 gives some results given by the PGD using this strategy for $n = 6$, $n = 8$ and $n = 10$ with the intermediate mesh. Using a small number of terms permits to reduce significantly the computational cost of the simulation but gives

Table 3
Results of the PGD with predefined number of terms at different positions with the intermediate mesh: Stress components (MPa) and Displacement components (mm).

PGD—fixed number of terms in the separated decomposition						
	σ_{yy}	σ_{yy}	σ_{yz}	σ_{zz}	u_y	u_z
(x, y)	$(0, \frac{\Delta L}{2})$	$(0, -\frac{\Delta L}{2})$	$(-4\Delta L, 0)$	$(0, \Delta L)$	$(-4\Delta L, \Delta L)$	$(0, 0)$
$n = 6$	−4.83	4.741	−4.092	−0.537	0.0092	−0.0567
$n = 8$	−5.084	4.869	−3.81	−0.991	0.00963	−0.0611
$n = 10$	−5.406	5.188	−3.736	−1.0176	0.0105	−0.0619

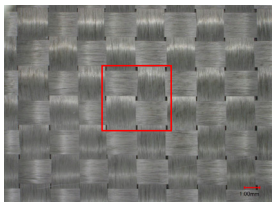


Fig. 10. Woven carbon fiber.

only an approximation of the solution. For instance, with 6 terms, we get mainly an order of magnitude of the solution. With 8 terms, the solution is much closer to the FEM solution and the computational cost remains smaller than the one of the classical PGD solver (71 s instead of 108 s for the intermediate mesh with 20 subdomains). With $n = 10$, the solution is almost converged and we are even closer to the FEM solution than with the classical PGD algorithm. The advantage of this strategy is that the computational time is weakly correlated to the number of subdomains (expected for very high number of subdomains).

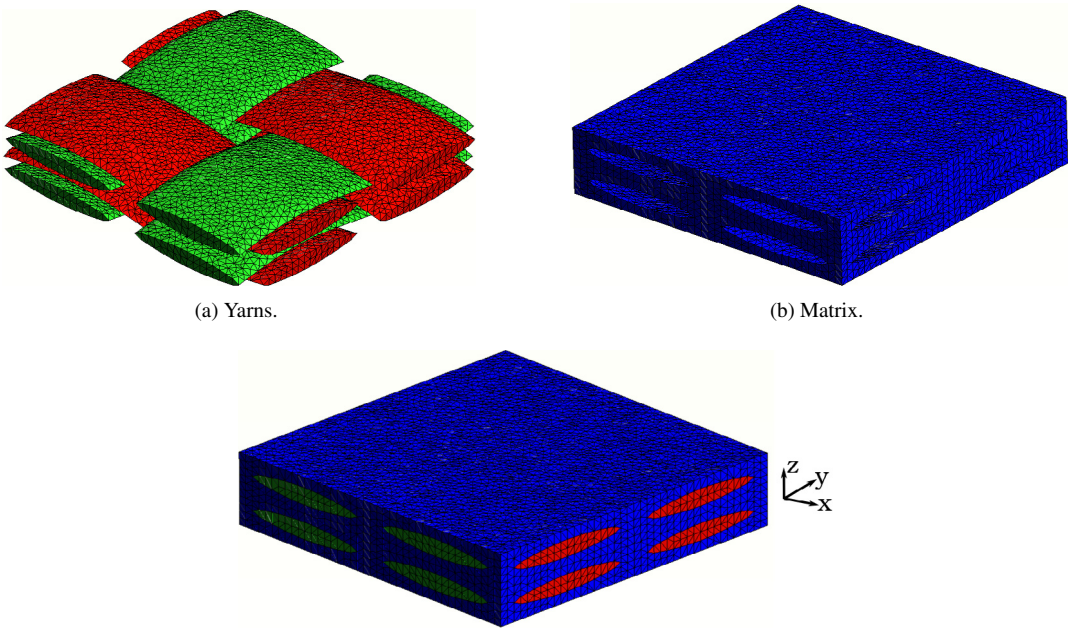
On a simple laptop (CPU Intel core i7-4510U at 2.00 GHz, 2 cores, RAM 8 Go DDR3 at 1600MHz) the computational cost required to get the solution with 10 terms is about 115 s for 20 subdomains, 127 s for 980 subdomains, 165 s for 2000 subdomains and 1945 s for 18000 subdomains. With 6 terms, the computational cost required to get the solution is about 34 s for 20 subdomains and 637 s for 18000 subdomains. The computational cost is correlated to the number of degrees of freedom in the separated representation, i.e, $n \times (N_{micro} + N_{macro})$, where N_{micro} is the number of DOF in the microscopic mesh and N_{macro} is the number of DOF in the macroscopic mesh. The number of terms n required to catch the physics is dependent on the complexity of the problem and not necessarily on the number of subdomains.

The computational cost is given only for sake of illustration because it depends strongly on the implementation. The codes related to the PGD and classical FEM are both home made codes developed in Python using Numpy [9] and Scipy [14] for sparse matrices. However, it is important to note that the resolution of the Finite Element is only based on Scipy sparse matrices that include some efficient compiled methods to solve sparse linear systems (umfpack [8], superlu [28]). In the case of the PGD, there remains many loops in python (Cpython) that are non optimized and a high factor could be saved with a good implementation using an efficient compiler.

3.2. Numerical results of 3d problem

For sake of illustration, a multiscale 3D model is used to simulate the mechanical behavior of woven composite materials. This model is difficult to treat using a classical finite element solver with 3D solid elements. It consists in a two-ply composite made from woven carbon fibers (taffeta) and epoxy resin.

Woven reinforcements are generally periodic media: in fact, they consist of a repeating pattern or a unit cell to reconstruct the complete composite fabric, as shown in Fig. 10. Due to the geometrical complexity of woven composites architecture, the 3D geometric model was built using TexGen software [25] as shown in Fig. 11(a). TexGen is a very powerful tool to generate geometric models for composite textiles. The TexGen model is transferred to ABAQUS [1]. As could be seen in figure Fig. 11(b), the matrix volume is then created by subtracting the yarns volume. The next step is to construct the finite element mesh of the unit cell. In order to do the simulations, the mesh of



(a) Yarns.

(b) Matrix.

(c) Unit cell mesh.

Fig. 11. Periodic tetrahedral mesh of unit cell model.

Table 4
Material property.

Matrix		Effective properties of yarn					
E (MPa)	ν	E_1 (GPa)	E_2 (GPa)	ν_{12}/ν_{13}	ν_{23}	G_{12}/G_{13} (GPa)	G_{23} (GPa)
3500	0.3	234.345	45.743	0.327	0.375	20.056	16.628

the unit cell must be periodic. Fig. 11(c) shown the 3D 4 nodes periodic tetrahedral mesh of unit cell model. The unit cell dimension is $2 \times 2 \times 0.42$ (mm). PGD simulations were carried out using linear elastic behaviors, the matrix and the yarns were assumed to be respectively isotropic and orthotropic. Material properties of the matrix and the effective properties of the yarns are given in Table 4. The subscript 1 corresponds to the longitudinal direction of the yarn, and the subscripts 2 and 3 correspond to the transverse directions. The properties of the yarn have been computed from simple rules of mixtures assuming a fiber volume fraction of 90% ($E_f = 250$ GPa).

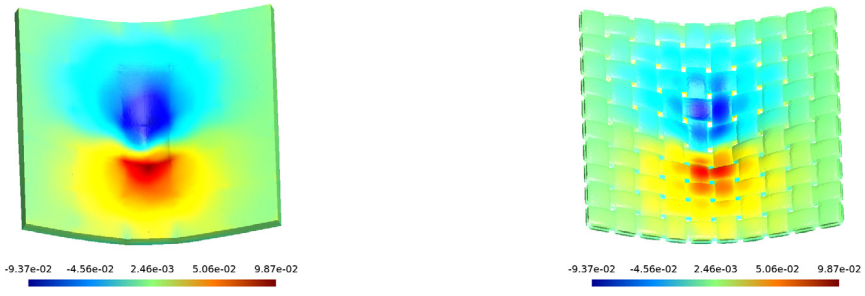
The unit cell mesh is composed of 28934 nodes. The PGD problem was divided into 25 substructures which give a total of 706390 nodes involving 2119170 degrees of freedom. The displacement along z of the composite plate is zero on both ends (for $x = 0$ and $x = x_{max}$). A constant pressure of 367 MPa is applied on the top face of the unit cell positioned at the center of the plate (surface force). A visualization of the deformed configuration is presented in Fig. 12 and puts emphasis on the displacements. The maximum displacement value in the z direction logically appears at the middle of the plate where the load is applied.

The stress values for the plate are analyzed and presented in Fig. 13. These stress fields are complex because of the heterogeneity of the model and of the orthotropic behavior of the yarn. With the proposed reduced strategy, these stress fields are accessible with a relatively low computational cost (30 min on a laptop).

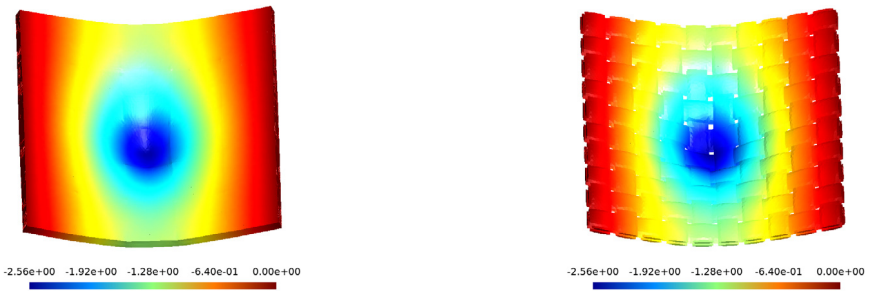
4. Conclusion

This paper presents a new separated representation method for solving efficiently multiscale problems. In this work, the Proper Generalized Decomposition is revisited using a separation of scales in periodic geometries. This separated

(a) u_x.



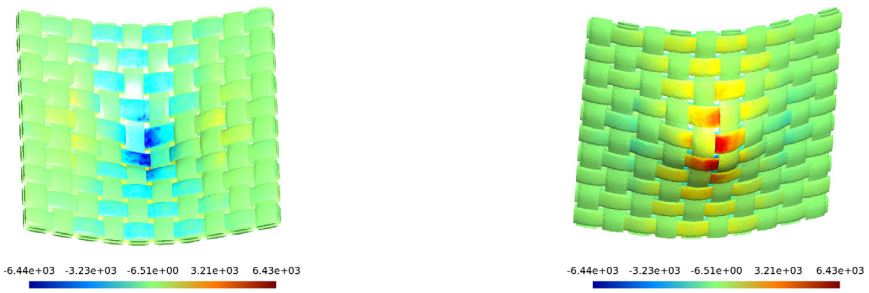
(b) u_y.



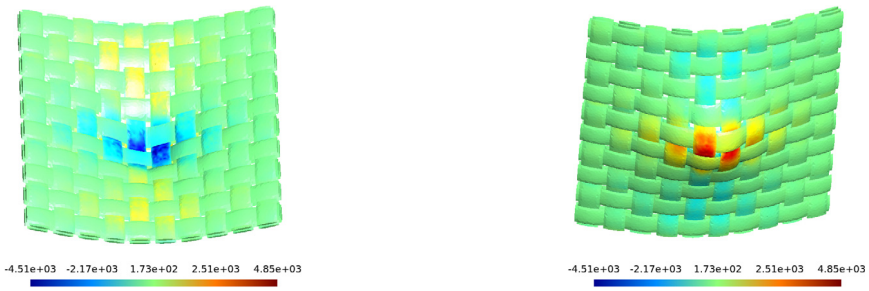
(c) u_z.

Fig. 12. Displacement fields and deformation of a plain weave based composite: Full model is shown (left)—only yarn type materials are shown (right).

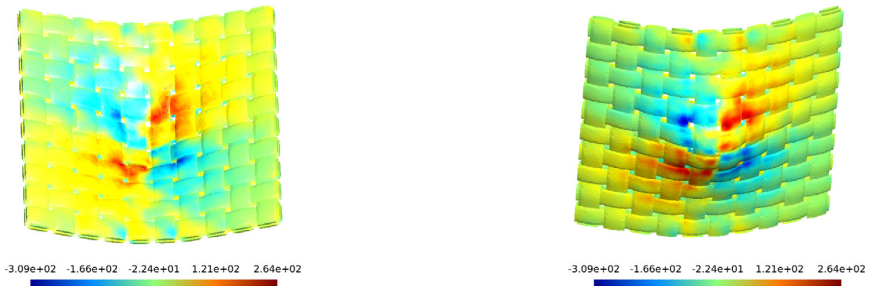
representation involves both the space coordinates of the microscopic scale as well as the space coordinates of the macroscopic scale. For coupling each subdomain, an efficient algorithm is proposed and validated: the proposed technique has a satisfying precision when the convergence tolerance and the number of terms in the separated representation are chosen adequately. The estimation of convergence error remains a hard point in the algorithm that may be addressed in further studies. When the number of periodic cells is high (several thousands at least), especially in 3D problems, the method can be very impressive in terms of computational cost. Another perspective that has not to be investigated in this article is that the method is naturally capable of treating non-linear behavior since a non linear solver is already used to reach the convergence. However, in the case of strong non-linearity, we can expect that the number of terms required to catch the physics will increase and therefore the computational cost will also increase. In this context, the amelioration of the algorithm that build the Proper Generalized Decomposition for problems with a high number of operators is a key point that also needs further developments.



(a) σ_{xx}.



(b) σ_{yy}.



(c) σ_{xy}.

Fig. 13. Stress distributions of a plain weave based composite: only yarn type materials are shown. Upper surface (left) — lower surface (right).

References

[1] Abaqus cae, dassault systemes, 2014, http://www.3ds.com/products-services/simulia/products/abaqus/ (Accessed: 17.05.16).

[2] A. Ammar, F. Chinesta, A. Falcó, On the convergence of a greedy rank-one update algorithm for a class of linear systems, Arch. Comput. Methods Eng. 17 (4) (2010) 473–486.

[3] A. Ammar, B. Mokdad, F. Chinesta, R. Keunings, A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids, J. Non-Newton Fluid Mech. 139 (3) (2006) 153–176.

[4] A. Ammar, B. Mokdad, F. Chinesta, R. Keunings, A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modelling of complex fluids: Part II: Transient simulation using space-time separated representations, J. Non-Newton Fluid Mech. 144 (2–3) (2007) 98–121.

[5] J.-M. Berthelot, Matériaux Composites : Comportement Mécanique et Analyse des Structures, Masson, 1996.

[6] B. Bognet, F. Bordeu, F. Chinesta, A. Leygue, A. Poitou, Advanced simulation of models defined in plate geometries: 3D solutions with 2D computational complexity, Comput. Methods Appl. Mech. Engrg. 201–204 (2012) 1–12.

[7] F. Chinesta, A. Ammar, E. Cueto, Recent advances and new challenges in the use of the proper generalized decomposition for solving multidimensional models, Arch. Comput. Methods Eng. 17 (4) (2010) 327–350.

[8] T.A. Davis, Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method, ACM Trans. Math. Software (TOMS) 30 (2) (2004) 196–199.

[9] S.V. der Walt, S.C. Colbert, G. Varoquaux, The numpy array: A structure for efficient numerical computation, Comput. Sci. Eng. 13 (2011) 22–30.

[10] W. E, B. Engquist, The heterogeneous multiscale methods, Commun. Math. Sci. 1 (2003) 87–132.
[11] C. Farhat, F.X. Roux, A method of finite element tearing and interconnecting and its parallel solution algorithm, Internat. J. Numer. Methods Engrg. 32 (1991) 1205–1227.
[12] F. Feyel, Multiscale FE2 elastoviscoplastic analysis of composite structures, Comput. Mater. Sci. 16 (1–4) (1999) 344–354.
[13] P. Gosselet, C. Rey, Non-overlapping domain decomposition methods in structural mechanics, Arch. Comput. Methods Eng. 13 (2006) 515–572.
[14] E. Jones, T. Oliphant, P. Peterson, et al., SciPy: Open source scientific tools for Python, [Online; accessed <today >] (2001–). URL http://www.scipy.org/.
[15] T. Kanit, S. Forest, I. Galliet, V. Mounoury, D. Jeulin, Determination of the size of the representative volume element for random composites: statistical and numerical approach, Int. J. Solids Struct. 40 (2003) 3647–3679.
[16] P. Ladeveze, A. Nouy, On a multiscale computational strategy with time and space homogenization for structural mechanics, Comput. Methods Appl. Mech. Engrg. 192 (28–30) (2003) 3061–3087.
[17] P. Lions, On the Schwarz alternating method. I, in: First International Symposium on Domain Decomposition Methods for Partial Differential Equations, 1988, pp. 1–42..
[18] P.L. Lions, On the Schwarz alternating method. II, in: Domain Decomposition Methods, 1989, pp. 47–70.
[19] J. Mandel, Balancing domain decomposition, Comm. Numer. Meth. Engng. 9 (1993) 233–241.
[20] S. Metoui, E. Pruliere, A. Ammar, F. Dau, I. Iordanoff, The proper generalized decomposition for the simulation of delamination using cohesive zone model, Internat. J. Numer. Methods Engrg. 99 (13) (2014) 1000–1022.
[21] A. Nouy, O.L. Maître, Generalized spectral decomposition for stochastic nonlinear problems, J. Comput. Phys. 228 (1) (2009) 202–235.
[22] E. Pruliere, 3D simulation of laminated shell structures using the proper generalized decomposition, Compos. Struct. 117 (2014) 373–381.
[23] E. Pruliere, F. Chinesta, A. Ammar, On the deterministic solution of multidimensional parametric models using the proper generalized decomposition, Math. Comput. Simulation 81 (4) (2010) 791–810.
[24] H.A. Schwarz, Über einen Grenzübergang durch alternierendes Verfahren, in: Vierteljahrsschrift Der Naturforschenden Gesellschaft in Zürich 15, 1870, pp. 272–286..
[25] TexGen (version 3.7), University of Nottingham, 2014, http://http://texgen.sourceforge.net/ (Accessed: 17.05.16).
[26] P. Vidal, L. Gallimard, O. Polit, Composite beam finite element based on the proper generalized decomposition, Comput. Struct. 102–103 (2012) 76–86.
[27] P. Vidal, L. Gallimard, O. Polit, Proper Generalized Decomposition and layer-wise approach for the modeling of composite plate structures, Int. J. Solids Struct. 50 (14–15) (2013) 2239–2250.
[28] S.L. Xiaoye, An overview of SuperLU: Algorithms, implementation, and user interface, ACM Trans. Math. Software (TOMS) 31 (3) (2005) 302–325.