A k-medoids-based partitioned method for computational homogenization of heterogeneous materials

Modesar Shakoor

A k-medoids-based partitioned method for computational homogenization of heterogeneous materials

Modesar Shakoor*

IMT Nord Europe, Institut Mines-Télécom, Univ. Lille, Centre for Materials and Processes, F-59000 Lille, France

December 30, 2024

Abstract

Although they are very interesting for structural simulations involving heterogeneous materials, Finite Element (FE) squared approaches, often coined FE×FE or FE^2, are well-known to require great computating resources. The main challenge is that, for each integration point of the *coarse* structure, a so-called *fine* scale problem should be solved. In this work, a k-medoids-based partitioned FE^2 approach is proposed to directly tackle this challenge by effectively reducing the number of fine scale solves. At each coarse scale nonlinear iteration, coarse scale integration points are partitioned *a priori* based on the current coarse scale displacement gradient and fine scale internal variables using the k-medoids-based clustering algorithm. Stresses and tangent modulii are computed only for cluster medoids, and are then

*Corresponding author: modesar.shakoor@imt-nord-europe.fr

extended to the remaining non-medoid coarse scale integration points. Results with nonlinear material behavior such as hyperelasticity and elasto-plasticity show that the proposed method is a promising candidate for reducing the computational cost of FE^2 simulations.

Keywords: computational homogenization, FE×FE, FE^2, clustering, k-medoids, reduced-order modeling

Article highlights:

- Proposition of a k-medoids-based partitioned finite element squared approach
- Linearization with cross contributions within clusters and analysis of their effect
- Solution of fine scale problems after convergence to update internal variables
- Simplification of the nonlinear solution algorithm

1 Introduction

Numerical modeling of heterogeneous materials is well-known to be challenging because the simultaneous discretization of the heterogeneity at the scale of the simulation domain is often intractable computationally. For instance, fiber-reinforced composite structures typically contain millions of fibers. The idea of separating the fine scale of the heterogeneity from the coarse scale of the domain or structure through homogenization theory has been investigated in various studies Geers et al. (2010); Matouš et al. (2017). In this approach, as shown in Fig. 1, the heterogeneity is modeled through small unit cells or Representative Volume Elements (RVEs), which alleviates the computational burden substantially. Boundary conditions for fine scale problems are provided by point-wise strains from the coarse scale, while stresses and tangent modulii for the coarse scale problem are provided by homogenized stresses and tangent modulii from the fine scale Geers et al. (2010); Matouš et al. (2017).

Even with this scale separation, however, the computational cost remains an issue because a fine scale problem should be solved for each point of the coarse scale domain. For nonlinear problems, in addition, all fine scale computations should happen on-the-fly during the coarse scale simulation with two-way coupling between all solvers (coarse and

fine). This is the reason why various approaches can be found in the literature to solve the fine scale problems, even though the coarse scale problem is often discretized and solved using the Finite Element (FE) method Geers et al. (2010); Matouš et al. (2017). In the FE×FE or FE^2 method (Feyel, 1999), fine scale problems are also solved, in their full complexity, with the FE method. Alternative full-field methods such as the fast Fourier transform based numerical method have also been proposed in the literature to solve fine scale problems (Gierden et al., 2022). Simplifying assumptions have been employed in many works to reduce the computational cost of fine scale problems, for instance using mean-field models such as the Hashin-Shtrikman model (Hashin and Shtrikman, 1963) or the Mori-Tanaka model (Mori and Tanaka, 1973). In the recent literature, many researchers have attempted to reduce the computational cost without such assumptions, mainly using reduced-order models such as proper orthogonal decomposition (Yvonnet and He, 2007) or k-means clustering (Gao et al., 2020). Recent advances in the field of machine learning and deep learning have also been exploited (Le et al., 2015).

An interesting and promising alternative path has recently been proposed in Refs. (Benaimeche et al., 2022; Chaouch and Yvonnet, 2024a). Instead of attempting to simplify or reduce fine scale problems, this alternative approach called Partitioned FE^2 (PFE^2) in the following consisted in applying model-order reduction directly at the coarse scale to effectively reduce the number of fine scale problems to solve. At each nonlinear iteration of the coarse scale solver, coarse scale integration points were partitioned using k-means clustering with some *a priori* criterion, and only one fine scale problem was solved per cluster. Stresses were then computed for all coarse scale integration points by using an approximation which avoided solving all fine scale problems. This approach has been shown to provide accurate results with reduced computation times as compared to the FE^2 method, first for hyperelastic, viscoelastic and elasto-plastic material behaviors (Chaouch and Yvonnet, 2024a), and then for problems involving localization and damage (Chaouch and Yvonnet, 2024b).

Although a number of issues with the initial version have been addressed in Refs. (Chaouch and Yvonnet, 2024a,b), a few challenges remain with the PFE^2 method and are the object of the present work:

- K-means clustering has been used in Refs. (Benaimeche et al., 2022; Chaouch and

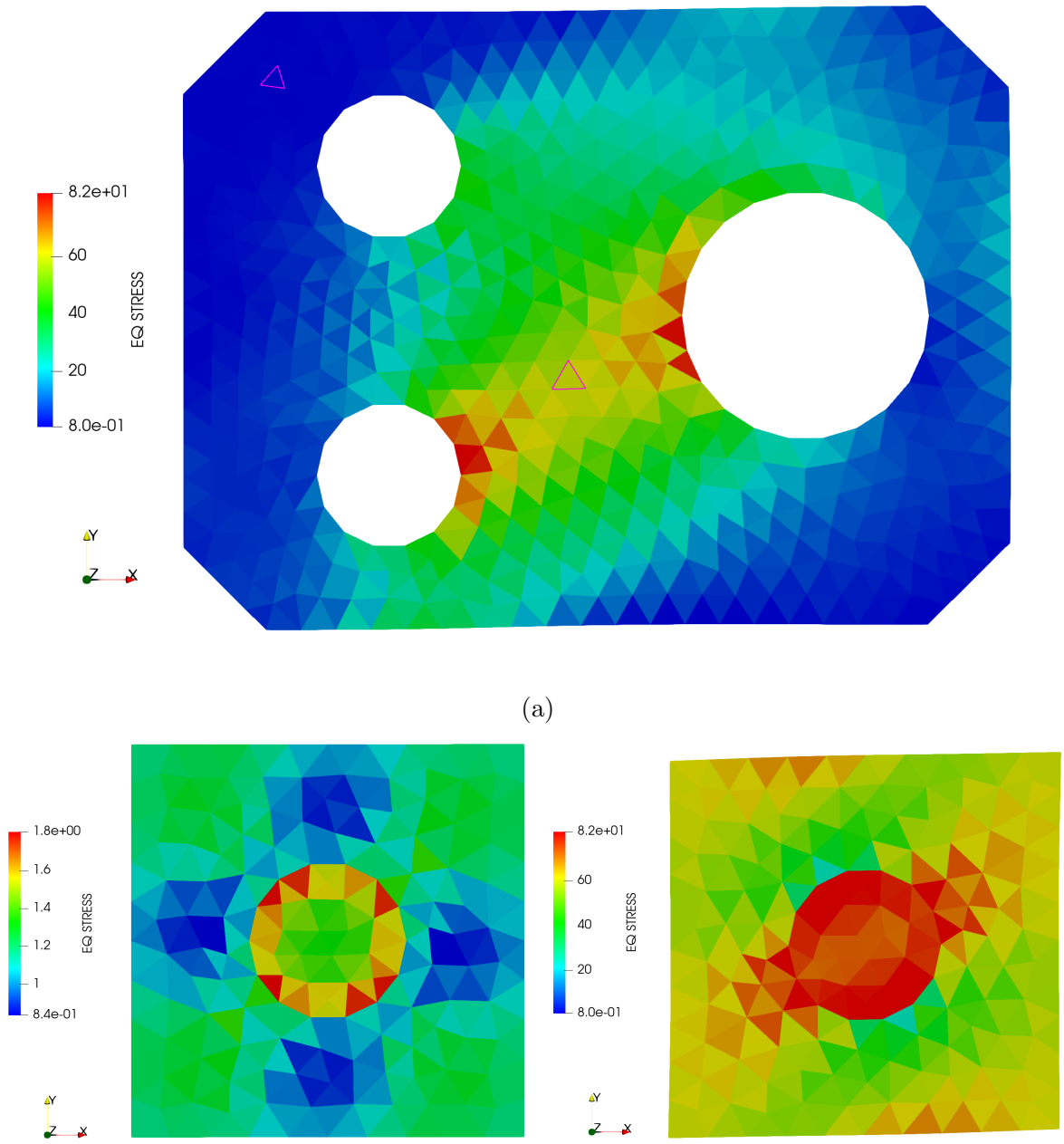


Figure 1: Example of FE^2 result showing the equivalent stress field: (a) at the coarse scale, (b) at the fine scale for a coarse scale integration point at the top left corner, (c) at the fine scale for a coarse scale integration point between holes B and C.

Yvonnet, 2024a) and the fine scale problem solved for each cluster was some virtual fine scale problem whose loading conditions corresponded to some average of all fine scale problems within the cluster. This approach required to create and maintain a virtual model with its virtual internal variables for each cluster.

- At each clustering change, virtual internal variables were reassigned between old and new clusters through a mapping algorithm. For each new cluster, this algorithm consisted in looking for the closest old cluster and copying its internal variables.

- Challenging convergence issues were revealed in Ref. (Chaouch and Yvonnet, 2024a) and an elaborate nonlinear solution algorithm relying on clustering freezing and iteration restarts was proposed to address them.

In the present work, the following original improvements are proposed to the PFE^2 method:

- K-means clustering is replaced by k-medoids clustering. This partitioning technique defines as cluster centroid one of the fine scale problems within the cluster (*i.e.*, the cluster's medoid). Consequently, stresses and tangent modulii are computed from actual fine scale problems, which effectively avoids the introduction of a virtual fine scale problem.

- The approximation proposed in Ref. (Chaouch and Yvonnet, 2024a) is still employed, but it is shown that a second-order tangent modulus appears as well as cross contributions between all integration points within a cluster. The influence of these cross contributions on convergence is investigated.

- Internal variables are systematically updated for all fine scale problems, in addition to the ones associated to cluster medoids. At any step, due to clustering change, any integration point can become a cluster medoid and be solved.

- A simplified nonlinear solution algorithm based on clustering freezing, time-step reduction and fall back to the FE^2 scheme is proposed to tackle convergence issues. Except for the requirement to freeze clustering, this algorithm is very similar to the one used for the FE^2 scheme for stiff or strongly non linear problems.

Computational homogenization is introduced in Sec. 2. The new k-medoids-based PFE^2 method is then detailed in Sec. 3. Numerical results demonstrating the capabilities of this new method are presented in Sec. 4. Sec. 5 finally concludes the paper.

2 Governing equations

Under the assumption of scale separation, homogenization consists in introducing at each point $\boldsymbol{X}$ of the coarse scale domain $\Omega_0^M \subset \mathbb{R}^3$ a fine scale domain $\Omega_0^m = \Omega_0^m(\boldsymbol{X})$. The latter is either a repetitive unit cell or an RVE which represents the heterogeneity only for a small window. For fiber-reinforced composites for instance, this domain might contain a single fiber or up to hundred fibers with some particular distribution carefully calibrated with experimental measurements (*e.g.*, microscopy or tomography).
In the following, the coarse scale problem is first stated, and then the fine scale problem. This description is given under the assumption of finite strains, and an example of material model with no internal variable for the fine scale problem is also given to complete the problem statement. Finally, simplified equations are formulated under the assumption of small strains with an elasto-plastic material model which features an internal variable at the fine scale.

2.1 Coarse scale problem

The coarse scale domain is assumed homogeneous, and the unknown displacement $\boldsymbol{u}^M \in H^1(\Omega_0^M)^3$ is defined as the difference between the deformed coordinate $\boldsymbol{x}$ at some instant $t \in [0, T], T > 0$ and the initial coordinate $\boldsymbol{X}$. Neglecting body forces, this coarse scale displacement field can be obtained by solving the balance equation:

$$\nabla_{\boldsymbol{X}} . \boldsymbol{P}^M \left(\nabla_{\boldsymbol{X}} \boldsymbol{u}^M(\boldsymbol{X}) \right) = \boldsymbol{0}, \forall \boldsymbol{X} \in \Omega_0^M, \tag{1}$$

with appropriate boundary conditions. Functional spaces are

$$\begin{aligned} H^1(\Omega_0^M) &= \{ v \in L^2(\Omega_0^M), \nabla_{\boldsymbol{X}} v \in L^2(\Omega_0^M)^3 \}, \\ L^2(\Omega_0^M) &= \{ v : \Omega_0^M \longrightarrow \mathbb{R}, \int_{\Omega_0^M} v^2 \mathrm{d}\Omega_0^M < +\infty \}. \end{aligned} \tag{2}$$

2.2 Fine scale problem

As shown explicitly in Eq. (1), at each point $\boldsymbol{X} \in \Omega_0^M$, the first Piola-Kirchhoff stress tensor $\boldsymbol{P}^M$ depends on the displacement gradient $\nabla_{\boldsymbol{X}} \boldsymbol{u}^M(\boldsymbol{X})$. In computational homogenization, it is actually computed by solving the following fine scale problem:

$$\begin{cases} \nabla_{\boldsymbol{Y}} . \boldsymbol{P}^m \left(\nabla_{\boldsymbol{Y}} \boldsymbol{u}^m(\boldsymbol{Y}) \right) = \boldsymbol{\alpha}, \forall \boldsymbol{Y} \in \Omega_0^m, \\ \boldsymbol{P}^m \left(\nabla_{\boldsymbol{Y}} \boldsymbol{u}^m(\boldsymbol{Y}) \right) . \boldsymbol{N}^m(\boldsymbol{Y}) = \boldsymbol{\beta} . \boldsymbol{N}^m(\boldsymbol{Y}), \forall \boldsymbol{Y} \in \partial\Omega_0^m, \\ \dfrac{1}{|\Omega_0^m|} \displaystyle\int_{\Omega_0^m} \boldsymbol{u}^m(\boldsymbol{Y}) \mathrm{d}\boldsymbol{Y} = \boldsymbol{u}^M(\boldsymbol{X}), \\ \dfrac{1}{|\Omega_0^m|} \displaystyle\int_{\Omega_0^m} \nabla_{\boldsymbol{Y}} \boldsymbol{u}^m(\boldsymbol{Y}) \mathrm{d}\boldsymbol{Y} = \nabla_{\boldsymbol{X}} \boldsymbol{u}^M(\boldsymbol{X}), \end{cases} \tag{3}$$

where $\boldsymbol{P}^m$ is the first Piola-Kirchhoff stress tensor field at the fine scale and $\boldsymbol{N}^m$ is the outgoing normal vector at the fine scale domain boundary.

This fine scale problem is derived from the principle of multiscale virtual power (Blanco et al., 2016). As previously developed for unsteady flows in porous media (Shakoor and Park, 2023), this approach can rely on periodic boundary conditions as most works on computational homogenization, but loading conditions can also be imposed at the fine scale through Lagrange multipliers over the whole fine scale domain. In Eq. (3), indeed, averages of the fine scale displacement field and its gradient are tied to be equal, respectively, to the coarse scale displacement and its gradient through Lagrange multipliers $\boldsymbol{\alpha} \in \mathbb{R}^3$ and $\boldsymbol{\beta} \in \mathbb{R}^{3\times3}$. The fine scale problem in Eq. (3) should hence be simultaneously solved to find both Lagrange multipliers and the fine scale displacement field $\boldsymbol{u}^m \in H^1(\Omega_0^m)^3$.

On the one hand, as acceleration is neglected, $\boldsymbol{\alpha}$ has solely the role of suppressing rigid body translations at the fine scale. On the other hand, the coarse scale stress tensor can be computed through $\boldsymbol{P}^M(\boldsymbol{X}) = \boldsymbol{\beta}$. In order to solve the coarse scale problem in Eq. (1), linearization is typically used, which requires to compute $\dfrac{\partial \boldsymbol{P}^M}{\partial \nabla_{\boldsymbol{X}} \boldsymbol{u}^M}(\boldsymbol{X}) = \dfrac{\partial \boldsymbol{\beta}}{\partial \nabla_{\boldsymbol{X}} \boldsymbol{u}^M}$.

2.3 Example of finite strain model

It is reminded that the first Piola-Kirchhoff stress tensor is defined by $\boldsymbol{P}^m = \boldsymbol{F}^m \boldsymbol{S}^m$, with $\boldsymbol{F}^m$ the deformation gradient tensor defined by $\boldsymbol{F}^m = \boldsymbol{I} + \nabla_{\boldsymbol{Y}} \boldsymbol{u}^m$, $\boldsymbol{I}$ the second-order identity tensor and $\boldsymbol{S}^m$ the second Piola-Kirchhoff stress tensor. For instance, the

isotropic Saint-Venant-Kirchhoff model is given by

$$\boldsymbol{S}^m = 2\mu^m \boldsymbol{E}^m + \lambda^m tr(\boldsymbol{E}^m), \tag{4}$$

with $\boldsymbol{E}^m = \frac{1}{2}\left(\left(\boldsymbol{F}^m\right)^T \boldsymbol{F}^m - \boldsymbol{I}\right)$ the Green-Lagrange strain tensor and $tr(\boldsymbol{A}) = \sum_{i=1}^{3} \boldsymbol{A}_{ii}$ the trace operator. It should be noted that Lamé parameters $\mu^m = \mu^m(\boldsymbol{Y})$ and $\lambda^m = \lambda^m(\boldsymbol{Y})$ are heterogeneous as, for instance, the stiffness of a carbon fiber is typically different from that of a polymer matrix.

2.4 Example of small strain model

Firstly, under the small strains assumption, the coarse scale problem simplifies to

$$\nabla_{\boldsymbol{X}}.\boldsymbol{\sigma}^M\left(\nabla_{\boldsymbol{X}}\boldsymbol{u}^M(\boldsymbol{X})\right) = \boldsymbol{0}, \forall \boldsymbol{X} \in \Omega_0^M, \tag{5}$$

with $\boldsymbol{\sigma}^M$ the Cauchy stress tensor. Secondly, the fine scale problem simplifies to

$$\begin{cases} \nabla_{\boldsymbol{Y}}.\boldsymbol{\sigma}^m\left(\nabla_{\boldsymbol{Y}}\boldsymbol{u}^m(\boldsymbol{Y})\right) = \boldsymbol{\alpha}, \forall \boldsymbol{Y} \in \Omega_0^m, \\ \boldsymbol{\sigma}^m\left(\nabla_{\boldsymbol{Y}}\boldsymbol{u}^m(\boldsymbol{Y})\right).\boldsymbol{N}^m(\boldsymbol{Y}) = \boldsymbol{\beta}.\boldsymbol{N}^m(\boldsymbol{Y}), \forall \boldsymbol{Y} \in \partial\Omega_0^m, \\ \dfrac{1}{|\Omega_0^m|}\displaystyle\int_{\Omega_0^m} \boldsymbol{u}^m(\boldsymbol{Y})\mathrm{d}\boldsymbol{Y} = \boldsymbol{u}^M(\boldsymbol{X}), \\ \dfrac{1}{|\Omega_0^m|}\displaystyle\int_{\Omega_0^m} \nabla_{\boldsymbol{Y}}\boldsymbol{u}^m(\boldsymbol{Y})\mathrm{d}\boldsymbol{Y} = \nabla_{\boldsymbol{X}}\boldsymbol{u}^M(\boldsymbol{X}). \end{cases} \tag{6}$$

The von Mises elasto-plastic model with isotropic elasticity and combined isotropic and kinematic hardening, for instance, is given by

$$\begin{aligned} \boldsymbol{\sigma}^m &= 2\mu^m \boldsymbol{\varepsilon}^{m,e} + \lambda^m tr(\boldsymbol{\varepsilon}^{m,e}), \\ \boldsymbol{\varepsilon}^{m,e} &= \boldsymbol{\varepsilon}^m - \boldsymbol{\varepsilon}^{m,p} \\ \boldsymbol{\varepsilon}^m &= \frac{1}{2}\left(\nabla_{\boldsymbol{Y}}\boldsymbol{u}^m + \nabla_{\boldsymbol{Y}}^T\boldsymbol{u}^m\right), \end{aligned} \tag{7}$$

where an additive decomposition into an elastic part $\boldsymbol{\varepsilon}^{m,e}$ and a plastic part $\boldsymbol{\varepsilon}^{m,p}$ has been assumed for the small strain tensor $\boldsymbol{\varepsilon}^m$. Plastic flow is ruled by the condition $\overline{f}\left(\boldsymbol{\sigma}^{m,dev} - \boldsymbol{\gamma}^m\right) - \overline{\sigma}(\overline{\varepsilon}^{m,p}) \leq 0$, with the von Mises equivalent stress $\overline{f}$ given by

$$\begin{aligned} \overline{f}\left(\boldsymbol{\sigma}^{m,dev} - \boldsymbol{\gamma}^m\right) &= \sqrt{\frac{3}{2}\left(\boldsymbol{\sigma}^{m,dev} - \boldsymbol{\gamma}^m\right) : \left(\boldsymbol{\sigma}^{m,dev} - \boldsymbol{\gamma}^m\right)}, \\ \boldsymbol{\sigma}^{m,dev} &= \boldsymbol{\sigma}^m - \frac{1}{3}tr(\boldsymbol{\sigma}^m), \end{aligned} \tag{8}$$

and the yield stress $\overline{\sigma}$ given by

$$\begin{aligned} \overline{\sigma}(\overline{\boldsymbol{\varepsilon}}^{m,p}) &= \sigma_Y + K\overline{\sigma}(\overline{\boldsymbol{\varepsilon}}^{m,p}), \\ \overline{\boldsymbol{\varepsilon}}^{m,p} &= \int_0^t \dot{\overline{\boldsymbol{\varepsilon}}}^{m,p}(s)ds, \\ \dot{\overline{\boldsymbol{\varepsilon}}}^{m,p} &= \sqrt{\frac{2}{3}\dot{\boldsymbol{\varepsilon}}^{m,p} : \dot{\boldsymbol{\varepsilon}}^{m,p}}. \end{aligned} \tag{9}$$

Material parameter σ_Y is the initial yield stress and K is the isotropic hardening modulus, while internal variable $\overline{\boldsymbol{\varepsilon}}^{m,p}$ is the equivalent plastic strain. The latter is defined as the integral over time of the equivalent plastic strain rate $\dot{\overline{\boldsymbol{\varepsilon}}}^{m,p}$. Under plastic flow (*i.e.*, when $\overline{f}\left(\boldsymbol{\sigma}^{m,dev} - \boldsymbol{\gamma}^m\right) - \overline{\sigma}(\overline{\boldsymbol{\varepsilon}}^{m,p}) = 0$), the plastic strain rate is given by

$$\dot{\boldsymbol{\varepsilon}}^{m,p} = \dot{\overline{\boldsymbol{\varepsilon}}}^{m,p}\frac{3}{2}\frac{\boldsymbol{\sigma}^{m,dev}}{\overline{f}}. \tag{10}$$

Kinematic hardening is modeled using a linear law where

$$\dot{\boldsymbol{\gamma}}^m = H\dot{\boldsymbol{\varepsilon}}^{m,p}, \tag{11}$$

H being the kinematic hardening modulus.

The Next Increment Corrects Error (NICE) scheme is used to integrate this nonlinear material law (Halilovič et al., 2009).

3 Numerical method

In this section, the novel k-medoids-based PFE^2 approach is presented. Details on the reference FE^2 algorithm are given in the Appendix. The idea of the PFE^2 approach as proposed in Refs. (Benaimeche et al., 2022; Chaouch and Yvonnet, 2024a) is to group coarse scale integration points into a user-defined number of clusters. In the present work, it is proposed to do so using the k-medoids clustering algorithm. This clustering is performed at each iteration of the coarse scale solver for the first N_k iterations, and then clustering is frozen for the remaining iterations. At each iteration, fine scale problems are only solved for the cluster medoid, and an approximation is used to compute stresses for all non-medoid integration points.

3.1 Clustering

The input variable $\boldsymbol{C}$ for the k-medoids clustering algorithm is simply the concatenation of the current coarse scale displacement gradient guess at each coarse scale integration point, and the averaged fine scale internal variables over each fine scale domain. For the finite strain model described in Subsec. 2.3, therefore, the input variable is:

$$\boldsymbol{C}(\boldsymbol{X}_p) = \left\{ \left(\nabla_{\boldsymbol{X}} \boldsymbol{u}_{kl}^{M,n+1,i}(\boldsymbol{X}_p) \right)_{k,l=1\ldots3} \right\}, \tag{12}$$

where p is the integration point number. For the small strain model in Subsec. 2.4, the input variable is changed to:

$$\boldsymbol{C}(\boldsymbol{X}_p) = \left\{ \left(\nabla_{\boldsymbol{X}} \boldsymbol{u}_{kl}^{M,n+1,i}(\boldsymbol{X}_p) \right)_{k,l=1\ldots3}, \overline{\varepsilon}^{M,p,n+1,i}(\boldsymbol{X}_p), \left(\varepsilon_{kl}^{M,p,n+1,i}(\boldsymbol{X}_p) \right)_{k,l=1\ldots3} \right\}, \tag{13}$$

where $\overline{\varepsilon}^{M,p,n+1,i} = \int_{\Omega_0^m} \overline{\varepsilon}^{m,p,n+1,i} \mathrm{d}\Omega_0^m$, $\varepsilon_{kl}^{M,p,n+1,i} = \int_{\Omega_0^m} \varepsilon_{kl}^{m,p,n+1,i} \mathrm{d}\Omega_0^m$, $\Omega_0^m = \Omega_0^m(\boldsymbol{X}_p)$ are computed only to incorporate history information into clustering and do not have a physical meaning.

In the k-medoids clustering algorithm, each cluster is identified by its medoid, which is an integration point of the coarse scale FE mesh. The algorithm requires the definition of a distance between cluster medoids. Here the following Euclidean distance is used:

$$d(\boldsymbol{X}_p, \boldsymbol{X}_q) = \sum_i \left(\boldsymbol{C}_i(\boldsymbol{X}_p) - \boldsymbol{C}_i(\boldsymbol{X}_q) \right)^2. \tag{14}$$

All distances are pre-computed and stored into a symmetric distance matrix. As presented in Ref. (de Hoon et al., 2004), the k-medoids clustering algorithm then consists in:

- Recovering the last cluster assignment, if any, or randomly assigning integration points to clusters.
- Computing the medoid of each cluster as the integration point with the smallest sum of distances to the other integration points in the cluster.
- Iterating:
  - Comparing the distance between each integration point and all current cluster medoids.

– Reassigning each integration point to the closest cluster (*i.e.*, the one whose current medoid is the closest).

– Recomputing cluster medoids.

– Testing for convergence and stopping if the criterion is met under some tolerance.

As detailed in Ref. (de Hoon et al., 2004), the clustering convergence criterion consists in checking for cluster medoids changes over consecutive iterations and also over small cycles of a few iterations to check whether the algorithm is not cycling over the same periodic solution.

It should be noted that the input variable $\boldsymbol{C}$ might be constant or lead to a distance matrix mostly filled by zeroes, especially at the first nonlinear iteration of the first time increment. Due to the random initialization of clusters, the clustering algorithm still respects the user-defined number of clusters in such cases, even though cluster sizes might be uneven.

3.2 Computation of stresses and tangent modulii

As proposed in Ref. (Chaouch and Yvonnet, 2024a), stresses of all integration points within a cluster are computed through the approximation:

$$\begin{aligned} \boldsymbol{P}^M\left(\nabla_{\boldsymbol{X}}\boldsymbol{u}^M(\boldsymbol{X}_p)\right) = \; & \boldsymbol{P}^M\left(\nabla_{\boldsymbol{X}}\boldsymbol{u}^M(\boldsymbol{X}_c)\right) \\ & + \frac{\partial \boldsymbol{P}^M}{\partial \nabla_{\boldsymbol{X}}\boldsymbol{u}^M}(\nabla_{\boldsymbol{X}}\boldsymbol{u}^M(\boldsymbol{X}_c)) : \left(\nabla_{\boldsymbol{X}}\boldsymbol{u}^M(\boldsymbol{X}_p) - \nabla_{\boldsymbol{X}}\boldsymbol{u}^M(\boldsymbol{X}_c)\right), \end{aligned} \tag{15}$$

or, introducing $\boldsymbol{G}_p = \nabla_{\boldsymbol{X}}\boldsymbol{u}^M(\boldsymbol{X}_p)$ and $\boldsymbol{G}_c = \nabla_{\boldsymbol{X}}\boldsymbol{u}^M(\boldsymbol{X}_c)$:

$$\begin{aligned} \boldsymbol{P}^M\left(\boldsymbol{G}_p\right) = \; & \boldsymbol{P}^M\left(\boldsymbol{G}_c\right) \\ & + \frac{\partial \boldsymbol{P}^M}{\partial \nabla_{\boldsymbol{X}}\boldsymbol{u}^M}(\boldsymbol{G}_c) : \left(\boldsymbol{G}_p - \boldsymbol{G}_c\right). \end{aligned} \tag{16}$$

It is clear from this expression that

$$\frac{\partial \boldsymbol{P}^M}{\partial \boldsymbol{G}_p}(\boldsymbol{G}_p) = \frac{\partial \boldsymbol{P}^M}{\partial \nabla_{\boldsymbol{X}}\boldsymbol{u}^M}(\boldsymbol{G}_c). \tag{17}$$

The authors of Ref. (Chaouch and Yvonnet, 2024a), however, did not note that

$$\begin{aligned}\frac{\partial \boldsymbol{P}^M}{\partial \boldsymbol{G}_c}(\boldsymbol{G}_p) &= \frac{\partial \boldsymbol{P}^M}{\partial \nabla_{\boldsymbol{X}} \boldsymbol{u}^M}(\boldsymbol{G}_c) - \frac{\partial \boldsymbol{P}^M}{\partial \nabla_{\boldsymbol{X}} \boldsymbol{u}^M}(\boldsymbol{G}_c) \\ &\quad + \frac{\partial^2 \boldsymbol{P}^M}{\partial \left(\nabla_{\boldsymbol{X}} \boldsymbol{u}^M\right)^2}(\boldsymbol{G}_c) : (\boldsymbol{G}_p - \boldsymbol{G}_c) \\ &= \frac{\partial^2 \boldsymbol{P}^M}{\partial \left(\nabla_{\boldsymbol{X}} \boldsymbol{u}^M\right)^2}(\boldsymbol{G}_c) : (\boldsymbol{G}_p - \boldsymbol{G}_c)\,.\end{aligned} \tag{18}$$

During the linearization of approximation (15), therefore, cross contributions arise. In Ref. (Chaouch and Yvonnet, 2024a), the authors used k-means clustering and defined $\boldsymbol{G}_c$ as the weighted average of all displacement gradients throughout each cluster. Cross contributions would have paired all integration points of a cluster with each other, thus reducing drastically the sparsity of the linearized coarse scale problem.

In the present work, because $\boldsymbol{G}_c$ is the displacement gradient at the cluster's medoid, cross contributions only pair each integration point with its cluster's medoid. It is reminded that, at each nonlinear iteration of the coarse scale solver, the weak form of Eq. (1) features the term $\displaystyle\int_{\Omega^M} \boldsymbol{P}^M(\boldsymbol{G}_p^{n+1,i+1}) : \nabla_{\boldsymbol{X}} \boldsymbol{v}^M(\boldsymbol{X}_p) \mathrm{d}\boldsymbol{X}_p$. As a consequence of Eqs. (18) and (17), it is linearized as:

$$\begin{aligned}&\int_{\Omega^M} \boldsymbol{P}^M(\boldsymbol{G}_p^{n+1,i+1}) : \nabla_{\boldsymbol{X}} \boldsymbol{v}^M(\boldsymbol{X}_p) \mathrm{d}\boldsymbol{X}_p \\ &= \int_{\Omega^M} \boldsymbol{P}^M(\boldsymbol{G}_p^{n+1,i}) : \nabla_{\boldsymbol{X}} \boldsymbol{v}^M(\boldsymbol{X}_p) \mathrm{d}\boldsymbol{X}_p \\ &\quad + \int_{\Omega^M} \left(\frac{\partial \boldsymbol{P}^M}{\partial \boldsymbol{G}_p}(\boldsymbol{G}_p^{n+1,i}) : \Delta \boldsymbol{G}_p^{n+1,i+1} \right) : \nabla_{\boldsymbol{X}} \boldsymbol{v}^M(\boldsymbol{X}_p) \mathrm{d}\boldsymbol{X}_p \\ &\quad + \int_{\Omega^M} \left(\frac{\partial \boldsymbol{P}^M}{\partial \boldsymbol{G}_c}(\boldsymbol{G}_p^{n+1,i}) : \Delta \boldsymbol{G}_c^{n+1,i+1} \right) : \nabla_{\boldsymbol{X}} \boldsymbol{v}^M(\boldsymbol{X}_p) \mathrm{d}\boldsymbol{X}_p,\end{aligned} \tag{19}$$

where $\Delta \boldsymbol{G}_p^{n+1,i+1} = \boldsymbol{G}_p^{n+1,i+1} - \boldsymbol{G}_p^{n+1,i}$, $\Delta \boldsymbol{G}_c^{i+1} = \boldsymbol{G}_c^{i+1} - \boldsymbol{G}_c^{i}$, and $\boldsymbol{v}^M$ is the test displacement field. If the last term of Eq. (19) is not neglected, then lines of the linearized coarse scale problem associated to the nodes of the element containing integration point $\boldsymbol{X}_p$ receive non-zero contributions for the columns associated to the nodes of the element containing cluster medoid $\boldsymbol{X}_c$. This should reduce slightly the sparsity of the linearized coarse scale problem.

The numerical differentiation formula to compute the second-order derivative tensor $\dfrac{\partial^2 \boldsymbol{P}^M}{\partial \left(\nabla_{\boldsymbol{X}} \boldsymbol{u}^M\right)^2}(\boldsymbol{G}_c^{n+1,i})$, moreover, is quite elaborated and requires a great number of fine

scale solves:

$$\begin{aligned}
\frac{\partial^2 \boldsymbol{P}_{hj}^M}{\partial \nabla_{\boldsymbol{X}} \boldsymbol{u}_{kl}^M \partial \nabla_{\boldsymbol{X}} \boldsymbol{u}_{rs}^M}(\boldsymbol{G}_c^{n+1,i}) &\approx \frac{\frac{\partial \boldsymbol{P}_{hj}^M}{\partial \nabla_{\boldsymbol{X}} \boldsymbol{u}_{kl}^M}(\boldsymbol{G}_c^{n+1,i} + \frac{\sqrt{\epsilon}}{2}\mathbf{1}^{rs}) - \frac{\partial \boldsymbol{P}_{hj}^M}{\partial \nabla_{\boldsymbol{X}} \boldsymbol{u}_{kl}^M}(\boldsymbol{G}_c^{n+1,i} - \frac{\sqrt{\epsilon}}{2}\mathbf{1}^{rs})}{\sqrt{\epsilon}}, \\
\frac{\partial \boldsymbol{P}_{hj}^M}{\partial \nabla_{\boldsymbol{X}} \boldsymbol{u}_{kl}^M}(\boldsymbol{G}_c^{n+1,i} + \frac{\sqrt{\epsilon}}{2}\mathbf{1}^{rs}) &\approx \frac{\boldsymbol{P}_{hj}^M(\boldsymbol{G}_c^{n+1,i} + \frac{\sqrt{\epsilon}}{2}\mathbf{1}^{rs} + \frac{\sqrt{\epsilon}}{2}\mathbf{1}^{kl}) - \boldsymbol{P}_{hj}^M(\boldsymbol{G}_c^{n+1,i} + \frac{\sqrt{\epsilon}}{2}\mathbf{1}^{rs} - \frac{\sqrt{\epsilon}}{2}\mathbf{1}^{kl})}{\sqrt{\epsilon}}, \\
\frac{\partial \boldsymbol{P}_{hj}^M}{\partial \nabla_{\boldsymbol{X}} \boldsymbol{u}_{kl}^M}(\boldsymbol{G}_c^{n+1,i} - \frac{\sqrt{\epsilon}}{2}\mathbf{1}^{rs}) &\approx \frac{\boldsymbol{P}_{hj}^M(\boldsymbol{G}_c^{n+1,i} - \frac{\sqrt{\epsilon}}{2}\mathbf{1}^{rs} + \frac{\sqrt{\epsilon}}{2}\mathbf{1}^{kl}) - \boldsymbol{P}_{hj}^M(\boldsymbol{G}_c^{n+1,i} - \frac{\sqrt{\epsilon}}{2}\mathbf{1}^{rs} - \frac{\sqrt{\epsilon}}{2}\mathbf{1}^{kl})}{\sqrt{\epsilon}}.
\end{aligned} \tag{20}$$

While Eq. (A.1) requires $3 \times 3 + 1 = 10$ fine scale solves, Eq. (20) requires $3 \times 3 \times 2 \times 3 \times 3 \times 2 = 324$ solves. Note that it is possible to save more than half these solves by carefully identifying and eliminating redundant ones within Eq. (20) and between Eqs. (A.1) and (20). It is possible to save even more solves by smartly choosing either forward or centered differentiation formulas depending on components. No optimization is done in the present work to keep the implementation as simple as possible.

3.3 Update of internal variables

An elaborate mapping algorithm was proposed in Ref. (Chaouch and Yvonnet, 2024a) to map internal variables by identifying origin and destination clusters when an integration point was assigned to a new cluster. In the present work, it is chosen to systematically solve all fine scale problems after convergence of the partitioned coarse scale solver. It is crucial to keep all integration points up to date, indeed, since after each run of the k-medoids clustering algorithm, any integration point may become a cluster medoid. With this scheme, therefore, partitioning does not completely eliminate the need to solve all fine scale problems, but it saves many fine scale solves during iterations of the coarse scale solver, including those required to compute tangent modulii.

3.4 Partitioned coarse scale solver

To summarize, at each time increment, the partitioned coarse scale solver consists in:

- Running any pre-processing operation such as the management of internal variables for each fine scale problem.
- Initializing $\boldsymbol{u}^{M,n+1,0} = \boldsymbol{u}^{M,n}$.

- Iterating over $i = 0 \dots N_c$:
  - Interpolating the displacement and its gradient at each integration point of the coarse scale mesh from the current guess $\boldsymbol{u}^{M,n+1,i}$.
  - If $i < N_k$:
    - Constructing the input variable $\boldsymbol{C}$ from the coarse scale displacement gradient and fine scale internal variables averages (if any).
    - Pre-computing the distance matrix.
    - Running the k-medoids clustering algorithm.
  - Solving each fine scale problem associated to a cluster medoid as per Eq. (3) to obtain the fine scale displacement, the coarse scale stress tensor $\boldsymbol{P}^M(\boldsymbol{G}_c^{n+1,i})$, the first-order derivative tensor $\dfrac{\partial \boldsymbol{P}^M}{\partial \nabla_{\boldsymbol{X}} \boldsymbol{u}^M}(\boldsymbol{G}_c^{n+1,i})$ and, if cross contributions are to be assembled, also the second-order derivative tensor $\dfrac{\partial^2 \boldsymbol{P}^M}{\partial \left(\nabla_{\boldsymbol{X}} \boldsymbol{u}^M\right)^2}(\boldsymbol{G}_c^{n+1,i})$.
  - Computing homogenized stresses as per Eq. (15) for all fine scale problems which are not cluster medoids.
  - Assembling the linearized coarse scale problem.
  - Testing for convergence and stopping if the criterion is met under some tolerance.
  - Solving the linearized coarse scale problem to obtain the new guess $\boldsymbol{u}^{M,n+1,i+1}$.
- Solving all fine scale problems to update internal variables.
- Running any post-processing operation and writing output files.

As in the non-partitioned coarse scale solver, the time step is automatically decreased (with restarts) if any issue is encountered during the prescribed number of Newton-Raphson iterations (here N_c). If convergence is not met at the last iteration, however, it is chosen to fall back to the full FE^2 method. This is estimated to be preferable and a better solution than the retry procedure proposed in Ref. (Chaouch and Yvonnet, 2024a). The latter consists in retrying nonlinear solves with slightly changed cluster assignments with the hope that convergence might be reached. In the present work, it is

preferred to fall back to the full FE^2 method in order to get a solution in a reasonable number of expensive nonlinear iterations, instead of accumulating a great number of cheap nonlinear iterations. The occurrence of fall backs and its impact on the computational cost of the new k-medoids-based PFE^2 method is analyzed in Sec. 4. Cluster freezing is also implemented to improve convergence, by choosing $N_k < N_c$.

It should be noted that if cross contributions are omitted, the coarse scale solver is actually of quasi-Newton type, but this subtlety is voluntarily ignored throughout the paper.

4 Results

Both the FE^2 and the new k-medoids-based PFE^2 schemes have been implemented within the open source FEMS software (Shakoor, 2022, 2021). To keep computation times reasonable for the reference FE^2 simulations, two-dimensional problems are considered under the plane strains assumption. Linear triangular elements are chosen for all simulations because a single integration point can be used per element, which simplifies the visualization of cluster assignments. In the following, therefore, the words integration point may be substituted by element and vice-versa.

Unless otherwise mentioned, second-order derivative tensors and the associated cross contributions are neither computed nor assembled. Numerical simulations are conducted to evaluate the accuracy of the k-medoids-based PFE^2 method as compared to reference FE^2 results, and its interest in the point of view of computation time. The number of coarse scale iterations is set to $N_c = 32$, and the number of iterations before freezing $N_k = 7$.

Accuracy is evaluated based on the following relative L^2 norm error on the displacement vector field $\boldsymbol{u}^M$ for some k-medoids-based PFE^2 simulation as compared to a reference FE^2 displacement vector field $\boldsymbol{u}^{M,ref}$:

$$\text{Error}(\boldsymbol{u}^M) = \sqrt{\frac{\int_{\Omega_0^M} \sum_{i=1}^{2} \left(\boldsymbol{u}_i^M - \boldsymbol{u}_i^{M,ref}\right)^2 \mathrm{d}\Omega_0^M}{\int_{\Omega_0^M} \sum_{i=1}^{2} \left(\boldsymbol{u}_i^{M,ref}\right)^2 \mathrm{d}\Omega_0^M}}, \tag{21}$$

and also a similar error indicator for the Cauchy stress tensor field:

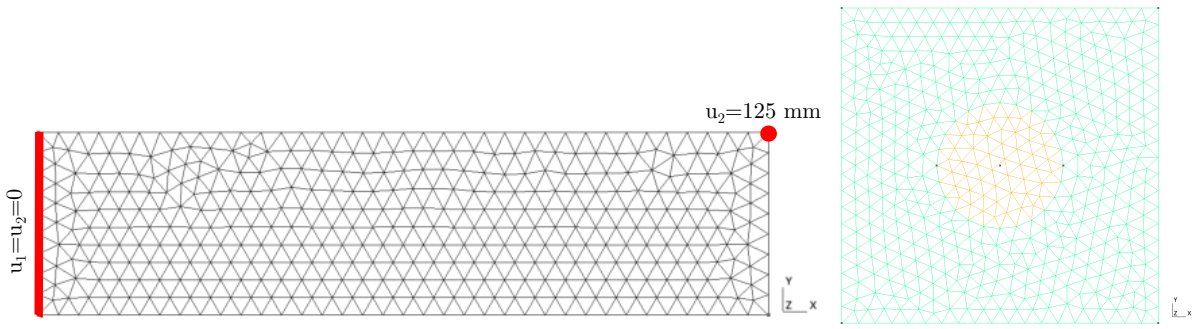
$$\text{Error}(\boldsymbol{\sigma}^M) = \sqrt{\frac{\int_{\Omega_0^M} \sum_{i,j=1}^{2} \left(\boldsymbol{\sigma}_{ij}^M - \boldsymbol{\sigma}_{ij}^{M,ref}\right)^2 \mathrm{d}\Omega_0^M}{\int_{\Omega_0^M} \sum_{i,j=1}^{2} \left(\boldsymbol{\sigma}_{ij}^{M,ref}\right)^2 \mathrm{d}\Omega_0^M}}. \tag{22}$$

The equivalent stress field computed with the formula $\sqrt{\sum_{i,j=1}^{2} \left(\boldsymbol{\sigma}_{ij}^M\right)^2}$ is shown for some results. Computation time is evaluated using a high-performance computing workstation with an *Intel(R) Xeon(R) W-2175* multi-processor with 14 CPUs and 64 GB of RAM. The code is sequential. Multi-threading or distributed parallel programming could be interesting in the future to take advantage of all the CPUs and reduce the computation time even more. Linearized problems are solved at both scales using the direct solver *UMFPACK* (Davis, 2004), which operates an LU decomposition. Computation times are always mentioned in terms of speed-up ratio relatively to the corresponding FE^2 simulation, with one meaning there was no speed-up, two meaning the PFE^2 simulation finished in half the time required for the FE^2 simulation, etc.

4.1 Hyperelastic beam

The first setup is similar to the one presented in Ref. (Chaouch and Yvonnet, 2024a). Geometries, meshes and boundary conditions are illustrated in Fig. 2. Under the assumption of plane strains, a beam of length 400 mm and height 100 mm is fixed on its left side and lifted at its top right corner with a vertical displacement of 125 mm. The fine scale domain is a square of size $1 \times 1\text{mm}^2$ composed of a soft matrix reinforced with an inclusion of radius 0.2 mm at its center. The Saint-Venant-Kirchhoff hyperelastic model presented in Subsec. 2.3 is used with a Young's modulus of 210 GPa and a Poisson's coefficient of 0.3 for the inclusion, and a Young's modulus of 2 GPa and a Poisson's coefficient of 0.25 for the matrix.

The loading is applied over five increments, and the mesh is composed of 736 linear triangular elements for the coarse scale, and 916 linear triangular elements for the fine scale. This first setup does not involve any internal variable.



(a) (b)

Figure 2: Geometries, meshes and boundary conditions for hyperelastic beam simulations: (a) coarse scale, (b) fine scale with colors distinguishing matrix and inclusion.

4.1.1 Analysis of medoid change

A first simulation is conducted with only one cluster over the whole beam. Because clustering cannot change with a single cluster, the objective is to analyze medoid changes during the simulation. Every time clustering is triggered (in the first nonlinear iterations of each time increment), the medoid position is registered. This sequence of medoid coordinates is analyzed in terms of mean and standard deviation.

The mean position is $(164\,\text{mm}, 47.1\,\text{mm})$ in the initial configuration. It is located at the center of the beam with a slight shift towards the fixed side. The standard deviation is $(21.5\,\text{mm}, 6.43\,\text{mm})$, which is small with respect to beam dimensions, and also to average mesh size (*i.e.*, $11.2\,\text{mm}$).

A large standard deviation would have revealed large fluctuations and numerical errors due to the approximation in Eq. (15), and an inability of the clustering algorithm to converge on a stable medoid for a monotonic loading. Since the standard deviation is small, the strategy proposed in the present work is shown to be relevant.

4.1.2 Convergence with an increasing number of clusters

Simulations are conducted with an increasing number of clusters in order to evaluate improvements in terms of accuracy. Equivalent stress fields are presented in Fig. 3(a,b,c) for some of those simulations. The reference FE^2 result is shown in Fig. 3(d). Although there is a slight stress overestimation at the top left side of the beam, it is corrected by

using more clusters.

This can be explained by looking at cluster assignments in Fig. 4. The number of clusters is first increased along the beam, which helps capture stress evolution in the horizontal direction. Then, a specific cluster is created for the top right corner which is a stress concentration spot, and some clusters become split in the vertical direction. These assignments are overall similar to those found in Ref. (Chaouch and Yvonnet, 2024a), especially when using a low number of clusters.

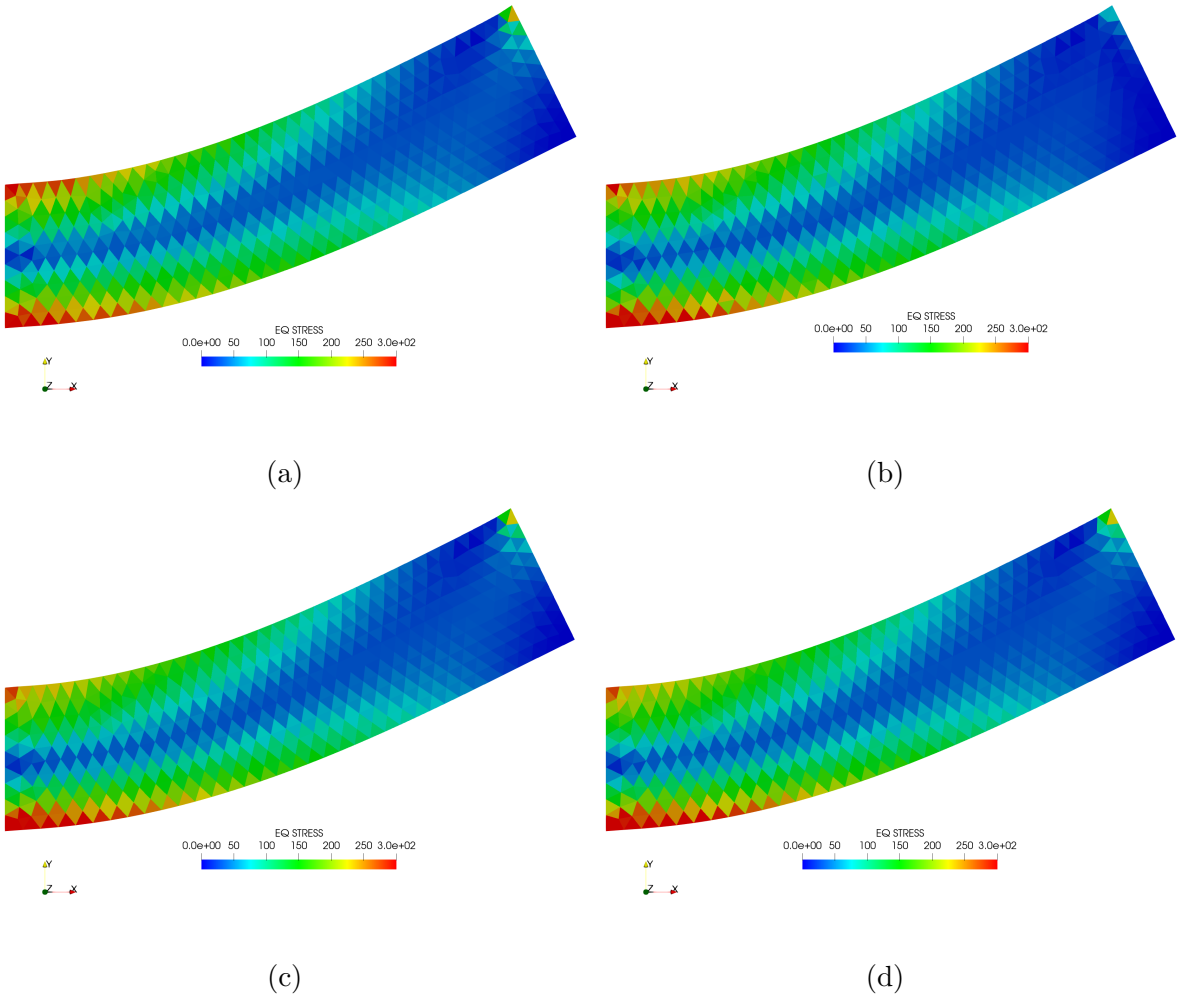
Errors are reported in Tab. 1. The relative displacement error is already quite low (below ten percent) using one cluster. For the stress, however, 23 clusters are required to reach the same bound. Regarding the convergence rate, the displacement error is divided by around two when multiplying the number of clusters by two from 1 to 2, 11 to 23, 46 to 92 and 184 to 368. The same observation applies to the stress error from 1 to 2 clusters, 5 to 11 and 11 to 23. Overall, nevertheless, convergence is much slower.

# clusters	Error($\boldsymbol{u}^M$)	Error($\boldsymbol{\sigma}^M$)	Speed-up
1	5.56×10^{-2}	2.53×10^{-1}	36.5
2	2.53×10^{-2}	1.35×10^{-1}	36.3
5	1.83×10^{-2}	1.20×10^{-1}	32.3
11	1.05×10^{-2}	6.23×10^{-2}	24.8
23	4.87×10^{-3}	3.74×10^{-2}	16.7
46	3.69×10^{-3}	3.35×10^{-2}	11.3
92	1.80×10^{-3}	2.88×10^{-2}	5.34
184	1.21×10^{-3}	2.82×10^{-2}	1.66
368	6.76×10^{-4}	2.78×10^{-2}	1.64

Table 1: Errors and speed-up ratios for hyperelastic beam simulations using an increasing number of clusters.

4.1.3 Computation time

Speed-up ratios are also reported in Tab. 1. For simulations with less than 11 clusters, the most expensive operation is solving all fine scale problems after convergence of the



(a) (b)

(c) (d)

Figure 3: Equivalent stress (in MPa) for the hyperelastic beam using: (a) five clusters, (b) 11 clusters, (c) 23 clusters, (d) the FE^2 scheme.

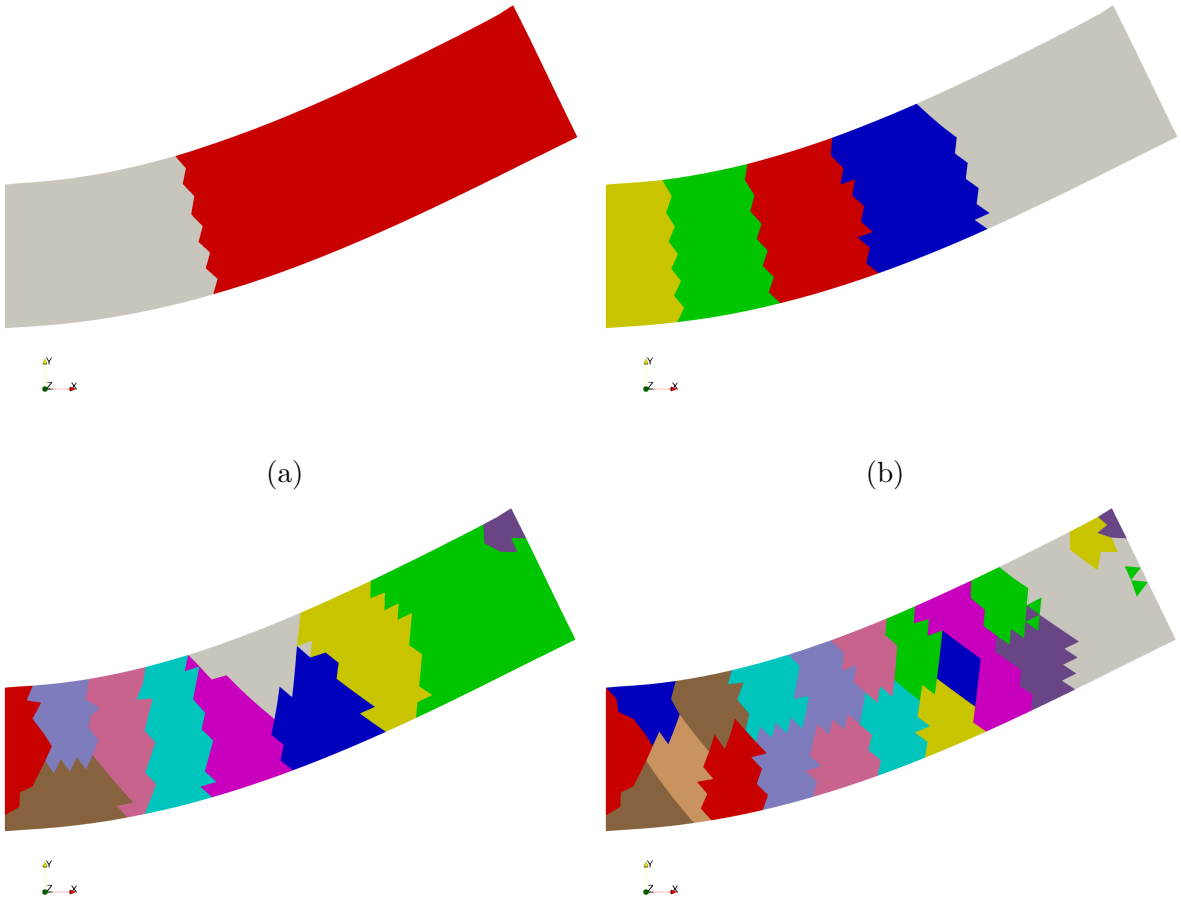


Figure 4: Final cluster assignments for the hyperelastic beam using: (a) two clusters, (b) five clusters, (c) 11 clusters, (d) 23 clusters.

partitioned coarse scale solver: 50-92% of the total computation time. It should be noted that this operation could be omitted for this setup because there are no internal variables. With more clusters, computation time is dominated by the partitioned coarse scale solver and, except for the case with 184 clusters, it increases linearly with the number of clusters. Except for the simulation with 184 clusters, there were no convergence issues and no restarts, including for the reference FE^2 simulation.

The reference FE^2 simulation required 13 470 s to finish. The simulation with 23 clusters, which leads to an error below ten percent for both displacement and stress fields, is interesting since it reduces the computation time by a factor of 16.7. This is similar to the speed-up ratio found in Ref. (Chaouch and Yvonnet, 2024a). Even for the simulation with 184 clusters, where a convergence issue led to a fall back and eight (among 82) coarse scale solver iterations without clustering, there is still a computation time reduction factor of 1.66.

Computation time spent clustering (including pre-computing distances), in addition, is negligible as it represented less than one percent of the total computation time for all simulations. This is explained by the fact that the last assignment is used as initial guess for each run of the k-medoids clustering algorithm, and that, as analyzed above, assignments and medoids do not change much during these simulations with monotonic loading.

4.1.4 Importance of cross contributions

For all results reported in Tab. 1, second-order derivative tensors and cross contributions were neither computed nor assembled. Since no convergence issue was encountered for most of these simulations, it can be concluded that assembling cross contributions is not a necessary condition for convergence of the coarse scale solver.

In order to assess the importance of these contributions, two analysis steps are proposed. First, a focus is made on the first nonlinear iteration of the simulation with a single cluster. A reference Jacobian matrix $\boldsymbol{J}^{ref}$ is computed using numerical differentiation over the whole nonlinear system, which means a formula similar to Eq. (A.1) is used but the perturbation ϵ is applied over each node-wise displacement component and the whole node-wise force vector is computed every time. The computation time for this

procedure is nearly prohibitive but it ensures that influences of displacements across the mesh are captured (*i.e.*, a nodal displacement affecting the force of a node with which it does not share an element). Jacobian matrices $\boldsymbol{J}$ are then computed and assembled with Eq. (A.1) by enabling or disabling cross contributions, and compared to the reference one $\boldsymbol{J}^{ref}$ through the error measure:

$$\text{Error}(\boldsymbol{J}) = \frac{||\boldsymbol{J} - \boldsymbol{J}^{ref}||_2}{||\boldsymbol{J}^{ref}||_2}, \tag{23}$$

An error of 1.04×10^{-1} is obtained without cross contribution, and 5.64×10^{-5} with cross contributions. This clearly shows that the Jacobian matrix is not complete when cross contributions are neglected.

The second analysis step is to evaluate the influence of cross contributions on the number of iterations of the partitioned coarse scale solver, and the occurrence of convergence issues. The simulations with 92 and 184 clusters are reran with cross contributions. The number of partitioned coarse scale solver iterations with 92 clusters are reduced from 57 to 41 by assembling cross contributions. The result is more impressive with 184 clusters, as the reduction is from 82 to 43. More importantly, no convergence issue and hence no fall back occurred in the simulation with cross contributions, as opposed to the one without.

One could think avoiding fall backs to the full non-partitioned coarse scale solver would be quite beneficial in terms of computation time. Computation of second-order derivatives is, nevertheless, very expensive, as the simulation with 184 clusters and cross contributions took seven times longer to complete. This is even worse for the simulation with 92 clusters, as it took 11 times longer.

As a conclusion, although they are not always necessary to reach convergence, cross contributions could be helpful to avoid some issues. The computational cost associated to second-order derivatives, is however, too prohibitive.

4.2 Elasto-plastic bracket

The second setup is also inspired from Ref. (Chaouch and Yvonnet, 2024a). Under the assumption of plane strains, the bracket shown in Fig. 5(a) is subjected to a non-

proportional loading path where holes A and B are fixed, while hole C is fixed in the horizontal direction and subjected to the time-dependent displacement shown in Fig. 5(c) in the vertical direction. The fine scale domain is the same as in the previous setup, but the material model is replaced by the small strain elasto-plastic model presented in Subsec. 2.4, and the inclusion is replaced by a void. The latter is modeled by using as properties a Young's modulus of 2 kPa, a Poisson's coefficient of 0.3 and an initial yield stress large enough to prevent plasticity. Matrix properties are a Young's modulus of 2 GPa, a Poisson's coefficient of 0.3, an initial yield stress of 24 MPa, and a kinematic hardening modulus of 80 MPa. Isotropic hardening is neglected.

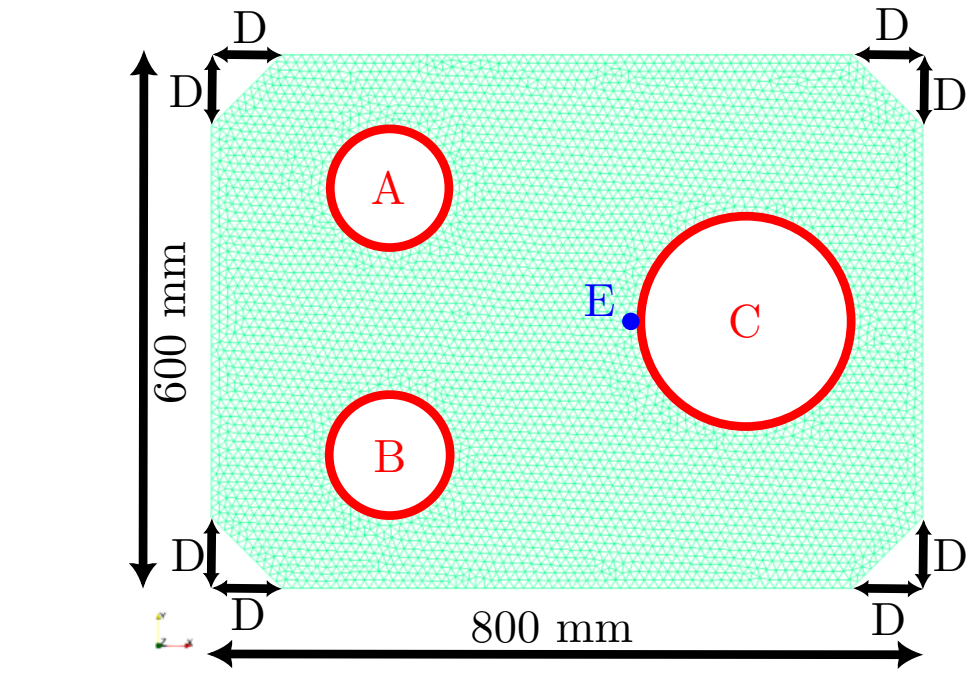
Loading is applied over 61 increments with a time step of 1 s, except for the first increment which uses a very small step to get an elastic response and initialize the NICE scheme. The mesh is composed of 8130 linear triangular elements for the coarse scale, and 916 linear triangular elements for the fine scale. This second setup involves an internal variable to track matrix plasticity at the fine scale, and a change in the loading direction at the coarse scale.

4.2.1 Convergence with an increasing number of clusters

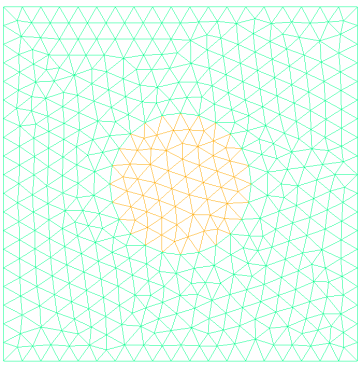
Equivalent stress fields at the loading minimum are presented in Fig. 6. They are all qualitatively similar, with the only difference being some small noise when using less clusters.

To compare results to Ref. (Chaouch and Yvonnet, 2024a), the local shear stress $\boldsymbol{\sigma}_{12}^{M}$ history is analyzed for the point E defined in Fig. 5(a). Shear stress-strain curves at this point are drawn in Fig. 7. The original method proposed in Ref. (Chaouch and Yvonnet, 2024a) could compute converged results with around 100 clusters. Clearly, the k-medoids-based PFE^2 method requires more clusters. The main issue is path direction change, which occurs twice in these simulations, and leads to severe inaccuracies with 200, 100 and 50 clusters. This issue was also observed in Ref. (Chaouch and Yvonnet, 2024a) when using a small number of clusters. As shown in Fig. 7, with less than 50 clusters, results seem to improve.

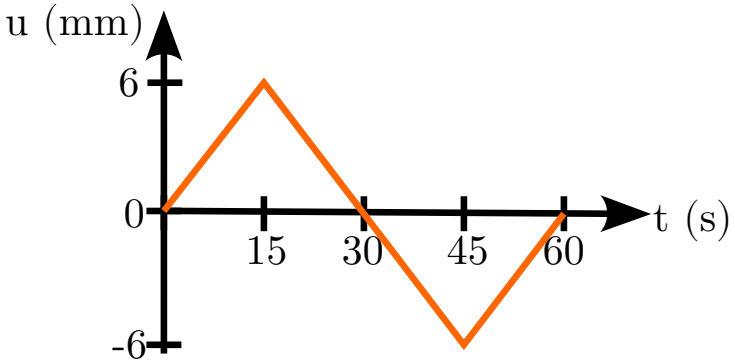
Errors for simulations with various numbers of clusters are reported in Tab. 2. Convergence rates are low for both displacement and stress errors and it is difficult to get



(a)



(b)



(c)

Figure 5: Geometries, meshes and loading for elasto-plastic bracket simulations: (a) coarse scale, (b) fine scale with colors distinguishing matrix and void, (c) vertical displacement applied at C.

Radii are 70 mm for holes A and B, and 120 mm for hole C. Hole centers for A and B are positioned 200 mm away from the left side and 150 mm away from the top and bottom side, respectively. Hole C's center is positioned 200 mm away from the right side, and $D = 80$ mm.

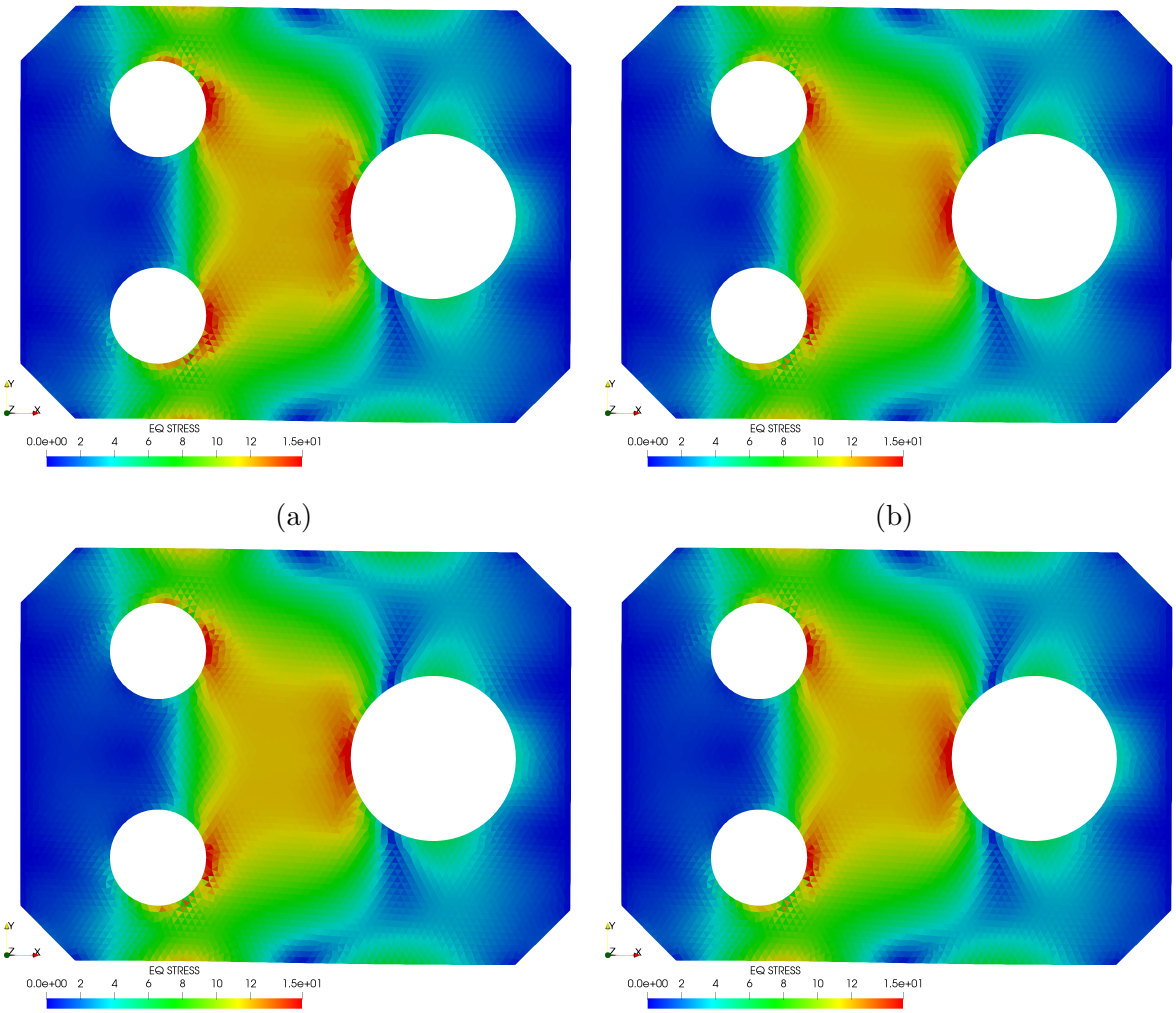
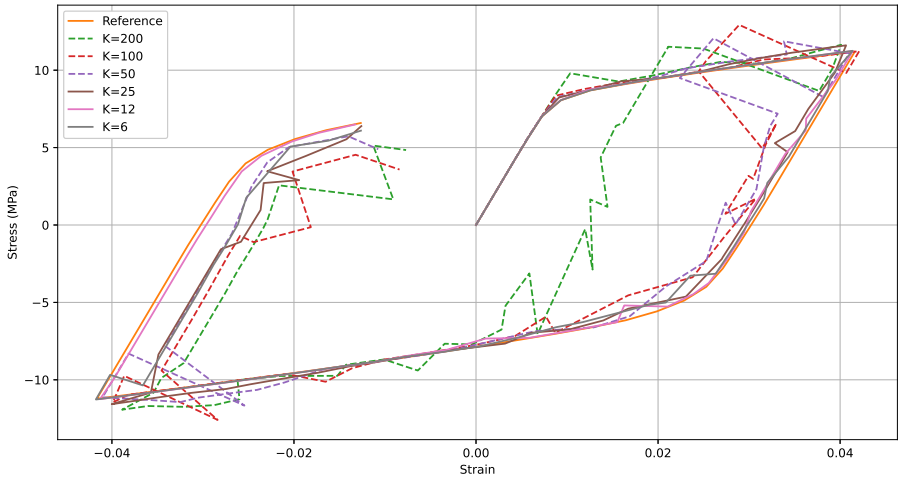


Figure 6: Equivalent stress (in MPa) at $t = 45\,\mathrm{s}$ for the elasto-plastic bracket using: (a) 81 clusters, (b) 162 clusters, (c) 325 clusters, (d), the FE^2 scheme.



(a)

Figure 7: Stress-strain curves at point E defined in Fig. 5(a): components $\boldsymbol{\varepsilon}_{12}^{M}$ and $\boldsymbol{\sigma}_{12}^{M}$.

accurate results for both fields at the final state. This is mainly due to the fact that both displacement and stress magnitudes are small at the final state. At $t = 45\,\mathrm{s}$, however, errors are already below ten percent for both fields using 40 clusters. This demonstrate that even though the proposed k-medoids-based PFE^2 method might not be appropriate to predict residual stresses, it could still be quite accurate in many situations.

4.2.2 Computation time

The reference FE^2 simulation required 437 787 s to finish. The best compromise in terms of error and computation time (Tab. 2) is obtained with 325 clusters, which leads to a reduction factor of 5.81. This is far below the theoretical factor of 25. This poor performance of the k-medoids-based PFE^2 method is due to convergence issues. Although the k-medoids-based PFE^2 simulation with 325 clusters encountered no restart nor fall back to the full FE^2 scheme, it required 771 coarse scale solver iterations, against 299 for the reference FE^2 simulation, which can explain its under-performance.

Computation times in Tab. 2, in addition, might seem quite random. Since no simulation led to a restart, this randomness is due to convergence issues leading to fall backs. There were three of them for the simulation with 40 clusters, seven with 81, eight with 162 and

# clusters	Error($\boldsymbol{u}^M$) $t = 45\,\mathrm{s}$	Error($\boldsymbol{\sigma}^M$) $t = 45\,\mathrm{s}$	Error($\boldsymbol{u}^M$) $t = 60\,\mathrm{s}$	Error($\boldsymbol{\sigma}^M$) $t = 60\,\mathrm{s}$	Speed-up
40	6.83×10^{-3}	6.10×10^{-2}	6.81×10^{-1}	5.55×10^{-1}	7.48
81	6.28×10^{-3}	6.75×10^{-2}	5.48×10^{-1}	5.39×10^{-1}	4.07
162	4.51×10^{-4}	5.12×10^{-3}	2.78×10^{-1}	4.79×10^{-1}	3.48
325	2.07×10^{-3}	2.42×10^{-2}	2.41×10^{-1}	3.29×10^{-1}	5.81
677	1.00×10^{-3}	1.58×10^{-2}	9.72×10^{-2}	2.21×10^{-1}	3.93
1355	6.58×10^{-4}	1.33×10^{-2}	1.63×10^{-1}	3.51×10^{-1}	2.32

Table 2: Errors and speed-up ratios for elasto-plastic bracket simulations using an increasing number of clusters.

one with 1355.

The comparison with Ref. (Chaouch and Yvonnet, 2024a) is difficult. The authors mentioned computation times between 115 200 s using 80 clusters and 145 980 s using 120 clusters, which could be multiplied by eight given that the authors used parallel computing. They also mentioned a speed-up ratio of 19, but they did not actually conduct the reference FE^2 simulation and rather estimated its computation time to a month. As mentioned previously, the FE^2 simulation actually took around five days with the present implementation, and the simulation with 81 clusters took 107 358 s without parallel computing. These differences could be due to the use of different workstations and, more likely, the use of Matlab in Ref. (Chaouch and Yvonnet, 2024a) against C in the present work (Shakoor, 2022).

Another important aspect is the time spent updating internal variables by solving all fine scale problems after convergence of the partitioned coarse scale solver. With 40 clusters, the computation time spent on this task is 32%, and with 325 clusters it is 25%. For all other simulations, it is 15%. As could be expected, this is highly dependent on the number of clusters and also coarse scale solver iterations. It is important, nevertheless, to observe that it remains small as compared to the total computation time.

4.2.3 Importance of cross contributions

Due to convergence issues mentioned above, cross contributions are assembled, with the hope that the computation time spent computing second-order derivative tensors will pay off. This is conducted for the setups with 40 and 81 clusters.

Surprisingly, the number of fall backs increases with cross contributions from three to nine with 40 clusters, and from seven to eight with 81. Although fall backs are related to a stagnation of the residual when using clustering, even with frozen cluster assignments, it seems that implementing cross contributions does not always help decreasing this residual. This is confirmed by analyzing the total number of coarse scale solver iterations, which is increased from 818 to 1053 with 40 clusters and from 995 to 1071 with 81. This is detrimental to computation time, which is already affected by the computation of second-order derivative tensors, as it is multiplied by around four in both cases.

All in all, convergence issues of the coarse scale solver do not seem to be systematically caused by the absence of cross contributions. Given that second-order derivative tensors are very expensive to compute, it can be concluded that their implementation is not advisable.

4.3 Double-notched test

The third setup is the well-known double-notched test (Seabra et al., 2013). Under the assumption of plane strains, the specimen shown in Fig. 8(a) is fixed on its two bottom and right boundaries B, and subjected to the non-proportional loading path shown in Fig. 8(c) on its two top and left boundaries A. The fine scale domain is more complex than previous ones as it contains seven inclusions. The small strain elasto-plastic model presented in Subsec. 2.4 is used. Inclusion properties are a Young's modulus of 210 GPa, a Poisson's coefficient of 0.3 and an initial yield stress large enough to prevent plasticity. Matrix properties are a Young's modulus of 2 GPa, a Poisson's coefficient of 0.3, an initial yield stress of 24 MPa, and an isotropic hardening modulus of 80 MPa. Kinematic hardening is neglected.

Loading is applied over 21 increments with a time step of 0.1 s, except for the first increment which uses a very small step to get an elastic response and initialize the NICE

scheme. The mesh is composed of 849 linear triangular elements for the coarse scale, and 2081 linear triangular elements for the fine scale.

4.3.1 Analysis of medoid change

A first simulation is conducted with six clusters over the whole specimen. Cluster assignments and medoids are shown in Fig. 9. In the first stage shown in Fig. 9(a,b), clusters are concentrated close to holes and in the shear band between the holes. Medoids also move towards the holes and the shear band between Fig. 9(a) and Fig. 9(b). Shear stress values are indeed significant in these regions, as illustrated in Fig. 10(a,b), which is associated to matrix plastic flow at the fine scale.

In the second stage, the stress field changes, as illustrated in Fig. 10(c,d), which also affects clustering assignments, as shown in Fig. 9(c,d). Medoids, as a result, move away from the shear band.

This double-notched test problem is clearly challenging, mainly due to matrix plastic flow at the fine scale and loading path direction change. The k-medoids clustering algorithm changes both cluster assignments and medoids during the simulation to try to represent the full complexity of the problem.

4.3.2 Convergence and computation time

Errors and computation times are analyzed in two steps. First, only the first loading stage is considered. Even though there is plasticity, loading is proportional because the loading direction remains constant. The reference FE^2 simulation took 35 245 s. With a speed-up ratio of 5.34, the k-medoids-based PFE^2 method computes results with an accuracy below ten percent for both displacements and stresses using six clusters, as shown in Tab. 3. There is a fall back to the full FE^2 scheme in this simulation, which, by comparison to the simulation with 13 clusters, appears detrimental to computation time but beneficial to accuracy. Errors close to ten percent are also obtained with 26 clusters and no fall backs, which leads to a computation time divided by 10.3.

Second, both loading stages are considered. Loading direction change is challenging especially regarding the conservation of internal variables. The reference FE^2 simulation took 74 426 s. As shown in Tab. 4, a good accuracy is achieved for all simulations except

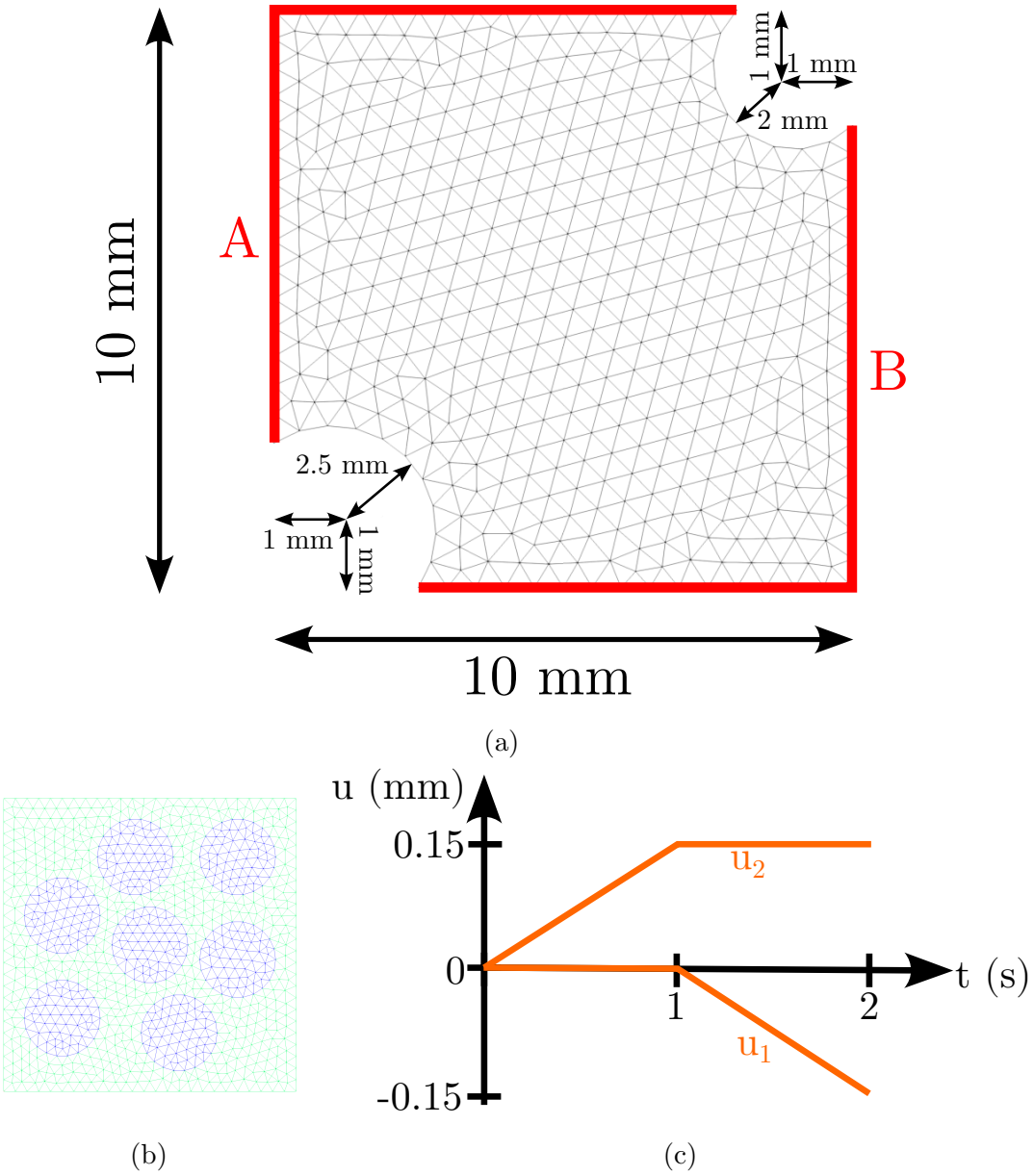
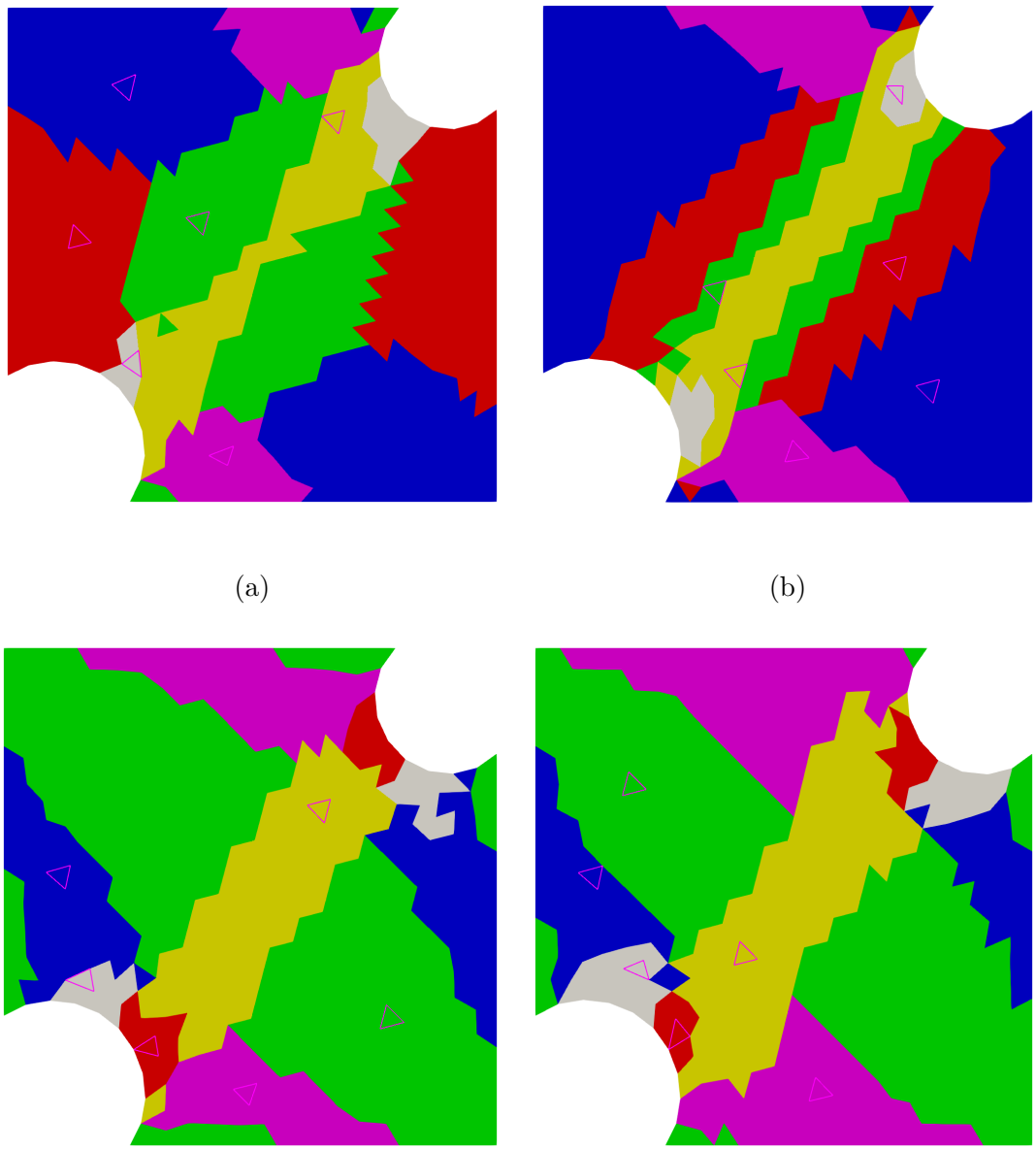


Figure 8: Geometries, meshes and loading for double-notched test simulations: (a) coarse scale, (b) fine scale (square size of 1 mm) with colors distinguishing matrix and inclusions (radius of 0.13 mm), (c) horizontal and vertical displacements applied at A.



(a) (b)

(c) (d)

Figure 9: Cluster assignments (colors) and medoids (highlighted elements) using six clusters for the double-notched test at: (a) $t = 0.5\,\mathrm{s}$, (b) $t = 0.9\,\mathrm{s}$, (c) $t = 1.5\,\mathrm{s}$, (d) $t = 2.0\,\mathrm{s}$.

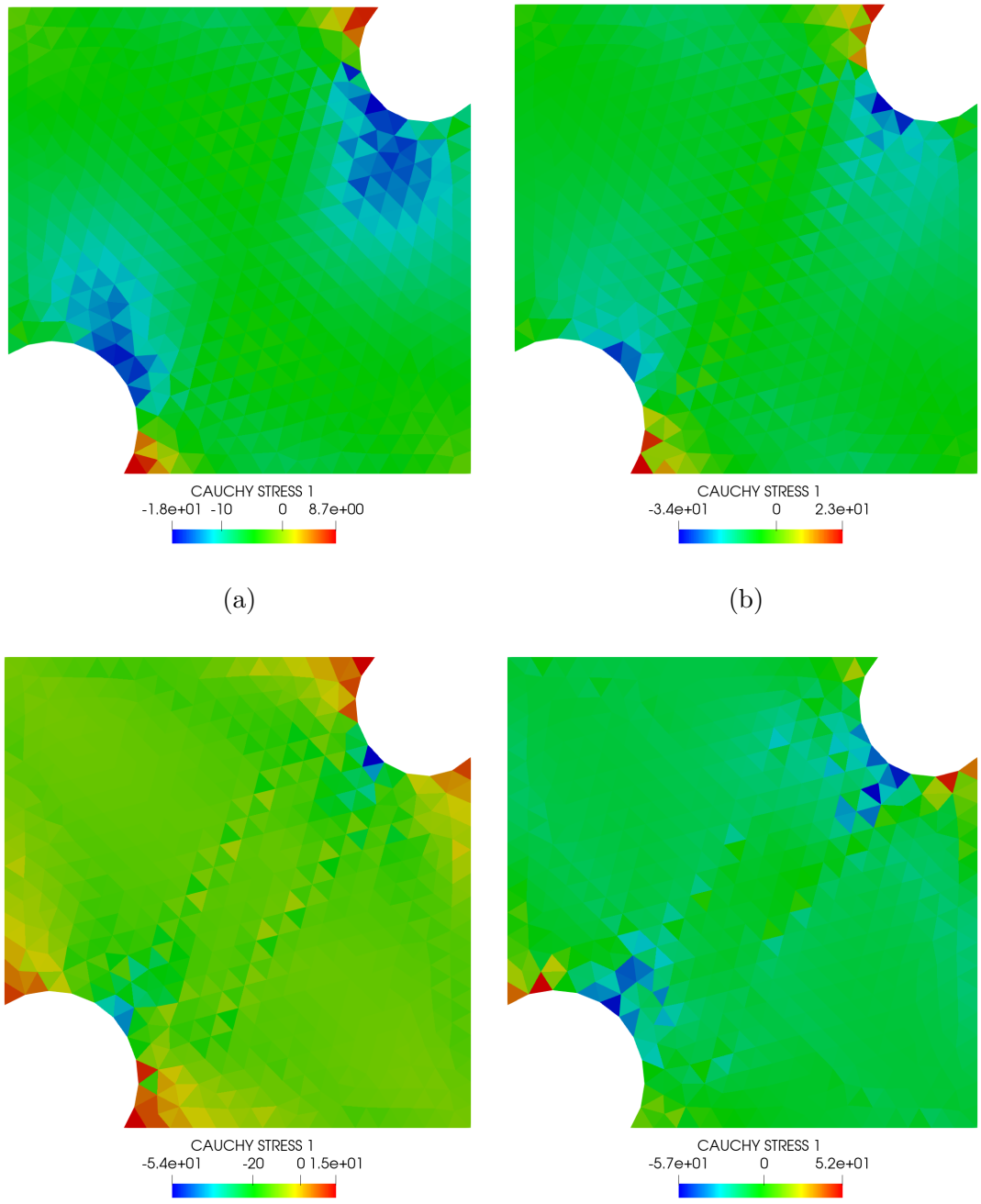


Figure 10: Shear stress component $\boldsymbol{\sigma}_{xy}^{M}$ using six clusters for the double-notched test at: (a) $t = 0.5\,\mathrm{s}$, (b) $t = 0.9\,\mathrm{s}$, (c) $t = 1.5\,\mathrm{s}$, (d) $t = 2.0\,\mathrm{s}$.

# clusters	Error($\boldsymbol{u}^M$)	Error($\boldsymbol{\sigma}^M$)	Speed-up	# fall backs
6	1.40×10^{-2}	4.51×10^{-2}	5.34	1
13	2.64×10^{-2}	1.37×10^{-1}	13.3	0
26	1.64×10^{-2}	1.08×10^{-1}	10.3	0
53	4.45×10^{-3}	3.03×10^{-2}	3.50	1

Table 3: Errors and speed-up ratios for double-notched test simulations up to $t = 1\,\mathrm{s}$ using an increasing number of clusters.

# clusters	Error($\boldsymbol{u}^M$)	Error($\boldsymbol{\sigma}^M$)	Speed-up	# fall backs
6	7.98×10^{-2}	3.04×10^{-1}	2.31	5
13	3.89×10^{-2}	9.70×10^{-2}	2.10	6
26	6.89×10^{-3}	2.00×10^{-2}	1.73	7
53	3.83×10^{-3}	1.32×10^{-2}	1.26	9

Table 4: Errors and speed-up ratios for double-notched test simulations up to $t = 2\,\mathrm{s}$ using an increasing number of clusters.

the one with only six clusters. Unfortunately, this is mainly attributed to a significant number of fall backs to the full FE^2 scheme after loading direction change. The speed-up ratio is hence of 2.10 when using 13 clusters.

Regarding the time spent solving all fine scale problems after convergence of the partitioned coarse scale solver, it varies significantly. It is maximum for the simulations with 13 and 26 clusters in the first loading stage, where it reaches respectively 48% and 35% of the total computation time. It is minimum for the simulation with 53 clusters and both stages, where it represents only 3% of the total computation time, mainly due to the number of fall backs. This shows once again that the cost of the strategy consisting in solving all fine scale problems for updating internal variables is acceptable.

4.3.3 Path-dependence

An important aspect of plasticity is path-dependence. In the PFE^2 method, this is dealt with by carefully updating internal variables at the fine scale. The latter play a key role

for fine scale problems associated to newly-made medoids when clusters and/or medoids change. To evaluate how the PFE^2 method can capture path-dependence, simulations are conducted for the same double-notched test as in Fig. 8 but with inverted loading stages. The final total displacement at A is therefore identical, but the path is different. Final shear stress fields for the two loading paths are presented in Fig. 11(c,d). Comparison with reference FE^2 results in Fig. 11(a,b) reveals that with only 13 clusters, the change of the shear band's angle is predicted by the k-medoids-based PFE^2 method.

Cluster assignments at the end of each stage using six clusters are shown in Fig. 12(a,b). Clustering itself seems to be path-dependent. Comparing Fig. 12(a,b) and Fig. 9(b,d), clearly, there is a change in the orientation of the elongated cluster capturing the shear band.

The FE^2 simulation with inverted loading path finished in 85 370 s. The PFE^2 simulations with inverted loading path encountered less convergence issues as there were three fall backs with six clusters, four with 13, three with 26 and seven with 53. Speed-up ratios were consequently improved, respectively, to 4.00, 3.06, 3.19 and 1.54.

5 Conclusions

An original k-medoids-based PFE^2 approach has been introduced in this paper. As the computation time of FE^2 schemes typically scales with the number of fine scale problems to solve, this approach directly tackles the issue by effectively reducing the number of fine scale solves. At each coarse scale nonlinear iteration, coarse scale integration points are partitioned *a priori* based on the current coarse scale displacement gradient and fine scale internal variables using the k-medoids clustering algorithm. Fine scale solves are only triggered for cluster medoids, and obtained stresses and tangent modulii are extended to the remaining non-medoid coarse scale integration points.

As compared to previous works, the proposed method has the advantage of solving fine scale problems associated to actual coarse scale integration points (*i.e.*, the medoids) instead of creating virtual ones. Fine scale displacement fields and internal variables, moreover, are systematically updated for all integration points.

Convergence issues are dealt with by combining cluster freezing, restarts with time step

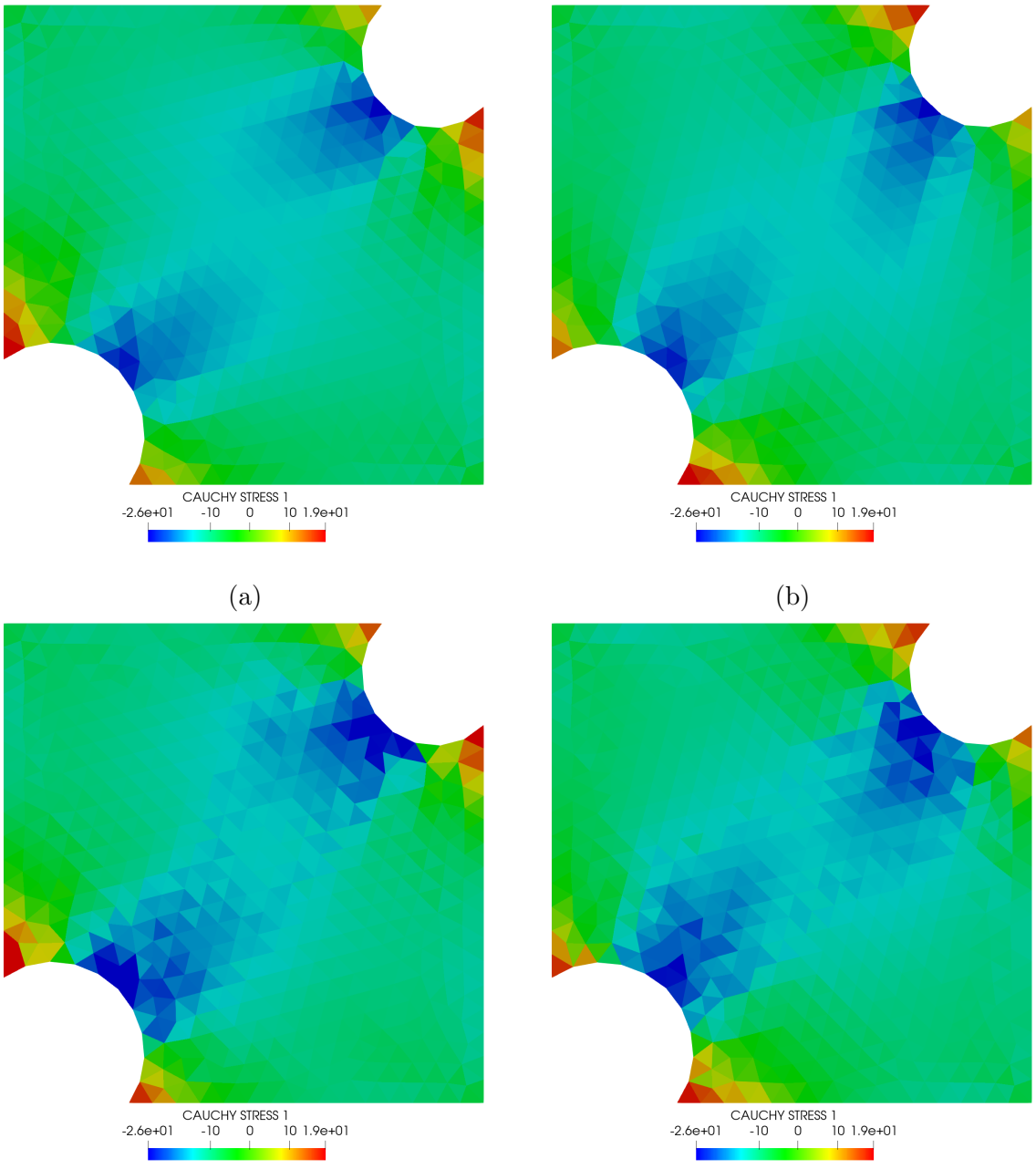
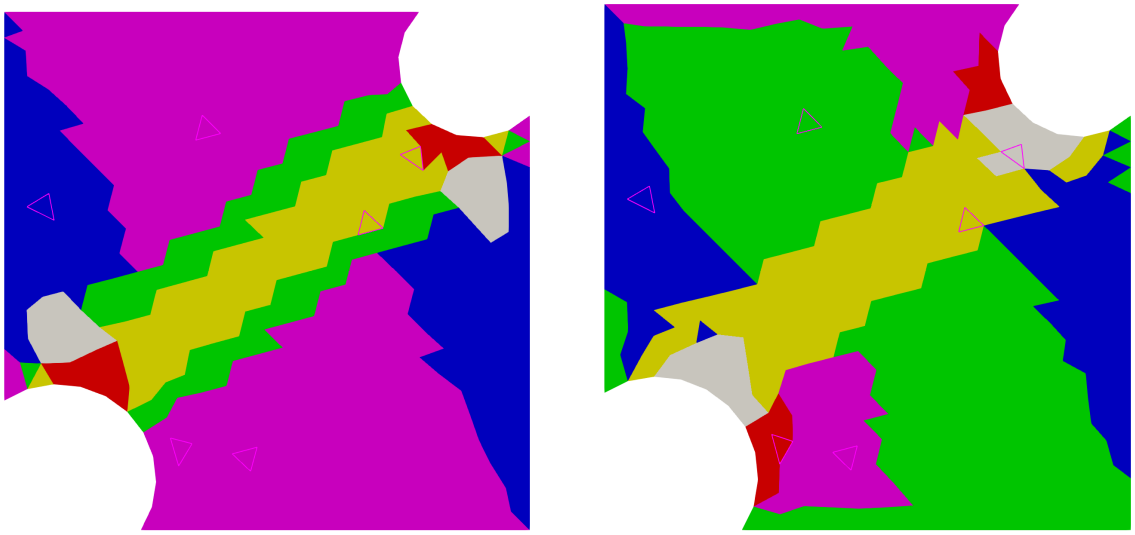


Figure 11: Shear stress component $\boldsymbol{\sigma}_{xy}^{M}$ for the double-notched test using: (a) reference loading path and FE^2 scheme, (b) inverted loading path and FE^2 scheme, (c) reference loading path and PFE^2 method with 13 clusters, (d) inverted loading path and PFE^2 method with 13 clusters.



(a) (b)

Figure 12: Cluster assignments (colors) and medoids (highlighted elements) using six clusters for the double-notched test with inverted loading path at: (a) $t = 0.9\,\mathrm{s}$, (b) $t = 2.0\,\mathrm{s}$.

reduction, and fall backs to the full FE^2 scheme. Cross contributions involving second-order derivatives and pairing each coarse scale integration point with its cluster's medoid have also been introduced in this paper. Their positive effect on convergence and the number of coarse scale nonlinear iterations, however, was not deemed sufficient to justify their employment, given the number of supplementary fine scale solves that they require. Numerical results with nonlinear material behavior such as hyperelasticity and elasto-plasticity with either kinematic or isotropic hardening show that the new method is accurate and is a promising candidate for reducing the computational cost of FE^2 simulations. Its implementation is also less intrusive and simplified as compared to previous works. Convergence issues, however, deteriorate significantly the performance of this method and should be the object of more research.

Appendix

Computational homogenization with an FE^2 approach consists in discretizing and solving the problems in Eqs. (1) and (3) using the FE method. The first step is to discretize the coarse scale domain Ω_0^M with an appropriate FE mesh. Then, at each integration point of this coarse scale mesh, a new discrete fine scale model should be created with its own and independent FE mesh. This is the reason why the computational complexity of an FE^2 scheme is of $\mathcal{O}(N_q^M N_{DOF}^m)$, where N_q^M is the number of coarse scale integration points, and N_{DOF}^m is the number of fine scale degrees of freedom. The whole point of the PFE^2 method is to reduce this complexity to $\mathcal{O}(N_c^M N_{DOF}^m)$, where N_c^M is the number of clusters with, ideally, $N_c^M \ll N_q^M$.

In addition to the meshes themselves and the associated coordinate and connectivity data, the FE^2 scheme requires to store an unknown displacement vector for each node of the coarse scale mesh, and for each node of each fine scale mesh, as well as an internal variables vector for each integration point of each fine scale mesh. This is the strict minimum of information needed in theory, but in practice the numerical implementation usually requires more data such as basis functions, mappings, strains, stresses, etc.

A.1 Coarse scale solver

In this work, although all considered material models are rate-independent, loading is applied over several time increments $t^n = n\Delta t \in [0, T]$, with Δt the time step. At each time increment, the coarse scale solver consists in:

- Running any pre-processing operation such as the management of internal variables for each fine scale problem.
- Initializing $\boldsymbol{u}^{M,n+1,0} = \boldsymbol{u}^{M,n}$.
- Iterating over $i = 0 \dots N_c$:
  - Interpolating the displacement and its gradient at each integration point of the coarse scale mesh from the current guess $\boldsymbol{u}^{M,n+1,i}$.

  - Solving each fine scale problem as per Eq. (3) to obtain the coarse scale stress tensor $\boldsymbol{P}^{M,n+1,i}$ and the consistent tangent modulus $\dfrac{\partial \boldsymbol{P}^{M,n+1,i}}{\partial \nabla_{\boldsymbol{X}} \boldsymbol{u}^{M}}$.
  - Assembling the linearized coarse scale problem.
  - Testing for convergence and stopping if the criterion is met under some tolerance.
  - Solving the linearized coarse scale problem to obtain the new guess $\boldsymbol{u}^{M,n+1,i+1}$.
- Running any post-processing operation and writing output files.

The convergence criterion checks whether the residual is small enough under some absolute tolerance, here 10^{-6}.

On the one hand, if convergence is not met within the prescribed number of Newton-Raphson iterations (here N_c), the time step Δt is divided by two and then the algorithm is restarted. This also happens if the linearized coarse scale problem could not be solved (ill-conditioned matrix), if the coarse scale residual increased too much (more than thousand times the initial residual), or if a fine scale problem could not be solved. Note that a similar Newton-Raphson algorithm with restarts is used for fine scale problems, due to the same issues, to which is added a possible failure of the return mapping algorithm used for plasticity.

On the other hand, if convergence is met without any restart, then the time step Δt is multiplied by $\dfrac{3}{2}$ up to an upper bound which is the user-defined initial time step Δt_0. Note that this is a pure Newton-Raphson algorithm. It does not, in particular, rely on any line search procedure neither at the coarse scale nor the fine scale. Restarting with time step decrease can be seen as a substitute for line searching.

A.2 Computation of tangent modulii

The main difficulty in solving the fine scale problems in Eq. (3) is the presence of Lagrange multipliers. Details about the assembly and solution of this formulation can be found in Refs. (Shakoor and Park, 2023, 2024). Each fine scale problem, in addition, should be solved several times in order to compute the tangent modulus $\dfrac{\partial \boldsymbol{P}^{M,n+1,i}}{\partial \nabla_{\boldsymbol{X}} \boldsymbol{u}^{M}} =$

$\frac{\partial \boldsymbol{P}^M}{\partial \nabla_{\boldsymbol{X}} \boldsymbol{u}^M}(\nabla_{\boldsymbol{X}} \boldsymbol{u}^{M,n+1,i})$ through numerical differentiation using the forward formula:

$$\frac{\partial \boldsymbol{P}^M_{hj}}{\partial \nabla_{\boldsymbol{X}} \boldsymbol{u}^M_{kl}}(\nabla_{\boldsymbol{X}} \boldsymbol{u}^{M,n+1,i}) \approx \frac{\boldsymbol{P}^M_{hj}(\nabla_{\boldsymbol{X}} \boldsymbol{u}^{M,n+1,i} + \epsilon \mathbf{1}^{kl}) - \boldsymbol{P}^M_{hj}(\nabla_{\boldsymbol{X}} \boldsymbol{u}^{M,n+1,i})}{\epsilon}, \tag{A.1}$$

where $\epsilon = 10^{-4}$ and $\mathbf{1}^{kl} = (\delta_{h,k}\delta_{j,l})_{h,j=1...3}$, with $\delta_{i,j} = \begin{cases} 1, i = j, \\ 0, i \neq j, \end{cases}$ the Kronecker delta.

References

Benaimeche, M. A., Yvonnet, J., Bary, B., and He, Q.-C. (2022). A k-means clustering machine learning-based multiscale method for anelastic heterogeneous structures with internal variables. International Journal for Numerical Methods in Engineering, 123(9):2012–2041.

Blanco, P. J., Sánchez, P. J., de Souza Neto, E. A., and Feijóo, R. A. (2016). Variational Foundations and Generalized Unified Theory of RVE-Based Multiscale Models. Archives of Computational Methods in Engineering, 23(2):191–253.

Chaouch, S. and Yvonnet, J. (2024a). An unsupervised machine learning approach to reduce nonlinear fe2 multiscale calculations using macro clustering. Finite Elements in Analysis and Design, 229:104069.

Chaouch, S. and Yvonnet, J. (2024b). Unsupervised machine learning classification for accelerating FE2 multiscale fracture simulations. Computer Methods in Applied Mechanics and Engineering, 432:117278.

Davis, T. A. (2004). Algorithm 832: UMFPACK V4.3—an unsymmetric-pattern multifrontal method. ACM Transactions on Mathematical Software, 30(2):196–199.

de Hoon, M. J. L., Imoto, S., Nolan, J., and Miyano, S. (2004). Open source clustering software. Bioinformatics, 20(9):1453–1454.

Feyel, F. (1999). Multiscale FE2 elastoviscoplastic analysis of composite structures. Computational Materials Science, 16(1-4):344–354.

Gao, J., Shakoor, M., Domel, G., Merzkirch, M., Zhou, G., Zeng, D., Su, X., and Liu, W. K. (2020). Predictive multiscale modeling for unidirectional carbon fiber reinforced polymers. Composites Science and Technology, 186:107922.

Geers, M., Kouznetsova, V., and Brekelmans, W. (2010). Multi-scale computational homogenization: Trends and challenges. Journal of Computational and Applied Mathematics, 234(7):2175–2182.

Gierden, C., Kochmann, J., Waimann, J., Svendsen, B., and Reese, S. (2022). A Review of FE-FFT-Based Two-Scale Methods for Computational Modeling of Microstructure Evolution and Macroscopic Material Behavior. Archives of Computational Methods in Engineering, 29:4115–4135.

Halilovič, M., Vrh, M., and Štok, B. (2009). NICE—An explicit numerical scheme for efficient integration of nonlinear constitutive equations. Mathematics and Computers in Simulation, 80(2):294–313.

Hashin, Z. and Shtrikman, S. (1963). A variational approach to the theory of the elastic behaviour of multiphase materials. Acta Metallurgica, 11(2):127–140.

Le, B. A., Yvonnet, J., and He, Q.-C. (2015). Computational homogenization of nonlinear elastic materials using neural networks. International Journal for Numerical Methods in Engineering, 104(12):1061–1084.

Matouš, K., Geers, M. G., Kouznetsova, V. G., and Gillman, A. (2017). A review of predictive nonlinear theories for multiscale modeling of heterogeneous materials. Journal of Computational Physics, 330:192–220.

Mori, T. and Tanaka, K. (1973). Average stress in matrix and average elastic energy of materials with misfitting inclusions. Acta Metallurgica, 21(5):571–574.

Seabra, M. R. R., Šuštarič, P., Cesar de Sa, J. M. A., and Rodič, T. (2013). Damage driven crack initiation and propagation in ductile metals using XFEM. Computational Mechanics, 52:161–179.

Shakoor, M. (2021). FEMS – A Mechanics-oriented Finite Element Modeling Software. Computer Physics Communications, 260:107729.

Shakoor, M. (2022). FEMS — Finite Element Modeling Software. https://hal.science/hal-03781711.

Shakoor, M. and Park, C. H. (2023). Computational homogenization of unsteady flows with obstacles. International Journal for Numerical Methods in Fluids, 95(4):499–527.

Shakoor, M. and Park, C. H. (2024). A pure Stokes approach for coupling fluid flow with porous media flow. Finite Elements in Analysis and Design, 231:104106.

Yvonnet, J. and He, Q.-C. (2007). The reduced model multiscale method (R3M) for the non-linear homogenization of hyperelastic media at finite strains. Journal of Computational Physics, 223(1):341–368.

Funding

This work was supported by the French Agence Nationale de la Recherche (ANR) through the MISSA projet (ANR-22-CE46-0002).

Competing Interests

The author has no relevant financial or non-financial interests to disclose.