Preconditioning strategies for vectorial finite element linear systems arising from phase-field models for fracture mechanics

M. A. Badri, G. Rastiello, E. Foerster

Preconditioning strategies for vectorial finite element linear systems arising from phase-field models for fracture mechanics

M. A. Badri[a,*], G. Rastiello[a,*], E. Foerster[a]

[a] *DES-Service d'études mécaniques et thermiques (SEMT), CEA, Université Paris-Saclay, 91191 Gif-sur-Yvette, France*

Abstract

Phase-field models are frequently adopted to simulate fracture mechanics problems in the context of the finite element method. To depict fracture, this method involves solving a coupled set of Helmholtz-like damage-field equation and augmented linear momentum balance equation. Solutions to these coupled equations are then used as descriptions of crack propagation phenomena within solids. However, this method imposes a constrain of using extremely fine meshing for properly predicting cracks. For practical problems of interest, this very often leads to linear systems with large sizes that have to be repetitively assembled and solved. As such, iterative solution procedures such as the Krylov subspace based methods for solving these large linear systems within the framework of serial/parallel computing environments become mandatory to obtain results in a feasible time. In this work, the vectorial finite discretization for a hybrid phase-field formulation — a monolithic solving scheme — is presented. The underlying nonlinearity present in the coupled set of equations of the the hybrid phase-field model is dealt through Picard iteration that helps to preserve the symmetry of the linearized system to solve. Due to the symmetric positive definite nature of the finite element linear systems obtained for this problem, the conjugate gradient method makes a standard choice of iterative solution algorithm. Within this article, to improve convergence rates, consequently time to solution, of the conjugate gradient method applied to crack propagation problems, different preconditioning strategies are analyzed, tuned, and discussed. Brittle fracture benchmarks are used to measure the performance of preconditioners which are then applied to massively parallel simulations with millions of unknowns. A series of numerical experiments show that the algebraic multigrid preconditioner is well suited for solving the phase-field model for fracture, being superior to the Jacobi and the block Jacobi preconditioning in all regards: ease of solving the problem, iterations to converge, time to solution, and parallel scaling on more than a thousand processes.

Keywords:
Preconditioning, Algebraic multigrid, Phase-field, Brittle fracture, Vectorial finite element method

1. Introduction

Fracture mechanics problems are often modeled using phase-field models (see e.g., Francfort and Marigo (1998), Bourdin et al. (2000, 2008), Amor et al. (2009), Miehe et al. (2010b)). Mainly due to their strong theoretical backgrounds and robust numerical implementation in the context of the finite element method (FEM), phase-field formulations are, today, regarded among the robust techniques to model fracture mechanics. Particularly when crack initiation, propagation, and branching need to be modeled accurately. Over the past decades, the phase-field formulations have been developed to model diverse problems of fracture

*Corresponding authors: mohd-afeef.badri@cea.fr; giuseppe.rastiello@cea.fr

Nomenclature

Acronyms

DOF	degrees of freedom
FEM	finite element method
AMG	algebraic multigrid
CG	conjugate gradient
IC	incomplete Cholesky
ILU	incomplete LU factorization
MPI	message passing interface
SOR	successive over relaxation

Symbols

$\boldsymbol{n}$	outward unit normal vector
$\boldsymbol{t}$	traction vector
$\boldsymbol{u}$	displacement vector
$\boldsymbol{x}$	Cartesian space coordinates
$\boldsymbol{\varepsilon}$	strain tensor
Ω	domain
Ω^h	mesh
$\partial\Omega$	domain boundary
$\partial\Omega^h$	mesh boundary
ψ	elastic energy
ψ^+	tensile elastic energy
ψ^-	compressive elastic energy
$\boldsymbol{\sigma}$	stress tensor
d	damage variable
G_c	material fracture toughness
h	mesh size
l_0	length scale parameter
N	number of
t	time

Subscripts

D	Dirichlet
e	elements
nz	non zeros
N	Neumann
p	processes
v	vertices

mechanics, e.g., to model brittle fracture (Francfort and Marigo, 1998, Bourdin et al., 2000, 2008, Amor et al., 2009, Miehe et al., 2010a,b, Ambati et al., 2014), ductile fracture (Miehe et al., 2015, Alessi et al., 2015, Ambati et al., 2015), large-deformation fracture (Miehe and Schänzel, 2014, Borden et al., 2018), dynamic fracture (Borden et al., 2012, Hofacker and Miehe, 2012), cohesive fracture (Verhoosel and de Borst, 2013, May et al., 2015), thermomechanical fracture (Kuhn and Müller, 2009), to cite but a few. It is backed by experimental evidence (Ambati et al., 2016, Nguyen et al., 2016, Pham et al., 2017) that these models can reliably predict the phenomena of fracture. A recent review article (Wu et al., 2018) provides a general overview of different phase-field models available for solving fracture mechanics problems.

Within the framework of phase-field method, the linear momentum balance equation is supplemented by a Helmholtz-like diffusion equation (damage-field equation) describing damage evolution. The resulting system of partial differential equations are discretized and solved simultaneously. In other words, to predict fracture, for a problem undergoing load variations, the scalar damage-field unknown $d = d \in [0, 1]$ is solved simultaneously with the vector displacement field $\boldsymbol{u}$ until the internal body forces are in equilibrium with the external applied forces. The resulting field d then serves as an indicator to the state of the solid, i.e., $d = 1$ indicates cracked state and $d = 0$ indicates otherwise. In a finite element context, phase-field formulations lead to regularize damage evolution through introducing an internal length scale such that damage is diffused over a band of elements, which is finite in width. In that sense, phase-field models share some features with other non-local formulations (Pijaudier-Cabot and Bažant, 1987, Bažant and Jirásek, 2002, Moës et al., 2011, Giry et al., 2011, Rastiello et al., 2018), and in particular with gradient-damage formulations (Frémond and Nedjar, 1996, Peerlings et al., 2001). Comparative studies between phase-field and gradient-damage formulations were presented in (de Borst and Verhoosel, 2016, Mandal et al., 2019).

Although the solution procedure seems fairly straightforward, prerequisite of extremely fine meshing for capturing cracks, and large-scale simulations lead to computationally challenging scenarios. For standard numerical methods, like the finite element method (FEM), one thus ends up solving linear systems with millions of unknowns. Further, nonlinear coupling and (pseudo) time-dependent simulations, requires repeated solutions of these linear systems. Such reasons highly motivate the application of preconditioned iterative solution algorithms to solve the arising linear systems in a serial or a parallel computing framework.

The primary aim of the current article is to identify and highlight different preconditioning options to improve convergence, consequently time to solution, for the finite element linear systems that arise in the phase-field modeling strategy for fracture. Besides this, we also propose a monolithic way of approximating the hybrid phase-field model via the vectorial finite element method which provides straightforward means of parallelization for coupled set of equations (Badri et al., 2018).

Literature pertaining use of preconditioners in finite element phase-field modeling of fracture mechanics is scarce. Particularly, till date no study on preconditioning strategies for the hybrid phase-field models for fracture has been presented. Nevertheless, preconditioned staggered FEM solution approach for phase-field model was used by Bilgen et al. (2017) to model conchoidal brittle fracture. The authors investigated the Jacobi and the geometric multigrid preconditioner for the Krylov subspace conjugate gradient (CG) method (Hestenes and Stiefel, 1952), in conclusion the geometric multigrid preconditioner outran the Jacobi preconditioner. Much recently, the Krylov subspace GMRES (Saad and Schultz, 1986) (for non-symmetric linear systems) preconditioned with geometric multigrid was used for solving brittle fracture problems in the framework of monolithic FEM solution approach (Jodlbauer et al., 2019). Since geometric multigrid preconditioning was used in Bilgen et al. (2017), Jodlbauer et al. (2019), the authors ended up using structured meshes for the numerical experiments that appear in these studies. A modified staggered FEM solution approach was proposed by Farrell and Maurini (2017) to solve brittle fracture problems. The authors ended up using the complete LU solver and the Krylov subspace MINRES method (Paige and Saunders, 1975) (for symmetric positive indefinite linear systems) preconditioned with algebraic multigrid to speed up the solver convergence. Heister and Wick (2018) solved pressurized phase-field brittle fracture problems using the Krylov subspace GMRES (for non-symmetric linear systems) preconditioned with a block diagonal preconditioner. For solving these block diagonal matrices authors used algebraic multigrid as a standalone solver.

Within this study, the underlying coupled partial differential equations of the hybrid phase-field model by Ambati et al. (2015) are solved by utilizing the vectorial FEM. This formulation was chosen for the sake of studying the solving techniques for phase-field models, but different formulations could have been used. Within the framework of vectorial FEM, the unknown fields $\boldsymbol{u}$ and d, are coupled together in a single vectorial field $\boldsymbol{w} = [\boldsymbol{u}, d]$ and a single vectorial equation then needs to be solved. Two main approaches — staggered approaches (Bourdin, 2007, Miehe et al., 2010a, Ambati et al., 2014, Bilgen et al., 2017) and monolithic approaches (Gerasimov and De Lorenzis, 2016, Liu et al., 2016, Wick, 2017, Jodlbauer et al., 2019) — have been used in the literature to handle nonlinearity in the governing equations of the phase-field model. In this study, the Picard iteration method is used to handle the nonlinearity between the coupled partial differential equations of the hybrid phase-field model. Alongside the vectorial FEM this leads to a monolithic approach. Consequently, the linear systems obtained are symmetric, sparse, and positive definite in nature. Naturally, for memory considerations, the Krylov subspace-based CG method makes a standard choice of iterative solution algorithm for such linear systems. For improving the performance of the CG method, the most effective preconditioner suggested so far is a geometric multigrid, when solving brittle fracture problems on regular structured grids (Bilgen et al., 2017).

Often for many practical problems of interest, unstructured meshes are preferred over structured ones. The reason is that unstructured meshes allow for better geometric representation of complex or simple geometries. Despite unstructured meshing is simpler to perform compared to a structured one, the recommended geometric multigrid preconditioners are not straightforward to set up in that case. Generating a hierarchy of unstructured meshes which are mandatory for setting up the geometric multigrid preconditioners is not a trivial task. It can be argued that the range of applicability of the geometric multigrid is, therefore, limited (Xu and Zikatanov, 2017). An appealing alternative for problems solved with unstructured meshes is another class of multilevel preconditioners, dubbed as algebraic multigrid (AMG). A review by Stüben (2001) provides an overview of this method. With the AMG methods, the coarser grids and associated transfer operators are generated algebraically by using the assembled coefficients of linear system itself. One of the motives for this study, is to tune, apply, and report the performance of AMG preconditioning with application to phase-field models and compare it to other common one-level preconditioners such as the Jacobi and the Block Jacobi.

The rest of the paper is organized as follows. In section 2, the boundary value problem of phase-field fracture mechanics model is introduced briefly. Section 3 then presents the vectorial finite element discretization procedure for the phase-field model. The iterative solution technique of CG along with different preconditioning options is presented in section 4. Results and discussion of the time performance, memory usage, convergence behavior, and parameter tuning of different preconditioners is presented next in section 5. The article then ends with some conclusions in section 6, which highlights the main results of this article and provides recommendations on the use of preconditioners for phase-field fracture mechanics modeling.

2. Mathematical formulations

In this section, we present the mathematical formulation of the hybrid phase-field model (Ambati et al., 2014) for fracture that is used in this work. Such a model is "hybrid" (Ambati et al., 2014, Wu, 2017, Wu and Nguyen, 2018) in the sense that it combines features of the so-called "isotropic" (Bourdin et al., 2000) and "anisotropic" (Amor et al., 2009, Miehe et al., 2010b, Wu et al., 2018) phase-field models. Here, terms isotropic and anisotropic should not be intended according to their meanings in the Continuum Mechanics theory, but following the nomenclature that is generally used in the phase-field community (Miehe et al., 2010a,b). In particular, the term anisotropic designates models such that special splits of the energy/strain/stress are introduced to represent different dissipative processes occurring in the material under prevailing tensile/compressive states.

2.1. Problem setting

We consider a n-dimensional solid body $\Omega \subset \mathbb{R}^n$ ($n = 1, 2,$ or 3) submitted to quasi-static external loading such that damage occurs. According to the phase-field model, on each point $\boldsymbol{x}(x, y, z) \in \Omega$ and at any pseudo-time $t \in [0, T]$, the state of the system is defined by two fields: the vector-valued displacement field $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, t) : \Omega \times [0, T] \to \mathbb{R}^n$ and the scalar-valued damage field $d = d(\boldsymbol{x}, t) : \Omega \times [0, T] \to [0, 1] \subset \mathbb{R}$. From now on, the dependency on the space and time variables will be omitted for the sake of compactness. Also, we denote $\boldsymbol{u} := [u_1, u_2, u_3]^{\mathsf{T}}$, where u_1, u_2, and u_3 correspond to displacements in x, y, and z directions, respectively.

2.2. Boundary value problem of hybrid phase-field model

Consider an arbitrary open bounded domain Ω with imposed tractions on Neumann boundary $\partial\Omega_{\mathrm{N}} \subset \mathbb{R}^n$, and provided with essential Dirichlet conditions on boundary $\partial\Omega_{\mathrm{D}} \subset \mathbb{R}^n$. The split of the domain boundary $\partial\Omega \subset \mathbb{R}^n$ is such that $\partial\Omega = \overline{\partial\Omega_{\mathrm{D}} \cup \partial\Omega_{\mathrm{N}}}$ and $\emptyset = \partial\Omega_{\mathrm{D}} \cap \partial\Omega_{\mathrm{N}}$, with over-line denoting a closure. Under quasi-static conditions and in the absence of body/volume forces within Ω, the hybrid phase-field model poses the following boundary value problem (BVP):

$$\text{find } \boldsymbol{u} : \Omega \times [0, T] \to \mathbb{R}^n \text{ and } d : \Omega \times [0, T] \to [0, 1] \text{ such that } \forall\, t \in [0, T]$$

$$\operatorname{div}\left[(1-d)^2 \boldsymbol{\sigma}(\boldsymbol{u})\right] = 0 \qquad \forall\, \boldsymbol{x} \in \Omega, \tag{1}$$

$$\left[\frac{G_{\mathrm{c}}}{l_0} + 2\mathcal{H}^+(\boldsymbol{u})\right] d - G_{\mathrm{c}} l_0 \Delta d - 2\mathcal{H}^+(\boldsymbol{u}) = 0 \qquad \forall\, \boldsymbol{x} \in \Omega, \tag{2}$$

given the boundary conditions

$$\boldsymbol{\sigma}(\boldsymbol{u}, d) \cdot \boldsymbol{n} = \overline{\boldsymbol{t}} \qquad \forall\, \boldsymbol{x} \in \partial\Omega_{\mathrm{N}}, \tag{3}$$

$$\boldsymbol{u} = \overline{\boldsymbol{u}} \qquad \forall\, \boldsymbol{x} \in \partial\Omega_{\mathrm{D}}, \tag{4}$$

$$\nabla d \cdot \boldsymbol{n} = 0 \qquad \forall\, \boldsymbol{x} \in \partial\Omega_{\mathrm{N}}, \tag{5}$$

where $\operatorname{div}(\bullet)$ is the divergence operator applied to $(\bullet)$, $\Delta(\bullet) = \operatorname{div}(\nabla(\bullet))$ denotes the Laplace operator, $G_{\mathrm{c}} \in \mathbb{R}^+$ quantifies the material fracture toughness, and $l_0 \in \mathbb{R}^+$ is the length-scale parameter to control

the influence of gradient term. $\boldsymbol{n} = \boldsymbol{n}(\boldsymbol{x}) : \Omega \to \mathbb{R}^n$ is the unit normal vector for surface $\partial\Omega$, $\overline{\boldsymbol{t}} = \overline{\boldsymbol{t}}(\boldsymbol{x}, t) : \Omega \times [0, T] \to \mathbb{R}^n$ denotes imposed tractions, and $\overline{\boldsymbol{u}} = \overline{\boldsymbol{u}}(\boldsymbol{x}, t) : \Omega \times [0, T] \to \mathbb{R}^n$ imposed displacements. $\boldsymbol{\sigma}(\boldsymbol{u}) = \frac{\partial \psi(\boldsymbol{u})}{\partial \boldsymbol{\varepsilon}}$ denotes the elastic/effective Chauchy's stress tensor with ψ being the elastic energy defined by

$$\psi(\boldsymbol{u}) = \frac{1}{2}\lambda \operatorname{tr}(\boldsymbol{\varepsilon}(\boldsymbol{u}))\mathbf{I} + \mu\boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{u}), \tag{6}$$

here $\lambda \in \mathbb{R}^+$ and $\mu \in \mathbb{R}^+$ are the Lamé parameters, $\mathbf{I}$ is the second order identity tensor, $\operatorname{tr}(\bullet)$ denotes the trace operator applied to $(\bullet)$, and $\boldsymbol{\varepsilon}(\boldsymbol{u}) : \Omega \times [0, T] \to \mathbb{R}^{n\times n}$ denotes the second order small strain tensor, i.e., the symmetric part of the gradient of the displacement field $\nabla\boldsymbol{u}$, such that $\boldsymbol{\varepsilon}(\boldsymbol{u}) = \left(\nabla\boldsymbol{u} + \nabla^{\mathsf{T}}\boldsymbol{u}\right)/2$.

To account for the experimental evidence that cracking is mainly controlled by tensile stress/strain states, Ambati et al. (2014) proposed to consider that damage diffusion (in eq. (2)) be driven by the maximum tensile elastic energy history function $\mathcal{H}^+(\boldsymbol{u}) : \Omega \times [0, T] \to \mathbb{R}^+$:

$$\mathcal{H}^+(\boldsymbol{u}) = \max_{\tau\in[0,t]} \psi^+(\boldsymbol{u}), \tag{7}$$

with tensile elastic energy $\psi^+(\boldsymbol{u}) : \Omega \times [0, T] \to \mathbb{R}$ being defined as $\psi^+(\boldsymbol{u}) = (1/2)\lambda \left\langle \operatorname{tr}\left(\boldsymbol{\varepsilon}(\boldsymbol{u})\right)\right\rangle_+^2 + \mu \operatorname{tr}\left(\boldsymbol{\varepsilon}_+^2(\boldsymbol{u})\right)$, where the bracket operator is expanded as $\langle\bullet\rangle_+ = (\bullet + |\bullet|)/2$ and $\boldsymbol{\varepsilon}_+(\boldsymbol{u}) =: \Omega \times [0, T] \to \mathbb{R}^{n\times n}$ denotes the positive part of stain tensor $\boldsymbol{\varepsilon}$. Definition (7) ensures avoiding cracking in compressed regions. Finally, to prevent crack interpenetration the following constrain is applied

$$\forall \boldsymbol{x} : \psi^+(\boldsymbol{u}) < \psi^-(\boldsymbol{u}) \Rightarrow d := 0, \tag{8}$$

with $\psi^-(\boldsymbol{u}) : \Omega \times [0, T] \to \mathbb{R}$ denoting the compressive elastic energy, $\psi^-(\boldsymbol{u}) = (1/2)\lambda \left\langle \operatorname{tr}\left(\boldsymbol{\varepsilon}(\boldsymbol{u})\right)\right\rangle_-^2 + \mu \operatorname{tr}\left(\boldsymbol{\varepsilon}_-^2(\boldsymbol{u})\right)$, where $\langle\bullet\rangle_- = (\bullet - |\bullet|)/2$ and $\boldsymbol{\varepsilon}_-(\boldsymbol{u}) : \Omega \times [0, T] \to \mathbb{R}^{n\times n}$ is now the negative part of strain tensor $\boldsymbol{\varepsilon}$[1].

While the chosen hybrid phase-field model suggests performing elastic energy split (to avoid cracking in compression regions) and makes use of the tensile elastic energy ψ^+ (in eq. (2) via the history function $\mathcal{H}^+$) to control the damage evolution seldom in tensile regions of the domain, there exists other approaches in the literature. Among others, the commonly used ones involve decomposition of elastic energy based on volumetric and deviatoric contributions (Amor et al., 2009) and spectral decomposition (Miehe et al., 2010a). However, following these approaches one needs to deal with (strong) non-linear system of equations for elasticity (eq. (1) becomes nonlinear due to strain tensor decomposition). One of the key advantages of using hybrid phase-field model (and perhaps due to which such model was proposed by Ambati et al. (2014)) is that the momentum balance eq. (1) is retained in its linear form, thereby reducing its computational complexity. The only nonlinearity to be dealt with in hybrid phase-field formulation is the one due to variational inequality.

Additionally, as the hybrid phase-field model avoids energy/strain/stress splitting for eq. (1), the operator $\left[(1-d)^2\right]\partial\psi^+(\boldsymbol{u})/\partial\boldsymbol{\varepsilon}$ does not exist in the momentum balance eq. (1), which is not the case for other phase-field models (Amor et al., 2009, Miehe et al., 2010a). It is well known that this operator is non-convex in both variables $\boldsymbol{u}$ and d and leads to difficulties in designing an efficient solution algorithm (Wick, 2017, Kopaničáková and Krause, 2020). As such, the hybrid phase-field model becomes an interesting option for solving fracture mechanics with ease.

The hybrid phase-field model has been proven to work for a wide variety of fracture mechanics problems (Doan et al., 2016, Jeong et al., 2018, Hirshikesh et al., 2018). While this model ensures accurate crack predictions, it is known to reduce the computational cost (due to avoiding strong nonlinearity) by about one order of magnitude compared with other anisotropic phase-field formulations Ambati et al. (2014).

[1]We recall that, given a second order tensor $\mathbf{A}$, its positive ($\mathbf{A}_+$) and negative ($\mathbf{A}_-$) parts are defined as $\mathbf{A}_\pm = \sum_{i=1}^n \langle A_i\rangle_\pm \boldsymbol{e}_i \otimes \boldsymbol{e}_i$, where $\{A_i\}_{i=1}^n$ and $\{\boldsymbol{e}_i\}_{i=1}^n$ denote the eigenvalues and eigenvectors of $\mathbf{A}$, and symbol $\otimes$ indicates the dyadic product between vectors.

3. Finite element formulations

This section proceeds with the finite element spatial discretization procedure for the BVP discussed in the previous section. A three-dimensional (3D) formulation ($n = 3$) is considered for the sake of illustration, but identical equations hold for other dimensions.

3.1. Discretized variational formulations

On a meshed domain $\Omega^h \in \Omega \subset \mathbb{R}^n$, for eq. (1), the mixed finite element variational formulation in the Lagrangian framework for searching the unknown nodal displacements vector $\boldsymbol{u}^h = [u_1, u_2, u_3]^{\mathsf{T}}$ reads:

$$\begin{aligned} &\text{search } \boldsymbol{u}^h \in \mathbb{V}^h \text{ that satisfies } \forall\, t \in [0,T]: \\ &\int_{\Omega^h} \left[(1-d^h)^2 + \kappa\right] \boldsymbol{\sigma}(\boldsymbol{u}^h) : \boldsymbol{\varepsilon}(\boldsymbol{v}^h)\, \mathrm{dv} = \int_{\partial\Omega^h_{\mathrm{N}}} \bar{\boldsymbol{t}} \cdot \boldsymbol{v}^h\, \mathrm{ds} \quad \forall\, \boldsymbol{v}^h \in \mathbb{V}^h, \end{aligned} \tag{9}$$

where $\kappa \ll 1$ is a model parameter to prevent numerical singularity when $d \to 1$. In this formulation, the notation ":" is used for the double contraction between tensors (i.e., component-wise tensor product) and $\mathbb{V}^h$ is a mixed third order vector-valued finite element functional space to approximate vector test function $\boldsymbol{v}^h$ and vector trial function $\boldsymbol{u}^h$:

$$\mathbb{V}^h = \left\{ \boldsymbol{u}^h \in [H^1(\Omega^h)]^3 \;\; \forall t \in [0,T] \mid \forall \boldsymbol{x} \in \partial\Omega^h_{\mathrm{D}} \;\; \boldsymbol{u}^h = \overline{\boldsymbol{u}} \right\}, \tag{10}$$

with $H^1(\Omega^h)$ denoting a square integrable Sobolev functional space. Similarly, for eq. (2), the standard finite element variational formulation for the unknown damage scalar d^h reads:

$$\begin{aligned} &\text{search } d^h \in V^h \text{ that satisfies } \forall\, t \in [0,T]: \\ &\int_{\Omega^h} \left[\frac{G_{\mathrm{c}}}{l_0} + 2\mathcal{H}^+(\boldsymbol{u}^h)\right] d^h \theta^h\, \mathrm{dv} + \int_{\Omega^h} G_{\mathrm{c}} l_0 \nabla d^h \cdot \nabla \theta^h\, \mathrm{dv} = \int_{\Omega^h} 2\mathcal{H}^+(\boldsymbol{u}^h)\theta^h\, \mathrm{dv} \quad \forall\, \theta^h \in V^h, \end{aligned} \tag{11}$$

where, V^h denotes the scalar finite element functional space to approximate scalar test function θ^h and scalar trial function d^h:

$$V^h = \left\{ d^h \in H^1(\Omega^h) \;\; \forall t \in [0,T] \;\middle|\; d^h \in [0,1] \right\}. \tag{12}$$

3.2. Vectorial FEM for the hybrid phase-field model

Using appropriate basis functions to approximate $\boldsymbol{u}^h$ and d^h, eqs. (9) and (11) then lead to two separate linear systems $\mathbf{A}_{\boldsymbol{u}}\mathbf{x}_{\boldsymbol{u}} = \mathbf{b}_{\boldsymbol{u}}$ and $\mathbf{A}_d\mathbf{x}_d = \mathbf{b}_d$. One can consider solving these linear systems repeatedly in a staggered fashion (one after the another). This way of solving the phase-field model has been used several times in the past, e.g., in Ambati et al. (2014), Bilgen et al. (2017), Molnár and Gravouil (2017), Hirshikesh et al. (2018). With the hybrid phase-field model, the staggered solution approach leads to linear equilibrium (with d^h frozen) and damage diffusion (with $\boldsymbol{u}^h$ frozen) equations to solve. As a consequence, the only iterations needed are those ensuring the coupling between the two equations.

Alternate to this staggered solving method, another elegant is a the monolithic approach. Here, it involves a reformulation of eqs. (9) and (11) in such a way that a single linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ arises and is solved repeatedly to handle nonlinearity in an iterative way. Studies such as Vignollet et al. (2014), Gerasimov and De Lorenzis (2016), Liu et al. (2016), Kopaničáková and Krause (2020) convey that monolithic solution strategy (due to faster convergence) in comparison with the staggered strategy lead to a faster solution time and higher accuracy for a problem. However, it should be pointed out that staggered strategies are more memory efficient than the monolithic ones. As monolithic solvers need to handles a larger matrix system ($\mathbf{A}\mathbf{x} = \mathbf{b}$) as opposed to two relatively smaller matrix systems ($\mathbf{A}_{\boldsymbol{u}}\mathbf{x}_{\boldsymbol{u}} = \mathbf{b}_{\boldsymbol{u}}$ and $\mathbf{A}_d\mathbf{x}_d = \mathbf{b}_d$) handled by a staggered solver.

3.2.1. Discretized variational formulation

Considerable gains in assembly and solving times can be achieved if a monolithic approach for the hybrid phase-field model is constructed by using the vectorial FEM discretization procedure. For that purpose, we fully couple the finite element fields $\boldsymbol{u}^h$ and d^h, and solve these by using vectorial finite elements or in other words fully coupled mixed finite element discretization procedure.

To write the vectorial variational formulation for the hybrid phase-field model, we first exploit the arbitrariness and independency of trial fields $\boldsymbol{v}^h$ and θ^h, and sum up the two variational eqs. (9) and (11) to obtain:

$$\begin{aligned}
&\text{search } (\boldsymbol{u}^h, d^h) \in \mathbb{V}^h \times V^h \text{ that satisfies } \forall\, t \in [0,T] : \\
&\int_{\Omega^h} \left[\left(1 + d^h\right)^2 + \kappa \right] \boldsymbol{\sigma}(\boldsymbol{u}^h) : \boldsymbol{\varepsilon}(\boldsymbol{v}^h)\, \mathrm{dv} + \int_{\Omega^h} \left[\frac{G_\mathrm{c}}{l_0} + 2\mathcal{H}^+(\boldsymbol{u}^h) \right] d^h \theta^h\, \mathrm{dv} \\
&+ \int_{\Omega^h} G_\mathrm{c} l_0 \nabla d^h \cdot \nabla \theta^h\, \mathrm{dv} = \int_{\partial\Omega^h_\mathrm{N}} \bar{\boldsymbol{t}} \cdot \boldsymbol{v}^h\, \mathrm{ds} + \int_{\Omega^h} 2\mathcal{H}^+(\boldsymbol{u}^h)\theta^h\, \mathrm{dv} \quad \forall\, (\boldsymbol{v}^h, \theta^h) \in \mathbb{V}^h \times V^h.
\end{aligned} \tag{13}$$

Then, we introduce a fourth order vectorial trial function $\boldsymbol{w}^h = [w_1^h, w_2^h, w_3^h, w_4^h]^\mathsf{T}$ and a fourth order vectorial test function $\boldsymbol{q}^h = [q_1^h, q_2^h, q_3^h, q_4^h]^\mathsf{T}$, with $\boldsymbol{w}^h \in \mathbb{W}^h$ and $\boldsymbol{q}^h \in \mathbb{W}^h$. Here, $\mathbb{W}^h$ denotes a fourth order vectorial finite element space:

$$\mathbb{W}^h = \left\{ \boldsymbol{w}^h \in [H^1(\Omega^h)]^4 \;\; \forall t \in [0,T] \;\middle|\; \forall \boldsymbol{x} \in \partial\Omega^h_\mathrm{D} \;\; \{w_i^h\}_{i=1}^3 = \bar{\boldsymbol{u}} \;\forall \boldsymbol{x} \in \Omega^h \;\; w_4^h \in [0,1] \right\}. \tag{14}$$

In fact, within the vectorial trial function $\boldsymbol{w}^h$ the first three components $\{w_i^h\}_{i=1}^3$ represent the displacement fields $\{u_i^h\}_{i=1}^3$ and the fourth component w_4^h represents d^h, i.e., the scalar damage field[2]. The vectorial test function $\boldsymbol{q}^h$ follows the same logic.

3.2.2. Nonlinear solving based on Picard iterations

The system of equations resulting from (13) is nonlinear. Different approaches are possible to handle this nonlinearity. A usual choice consists in using a Newton–Raphson iterative algorithm. This implies linearizing the variational formulation with respect to the unknown displacement and damage fields, and then solve for their variations between successive iterations. Linearization process leads, however, to a non-symmetric stiffness-like matrix $\mathbf{A}$, which makes CG method non applicable. Here, to obtain a symmetric formulation and preserve a certain similarity with the staggered solution of the chosen hybrid phase-field formulation, choice was made to use Picard iterations (also commonly known as the method of successive substitution).

Using $\boldsymbol{w}^h$ and $\boldsymbol{q}^h$, at a given t, the vectorial variational formulations for the hybrid phase-field model can be written down in the following canonical form:

$$\begin{aligned}
&\text{given } \boldsymbol{w}^{h,j} \in \mathbb{W}^h \text{ search } \boldsymbol{w}^{h,j+1} \in \mathbb{W}^h \text{ that satisfies :} \\
&\int_{\Omega^h} \left[\left(1 - w_4^{h,j}\right)^2 + \kappa \right] \left[\lambda \nabla(\{w_i^{h,j+1}\}_{i=1}^3) \cdot \nabla(\{q_i^h\}_{i=1}^3) + 2\mu \boldsymbol{\varepsilon}(\{w_i^{h,j+1}\}_{i=1}^3) : \boldsymbol{\varepsilon}(\{q_i^h\}_{i=1}^3) \right] \mathrm{dv} \\
&+ \int_{\Omega^h} \left[\frac{G_\mathrm{c}}{l_0} + 2\mathcal{H}^+(\{w_i^{h,j}\}_{i=1}^3) \right] w_4^{h,j+1} q_4^h\, \mathrm{dv} + \int_{\Omega^h} G_\mathrm{c} l_0 \nabla w_4^{h,j+1} \cdot \nabla q_4^h\, \mathrm{dv} \\
&= \int_{\partial\Omega^h_\mathrm{N}} \bar{\boldsymbol{t}} \cdot [\{q_i\}_{i=1}^3]^\mathsf{T}\, \mathrm{ds} + \int_{\Omega^h} 2\mathcal{H}^+(\{w_i^{h,j}\}_{i=1}^3) q_4^h\, \mathrm{dv} \qquad \forall\, \boldsymbol{q}^h \in \mathbb{W}^h.
\end{aligned} \tag{15}$$

In this equation to deal with the nonlinearity, the superscript indices 'j' and '$j+1$' are introduced (discretize pseudo-time). Associated to a finite element variable, these indices correspond to the solutions at previous and current iterations of the solving process.

[2]For a two-dimensional problem ($n = 2$) the vectorial trial function $\boldsymbol{w}^h$ is of third order, $\boldsymbol{w}^h = [w_1^h, w_2^h, w_3^h]^\mathsf{T}$ where the first two components $\{w_i^h\}_{i=1}^2$ represent the displacement fields $\{u_i^h\}_{i=1}^2$ and the third component w_3^h represents the scalar damage field d^h.

Finally, to perform a phase-field simulation for all $t \in [0, T]$, at a given t linear systems $\mathbf{A}\mathbf{x} = \mathbf{b}$ are assembled and solved consecutively. In this study, piecewise linear polynomial basis $[\mathbb{P}_1, \mathbb{P}_1, \mathbb{P}_1, \mathbb{P}_1]^\mathsf{T}$ are used to approximate the Sobolev spaces $\left[H^1(\Omega^h)\right]^4$ in order to assemble the matrix $\mathbf{A}$ and the vector $\mathbf{b}$ from the left-hand side and right-hand side of eq. (15) respectively. For a piecewise linear polynomial basis, the number of degrees of freedom N_{DOF} (associated to a finite element unknown in mesh Ω^h) is equal to the number of vertices N_v of the mesh ($N_{\text{DOF}} = N_\text{v}$). The solution vector $\mathbf{x}$ contains the finite element fields such that for a Ω^h with N_v vertices:

$$\mathbf{x} = \left[w^h_{1,1}, w^h_{2,1}, w^h_{3,1}, w^h_{4,1}, \ldots, \underbrace{w^h_{1,k}, w^h_{2,k}, w^h_{3,k}, w^h_{4,k}}_{k\text{th node}}, \ldots, w^h_{1,N_\text{v}}, w^h_{2,N_\text{v}}, w^h_{3,N_\text{v}}, w^h_{4,N_\text{v}} \right]^\mathsf{T} \tag{16}$$

where $\{w^h_{i,k}\}_{i=1}^4$ are the values of the four finite element fields $\{w^h_i\}_{i=1}^4$ at the kth node of the mesh Ω^h with $k = 1, 2, \ldots, N_\text{v}$.

In the context of phase-field brittle fracture models, monolithic iterative methods (such as ours) may fail to perform in case of "brutal" crack evolution especially in the post-peak regimes (Gerasimov and De Lorenzis, 2016). Convergence can be attained by employing infinitesimally small continuation steps (Alexander and Yorke, 1978) or by applying more advanced methods such as line-search assisted Newton iterations (Gerasimov and De Lorenzis, 2016), trust region methods (Kopaničáková and Krause, 2020), modified Newtons iterations (Wick, 2017), etc. For the chosen Picard iterations in this study we employ the infinitesimally small continuation steps in our numerical experiments.

3.3. Parallelization

For the vectorial finite element discretization procedure of the hybrid phase-field model, the linear system assembled from eq. (15) is of the form, $\mathbf{A}\mathbf{x} = \mathbf{b}$, with matrix $\mathbf{A} \in \mathbb{R}^{m \times m}$ and vectors $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{b} \in \mathbb{R}^m$, with $m = (n + 1) \times N_\text{v}$.

Simulating real-world three-dimensional problems of fracture mechanics with phase-field models prerequisite fine meshes, consequently leading to a large number of finite element unknowns. As such, assembly and solving of $\mathbf{A}\mathbf{x} = \mathbf{b}$ is associated with a large computational overhead in terms of time and memory usage. To further complexify the situation, these linear systems need to assembled and solved multiple times (thousands) to account for the nonlinear and quasi-static iterations.

To subdue these problems associated with the computational overhead, we have implemented a message passing interface (MPI)-based domain decomposition method of parallelization built on top of our vectorial FEM discretization. This enables the use of distributed-memory framework of computing to assemble and solve $\mathbf{A}\mathbf{x} = \mathbf{b}$ in parallel, and largely reduces the computational expenses. Briefly, domain decomposition applied in this study can be divided into three main steps:

(i) for a numerical simulation to be carried out on N_p number of processes, we start with partitioning of the global mesh Ω^h into N_p number of smaller meshes $\{\Omega^h_i\}_{i=1}^{N_\text{p}}$ (subdomains) with more or less equal number of mesh nodes $N_{\text{v},i} \approx N_\text{v}/N_\text{p}$. The subdomains $\{\Omega^h_i\}_{i=1}^{N_\text{p}}$ are then assigned to N_p number of MPI processes (computational units). These $\{\Omega^h_i\}_{i=1}^{N_\text{p}}$ contain a single layer of ghost-nodes that is used to build MPI communication patterns between neighboring subdomains;

(ii) concurrently, within each MPI process, using eq. (15) with mesh Ω^h_i, we assemble a local matrix $\mathbf{A}_i \in \mathbb{R}^{m_i \times m_i}$ and a local vector $\mathbf{b}_i \in \mathbb{R}^{m_i}$, with $m_i = (n+1) \times N_{\text{v},i}$. As such, we create parallel tables of degrees of freedom with local and global views. Globally, N_p number of matrices $\mathbf{A}_i$ and vectors $\mathbf{b}_i$ correspond to row-wise 1D partitioning (Grama et al., 2003) of the system $\mathbf{A}\mathbf{x} = \mathbf{b}$. For a parallel simulation with large number of processes N_p, since $\{\Omega^h_i\}_{i=1}^{N_\text{p}} \ll \Omega^h$, the local matrices and vectors

($\mathbf{A}_i$,$\mathbf{b}_i$) are of smaller dimensions than the global ($\mathbf{A}$,$\mathbf{b}$). This implies N_p times faster linear system assembly phase. This phase of assembly is 'embarrassingly-parallel', hence requires no communication between the involved MPI processes;

(iii) finally, by using N_p MPI processes, the global problem is solved iteratively by the CG method which uses parallel matrix-vector products with a small MPI communication overhead.

4. Solution algorithm and preconditioning

The matrix $\mathbf{A}$ resulting from the vectorial FEM discretization introduced above is sparse, symmetric, and positive definite. As such, the Krylov subspace-based CG method — developed by Hestenes and Stiefel (1952) — makes a standard choice of iterative solution algorithm for solving the assembled linear system $\mathbf{Ax} = \mathbf{b}$. Although the CG method is well established for solving such kind of linear systems, it is likely to suffer from slow convergence for practical problems of interest with millions of degrees of freedom, which are expected for hybrid phase-field simulations of brittle fracture mechanics. To speed up the convergence of CG, preconditioning is used (Málek and Strakos, 2014).

Preconditioning here refers to transforming the assembled linear system $\mathbf{Ax} = \mathbf{b}$ into an alternate linear system with favorable properties for the CG iterative solution. Generally speaking, preconditioning attempts to improve the spectral properties of $\mathbf{A}$, and speeds up the convergence of CG by reducing the spectral condition number of the problem. Hence, preconditioning transforms the original linear system $\mathbf{Ax} = \mathbf{b}$ into an alternate one with the same solution $\mathbf{x}$, but which is easier to solve.

At the linear system level, preconditioning involves in solving:

$$\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b}, \tag{17}$$

where matrix $\mathbf{M}$ is the preconditioner associated to $\mathbf{A}$. For successful preconditioning, the $\mathrm{cond}_2(\mathbf{M}^{-1}\mathbf{A})$ is expected to be less than the $\mathrm{cond}_2(\mathbf{A})$ and/or the eigenvalues of $\mathbf{M}^{-1}\mathbf{A}$ are clustered around 1. A perfect preconditioner would then be the case where $\mathbf{M}^{-1} = \mathbf{A}^{-1}$: in this case, the associated condition number $\mathrm{cond}_2(\mathbf{M}^{-1}\mathbf{A})$ is equal to one. However such preconditioner is not practical to use. Note that eq. (17), referring to the preconditioned linear system, represents a technique called the left-preconditioning. Besides this it is also possible to set up right-preconditioning:

$$\mathbf{AM}^{-1}\mathbf{y} = \mathbf{b}, \quad \mathbf{x} = \mathbf{M}^{-1}\mathbf{y}, \tag{18}$$

or to set up split-preconditioning:

$$\mathbf{M}_1^{-1}\mathbf{AM}_2^{-1}\mathbf{y} = \mathbf{M}_1^{-1}\mathbf{b}, \quad \mathbf{x} = \mathbf{M}_2^{-1}\mathbf{y}, \tag{19}$$

with preconditioner $\mathbf{M} = \mathbf{M}_1\mathbf{M}_2$. Since the matrices $\mathbf{M}^{-1}\mathbf{A}$, $\mathbf{AM}^{-1}$, and $\mathbf{M}_1^{-1}\mathbf{AM}_2^{-1}$ have the same eigenvalues, therefore, the convergence of the CG method will be same for these different cases of preconditioners, eqs. (17) to (19) (Benzi, 2002). As such, only left-preconditioning, eq. (17), is discussed and applied in this article. Also, for efficiently solving the linear systems of the hybrid phase-field model for fracture on parallel computing environment, the preconditioned CG method is a desirable solution technique since its main components — matrix-vector products, dot products, and norms — readily map to distributed memory implementations.

Broadly, preconditioning (constructing a preconditioner) can be achieved via two different approaches, the problem-specific approach and the purely algebraic approach. For a given problem, the problem-specific approach constructs the preconditioners that are based on approximated physics or method which is easier to solve but gives 'nearly' same solution than the given problem in hand, or preconditioners are constructed based on lower-order discretizations of the considered partial differential equations. On the other hand, the purely algebraic approach of preconditioning involves constructing the preconditioner solely by the

information contained in coefficients of the assembled matrix $\mathbf{A}$. These are often constructed after matrix $\mathbf{A}$ has been assembled. While it would be interesting to find/apply optimal preconditioners for fracture mechanics problems via the problem-specific approach, however such a preconditioner does not exist yet. Although, one should note that the geometric multigrid preconditioner can be considered to be of the problem-specific kind and these have been used alongside the CG method for resolving fracture mechanics problems with phase-field models in the recent past (see e.g., Bilgen et al. (2017), Jodlbauer et al. (2019)). The problem-specific preconditioning is generally very sensitive to the details of the involved problem, and even modest changes in the problem can compromise the effectiveness of the solver. For such reasons, purely algebraic preconditioners are often preferred. Besides this, other advantages of the purely algebraic preconditioners over the problem-specific ones are discussed in Benzi (2002).

In this study, different preconditioners (purely algebraic ones provided in PETSc package Balay et al. (2019)) are studied and tested. In particular, attention is focused on:

(i) *Jacobi preconditioning.* Also known as the diagonal preconditioner, the Jacobi preconditioning is considered to be one of the simplest preconditioning techniques which involves setting $\mathbf{M}$ as the diagonal of $\mathbf{A}$, such that $\mathbf{M} = \text{diag}(A_{11}, \ldots, A_{jj}, \ldots, A_{mm})$, with A_{jj} denoting the diagonal coefficients of $\mathbf{A}$ and $j = 1, 2, \ldots, m$. For the parallel variant of Jacobi preconditioning applied in this study, the distributed $\mathbf{M}_i \in \mathbb{R}^{m_i}$ is applicable to $\mathbf{A}_i$ so that $\mathbf{M}_i = \text{diag}(A_{11}, \ldots, A_{jj}, \ldots, A_{m_i m_i})$, with $i = 1, 2, \ldots, N_\text{p}$, $m = \sum_{i=1}^{N_\text{p}} m_i$, and $j = 1, 2, \ldots, m_i$. This diagonal matrix $\mathbf{M}_i$ is trivial to invert, as it simply involves replacing the coefficients of the matrix with its reciprocal, $\mathbf{M}_i^{-1} = \left[\{1/A_{jj}\}_{j=1}^{m_i}\right]$. As such, a Jacobi preconditioner is computationally cheap and straightforward to construct. As discussed earlier, a reasonable preconditioner $\mathbf{M}$ is such that $\mathbf{M}^{-1} \approx \mathbf{A}^{-1}$ and is also cheap to construct. While the Jacobi preconditioning fulfills the latter condition, it is unlikely for a physics like the fracture mechanics that the $\mathbf{M}^{-1}$ constructed solely from the diagonal of $\mathbf{A}$ is a good approximate of $\mathbf{A}^{-1}$.

(ii) *Block Jacobi preconditioning.* A computationally costlier but more robust preconditioner than the Jacobi preconditioner is the block Jacobi preconditioner. It involves setting $\mathbf{M}_i \in \mathbb{R}^{m_i \times m_i}$ to the diagonal block matrix of $\mathbf{A}_i$. The preconditioning matrix $\mathbf{M}_i$ is defined by $\mathbf{M}_i = \left[\{A_{jk}\}_{j=1,k=1}^{m_i,m_i}\right]$. Such a preconditioner is particularly more effective than the Jacobi preconditioner since the block diagonals succeed in reflecting the nonzero structure of the coefficient matrix $\mathbf{A}$ hence hold considerable information of the underlying physics. As such, $\mathbf{M}^{-1}$ from block Jacobi preconditioning is a good approximate of $\mathbf{A}^{-1}$ in comparison to the Jacobi preconditioning. This statement should hold true for the linear systems assembled with the vectorial FEM discretization of the hybrid phase-field model for fracture. In this paper, to extract $\mathbf{M}_i^{-1}$ from the local diagonal block matrices $\mathbf{M}_i$ two options are used, namely the incomplete LU (ILU) factorization (Dupont et al., 1968) and the incomplete Cholesky (IC) factorization (Kershaw, 1978), both with zero fill-in. For the block Jacobi setup with ILU, the preconditioner $\mathbf{M}_i = \tilde{L}_i \tilde{U}_i$, where $\tilde{L}_i$ and $\tilde{U}_i$ are incomplete lower and upper triangular factors of local diagonal block matrix of $\mathbf{A}_i$, and $\mathbf{M}_i^{-1} = (\tilde{L}_i \tilde{U}_i)^{-1}$. And for the block Jacobi setup with IC, the preconditioner $\mathbf{M}_i = \tilde{L}_i \tilde{L}_i^\text{T}$, and $\mathbf{M}_i^{-1} = (\tilde{L}_i \tilde{L}_i^\text{T})^{-1}$.

(iii) *Algebraic multigrid preconditioning.* The above discussed preconditioning options are from the class of 'one-level' preconditioners; the AMG preconditioning discussed here falls under the category of 'multi-level' preconditioners. The AMG preconditioning is widely used, due to its optimal computational cost and algorithmic scalability. It accelerates the convergence of the CG method by building a hierarchy of coarser grids operators (solutions) from the fine grid operator. In other words, a series of independent coarse grid systems are used to accelerate the convergence of the overall fine grid system $\mathbf{Ax} = \mathbf{b}$. Note that, the AMG does not require access to the geometric grid, "grid" within the context of AMG preconditioning refers to set of indices of the unknown variables within the linear system $\mathbf{Ax} = \mathbf{b}$. The hierarchy of grids are obtained for a problem, starting from the finest grid level $\mathbf{A}$ and reducing the number of unknowns to get the coarser levels, until the coarsest level is small enough to be solved with exact LU factorization. Generally, the coarsest level contains a small number of

Table 1: packages used for setting up the monolithic vectorial FEM fracture mechanics solver. Note that, except for Gmsh all other packages were compiled with the Message Passing Interface standard (MPI) library (Gropp et al., 1999) to set up workload in distributed parallel computing framework.

package	version	operation	reference
Gmsh	4.3.0	unstructured mesh generator	Geuzaine and Remacle (2009)
parMETIS	4.0.3	mesh partitioner	Karypis et al. (1997)
FreeFEM	4.5	finite element kernel	Hecht (2012)
PETSc	3.12	linear algebra solvers and preconditioners	Balay et al. (2019)
ParView	5.5.2	result post processing	Ahrens et al. (2005)

unknowns in comparison to the finer levels, thus the cost of performing the exact LU factorization for the coarsest level is negligible in comparison to the overall solving time for the finest level linear system. Based on how coarsening is achieved, AMG is classified as the classical AMG (Ruge and Stüben, 1987), the agglomeration AMG (Jones and Vassilevski, 2001), and the smoothed aggregation AMG (Vaněk et al., 1996). Due to symmetric positive definite nature of the assembled matrix $\mathbf{A}$, the smoothed aggregation AMG is employed in this study which was originally developed in (Vaněk et al., 1996) for elliptic systems. A key component of smoothed aggregation AMG is its smoother, which make the underlying error on the fine grids smooth so that it can be approximated accurately on the coarser grids. In other words, the smoother eliminates errors associated to large eigenvalues of the linear system, while the coarse grid correction eliminates the remaining errors associated to the small eigenvalues. There are many well known smoothers that are suitable for parallel AMG, c.f. Baker et al. (2011a,b), among these we employ the Richardson's and the Chebyshev smoothers in this paper.

5. Results and discussions

This section presents details, results, and discussions for the series of numerical experiments performed in this study. Our primary motive of performing different numerical experiments is to analyze, tune, and propose an optimal preconditioner for the hybrid phase-field fracture mechanics vectorial FEM solver capable of handling large-scale problems with millions of unknowns. We end this section by applying the tuned preconditioner to solve a complex three-dimensional brittle fracture problem with more than 64 million unknowns.

5.1. Hardware and software specifications

Different open-source frameworks (domain-specific language, mathematical libraries, etc.) were employed in this study to develop and implement the monolithic vectorial FEM solver and consequently perform the various sequential/parallel numerical experiments that are to appear in this section. Details of these open-source frameworks are provided in table 1.

The numerical experiments in this study were performed by utilizing computational resources that range from an ordinary "desktop PC" to a supercomputer "Inti" hosted at TGCC-CEA[3], Bruyères-le-Châtel, France. The specifications of these two computational resources are provided in table 2. The "desktop PC" was used for executing the numerical experiments appearing in section 5.2, and for the numerical experiments in sections 5.3 and 5.4 Skylake nodes of "Inti" were used. For all the numerical experiments to follow, the CG algorithm convergence is assumed to be reached when the relative residual norm of the linear system is lower than 10^{-10} or when the absolute residual norm is lower than 10^{-10}. Additionally, the nonlinear Picard iteration at a given t are stopped for tolerances lower than 10^{-8}.

Before proceeding further, in order to assist the analysis of results in the upcoming subsections, we introduce the following nomenclature:

[3]TGCC-CEA, "Très Grand Centre de Calcul" is an infrastructure for scientific high-performance computing and Big Data, able to host petascale supercomputers.

Table 2: details of computational resources used for performing the serial/parallel numerical experiments in this study. In the table header "PU" denotes processing units (cores/threads).

name	#nodes	#PU	architecture	test runs	interconnection
desktop PC	1	10	Intel Xeon E5-2680 v2-@2.80GHz	sequential/parallel	-
Inti	21	48	Intel Xeon E5-8890 v4-@3.4GHz	parallel	InfiniBand EDR

(i) to annotate different CPU timings, the total solver time t_{total} is computed as the sum of three contributions:

$$t_{\text{total}} = t_{\text{assembly}} + t_{\text{setup}} + t_{\text{solve}},$$

where t_{assembly} denotes the CPU time spent in the linear system assembly phase (i.e., to construct $\mathbf{A}$ and $\mathbf{b}$), t_{setup} is the CPU time spent in setting up the preconditioner operator $\mathbf{M}^{-1}$, and t_{solve} the CPU time spent in performing the CG iterations till convergence. We also denote the total solution phase CPU time t_{solution}:

$$t_{\text{solution}} = t_{\text{setup}} + t_{\text{solve}};$$

(ii) CG iteration number is denoted by k, the total number of CG iterations to convergence (within one iteration of the global iterative solving, i.e., from iteration jth to $j+1$th in eq. (15)) is denoted by k_{c}, and $\sum k_{\text{c}}$ denotes the total number of CG iteration during one full simulation (i.e. for executing the full phase-field simulation);

(iii) at kth CG iteration, r_k denotes the unpreconditioned residual, such that $r_k = ||\mathbf{A}\mathbf{x}_k - \mathbf{b}||_2$, with $\mathbf{x}_k$ being the solution of the linear system;

(iv) the CG residual reduction rate R_{r_k} is defined by:

$$R_{r_k} = \frac{\sum_{k=1}^{k_{\text{c}}} (r_{k-1}/r_k)}{k_{\text{c}} - 1}.$$

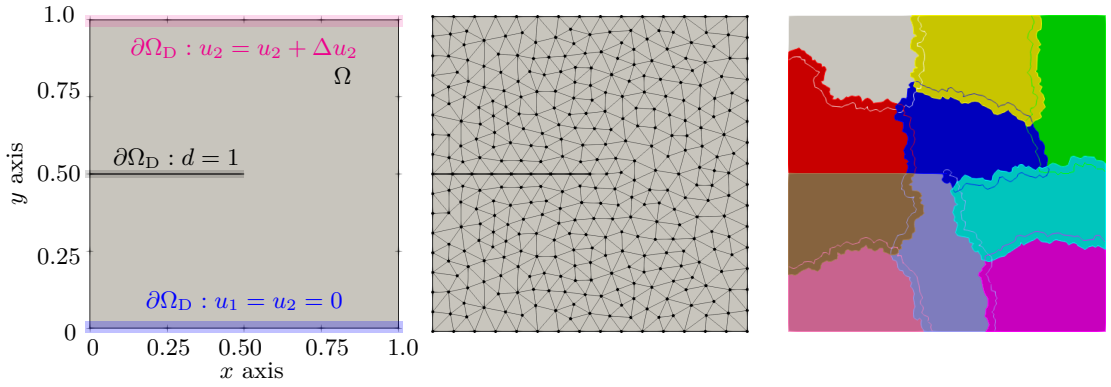
5.2. Two-dimensional numerical experiments: solver validation and preconditioner performance assessment

Commonly used numerical tests from literature (see e.g., Ambati et al. (2014), Liu et al. (2016), Jeong et al. (2018), Hirshikesh et al. (2018) to cite but a few), the two-dimensional (2D) single-edge notched tensile and shear fracture tests, are considered as benchmark problems in this subsection. From here-forth the tensile test is referenced as test 1 in the text.

5.2.1. Problem setting

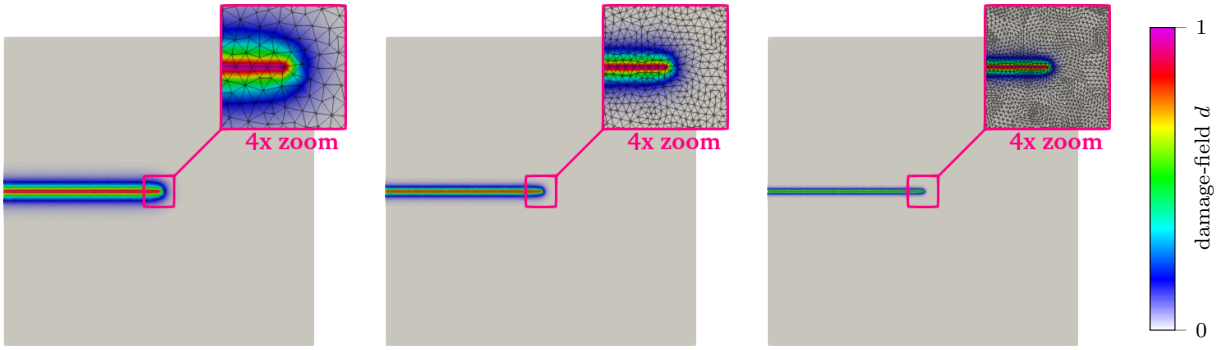
The domain of interest is an initially cracked square plate $(x, y) \in \Omega = [0\text{ cm}, 1\text{ cm}]^2$ (fig. 1a). With an initial crack and a constrained bottom edge $\partial\Omega_{\text{D}}(x, y : y = 0)$, the plate is subject to increasing displacements on its top edge $\partial\Omega_{\text{D}}(x, y : y = 1)$ until the plate fully cracks open. The initial crack is placed at the center of the plate, i.e., $\partial\Omega_{\text{D}}(x : 0 \leq x \leq 0.5, y : y = 0.5)$. These boundary conditions are also illustrated in fig. 1a. The plate material is characterized by $\lambda = 121.15$ kPa, $\mu = 80.77$ kPa, and $G_{\text{c}} = 2.7\text{ kN mm}^{-1}$.

Concerning the computational specifications of test 1, the displacement discontinuity imposed by the initial crack was modeled by nearly overlapping (tolerance $\delta y = 10^{-7}$ m) Dirichlet nodes placed along the cracks edge $\partial\Omega^h_{\text{D}}(x : 0 \leq x \leq 0.5, y : y = 0.5 \pm \delta y)$ within Ω^h. For illustration proposes, a coarse grid featuring Dirichlet nodes for the initial crack of test 1 is presented in fig. 1b. The displacement Dirichlet condition on the top edge is applied with an increment of $\Delta\bar{u}_2 = 1 \cdot 10^{-5}$ mm up to $u_2 = 5 \cdot 10^{-3}$ mm and $\Delta\bar{u}_2 = 1 \cdot 10^{-6}$ mm up to failure of the specimen. For the lower edge, the constrained displacement Dirichlet conditions $\bar{u}_1 = \bar{u}_2 = 0$ are applied. Further, for test 1 and for all the simulations that appear in this study,



(a) domain Ω. (b) a coarse mesh Ω^h. (c) partitioned mesh $\{\Omega_i^h\}_{i=1}^{10}$.

Figure 1: domain Ω, mesh Ω^h, and partitioned mesh $\{\Omega_i^h\}_{i=1}^{10}$ for test 1. (a) also illustrates the boundary conditions applied to test 1. (b) represents a coarse unstructured finite element mesh with 'nearly' duplicate Dirichlet nodes for the initial crack.



(a) d at mesh level 1. (b) d at mesh level 2. (c) d at mesh level 3.

Figure 2: initial crack visualization of test 1 via damage-field d at different mesh levels.

parameter κ is set to $1 \cdot 10^{-6}$ and l_0 is assumed equal to $2h$, where h is the characteristic size of the mesh Ω^h.

The unstructured Delaunay (triangular) meshes generated with Gmsh are used for solving the finite element problem of test 1. To establish mesh convergence, test 1 has been solved multiple times by varying the level of mesh refinements, details of these meshes are provided in table 3. The hierarchy of mesh refinements were generated by dividing each triangle in Ω^h into four equal triangles. As such in table 3, we observe that with every refinement, the mesh size h halves and the number of triangles quadruple. The initial crack fields for the three mesh refinements (visualized using damage-field d) are presented in fig. 2.

5.2.2. Solver validation

Besides being used in the upcoming subsection to assess the performance of different preconditioners, we also used the test 1 to cross-validate and compare our vectorial FEM solver (sequential and parallel) against benchmark solutions of this test available in the literature. In fig. 3, the top surface reaction force F_y versus applied displacements is plotted for finest mesh level 3 (detailed in table 3) and compared to reference hybrid phase-field solution from Ambati et al. (2014) and anisotropic phase-field solution from Miehe et al. (2010a). The force-displacement curve obtained from our computation at the finest mesh level 3 is in good agreement with the two reference solutions. This simulation was executed using 10 processes on the desktop PC mentioned in table 2. The parMETIS partitioned mesh with 10 subdomains is presented in fig. 1c.

Table 3: characteristics and computational details for the different finite element meshes used for test 1. $E1_{\max(F_y)}$ and $E2_{\max(F_y)}$ are are the the maximum reaction force errors computed against references Ambati et al. (2014) and Miehe et al. (2010a), respectively.

mesh	nodes	triangles	h	N_{DOF}	N_{nz}	$E1_{\max(\boldsymbol{f})}$	$E2_{\max(\boldsymbol{f})}$
level 1	8,353	16,384	0.0156	25,059	520,425	15.62%	14.78%
level 2	33,089	65,536	0.0078	99,267	2,073,033	6.71%	5.89%
level 3	131,713	262,144	0.0039	395,139	8,274,825	2.00%	1.01%

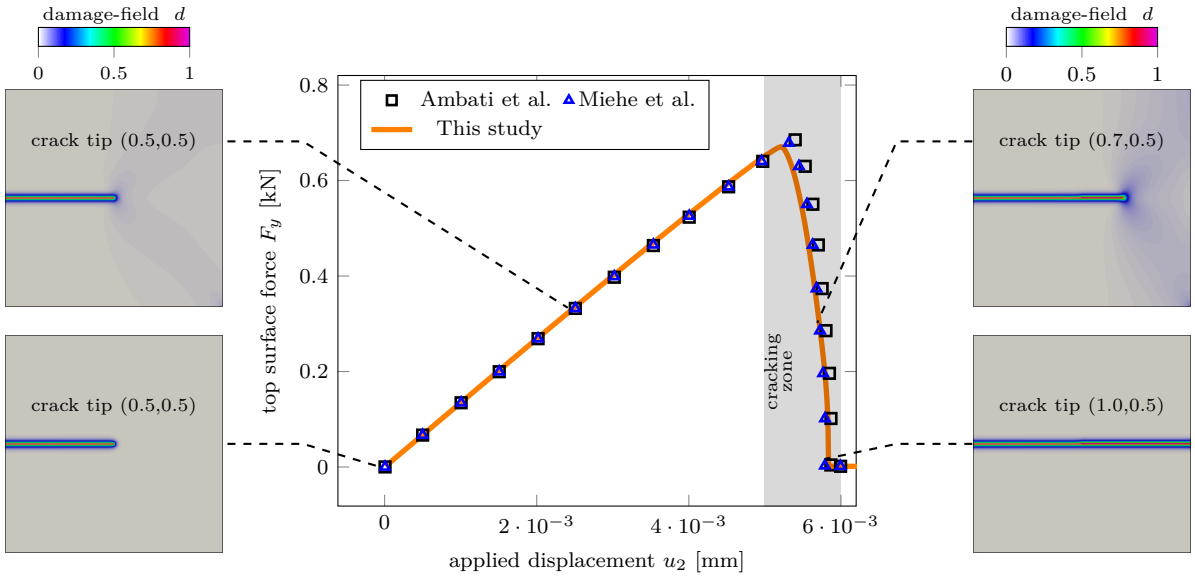
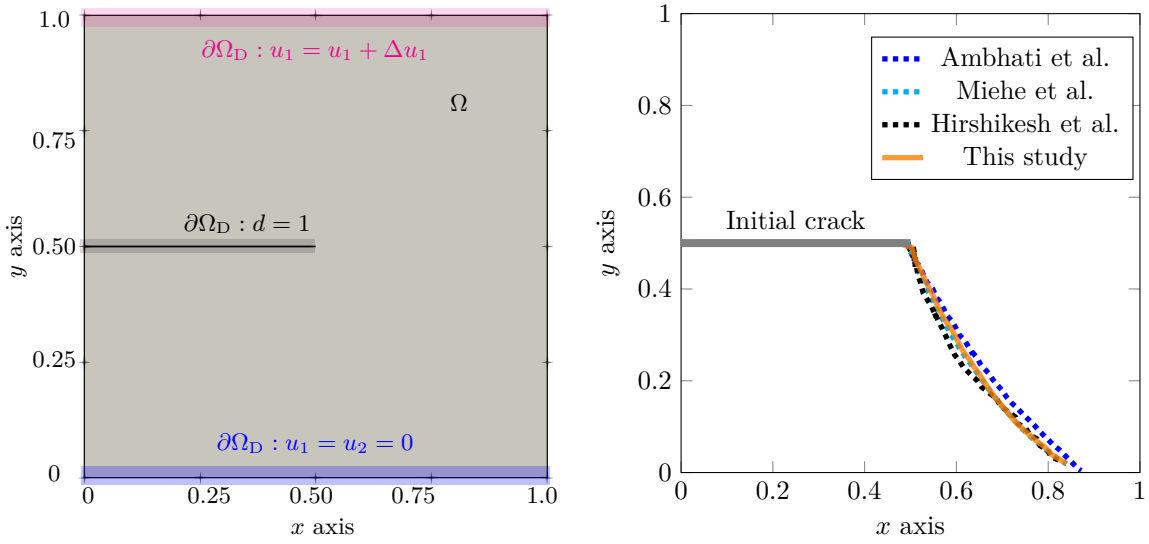


Figure 3: force-displacement plot for the two-dimensional single-edge notched tensile fracture simulation, test 1. The orange solid line refers to our vectorial FEM solution, the square markers denote the reference hybrid phase-field solution from Ambati et al. (2014) and the triangular markers denote anisotropic phase-field solution from Miehe et al. (2010b).

To further attest the correctness of the proposed vectorial FEM solver, we compare the errors in computing the top surface maximum reaction-force $\max(F_y)$ obtained from our solver against two different reference solutions provided in Ambati et al. (2014) and Miehe et al. (2010a). The last two columns of table 3 enumerate these errors at different mesh levels. At finest mesh level 3, these errors decrease down to 2% and 1%, respectively in comparison to the hybrid and the anisotropic phase-field solutions. This reduction in error while decreasing the mesh size suggests mesh convergence of vectorial FEM solutions towards the reference solutions from Ambati et al. (2014) and Miehe et al. (2010a).

Alongside the plot in fig. 3, four instantaneous snapshots of the calculated damage-fields are presented. These damage-fields are obtained from the simulation of test 1 at the finest mesh level 3. Damage-field evolution, crack initiation, and propagation can be observed in these snapshots. As expected, under extreme tensile loading, the crack can be seen to travel along a (almost) straight line dividing the square specimen into two (almost) equal halves.

Further validation is provided by performing the shear test for the same specimen of test 1. Now, the specimen is loaded in shear mode, i.e, the displacement Dirichlet condition (now on u_1) on the top edge is applied with an increment of $\Delta\bar{u}_1 = 1 \cdot 10^{-5}$ mm (see fig. 4a). Note that for this test and the previous one small displacement increments ($< 1 \cdot 10^{-5}$ mm) are applied. Such small displacement increments are taken in view of easily achieving convergence for the nonlinear Picard iterations. For this test, the applied shear loading causes the top half of the specimen (above the initial crack) to be in compressive stress state while its bottom half (below the initial crack) remains in tensile stress state. As such, the crack is expected to



(a) domain Ω.

(b) final crack trajectories.

Figure 4: shear cracking test domain Ω and final cracking trajectory in compared to the trajectories obtained by the hybrid phase-field model (Ambati et al., 2014), the anisotropic phase-field model (Miehe et al., 2010a), and adaptive phase-field modeling (Hirshikesh et al., 2019).

propagate within the tensile stress state region (underneath the initial crack). Such a cracking trajectory obtained from our solver is compared to other reference cracking trajectories from literature in fig. 4b. The finest mesh level 3 was used to obtain this cracking trajectory, and the simulation was executed using 10 processes on the desktop PC mentioned in table 2.

In fig. 5, the force-displacements plot for the shear test at the finest mesh level 3 is presented and compared to reference hybrid phase-field solution from Ambati et al. (2014) and anisotropic phase-field solution from Miehe et al. (2010a). Our solution is in good agreement with the reference hybrid phase-field solution from Ambati et al. (2014), however some differences are observed post peak force when compared to the anisotropic phase-filed solution from Miehe et al. (2010a).

The mentioned shear test approves the functionality of the use of tensile elastic energy ψ^+ to control the evolution of damage as suggested by the hybrid phase-field formulation (as previously mentioned in section 2.2). It has been previously shown in Ambati et al. (2014), Hirshikesh et al. (2018) use of such tensile elastic energy ψ^+ becomes inevitable to accurately reproduce the cracking trajectories presented in fig. 4b.

Note that for additional validation, other literature comparative tests (mode I, mode II, and mode III fracture) were also performed but these are not shown here for the sake of conciseness.

5.2.3. Effect of preconditioning on iteration count, condition number, and problem size

In the phase-field model, the coefficients of $\mathbf{A}$ are approximately dependent on $(1-d)^2\mathbb{E}$ (with $\mathbb{E}$ denoting the fourth-order elastic stiffness tensor, see eq. (1). As d approaches to 1 at certain (cracked) mesh nodes, this causes the condition number, $\text{cond}_2(\mathbf{A})$, to change during the numerical approximation of cracking processes. A direct effect of this change in the $\text{cond}_2(\mathbf{A})$ is sensed by the CG method, where the total number of iterations to converge (k_c) drastically change during the progression of the simulation.

Besides improving the convergence of the CG method, one of the roles of preconditioning operator $\mathbf{M}^{-1}$ is also to minimize the drastic change in k_c during the full phase-field simulation, or in other words control the drastic change in $\text{cond}_2(\mathbf{A})$. In our numerical experiments from this subsection, we use such quality of the preconditioner as a metric to judge the efficiency of the preconditioning operator.

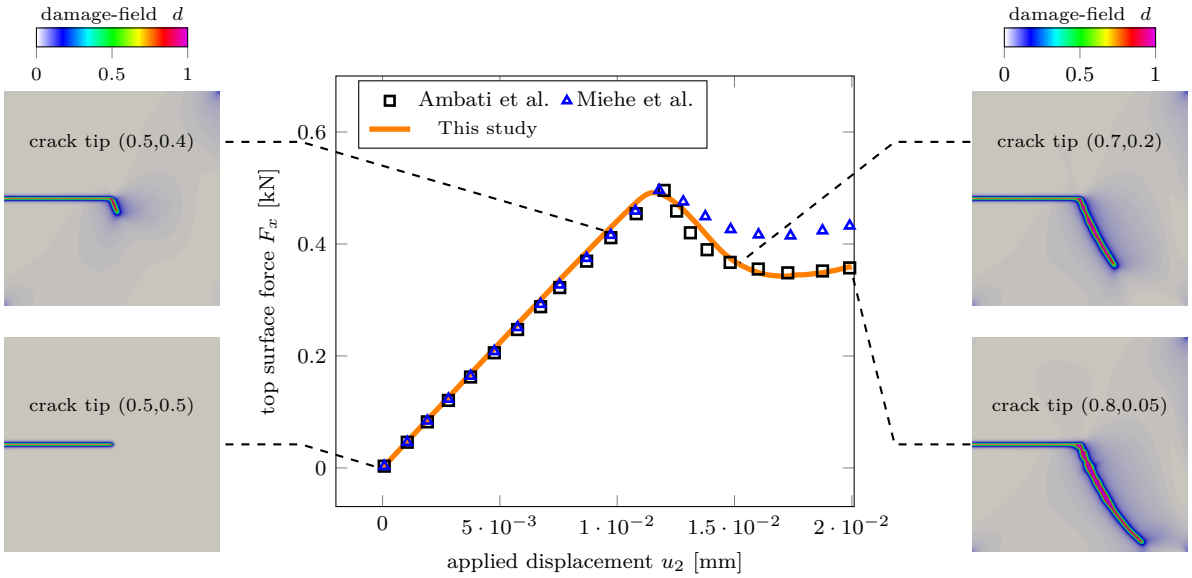


Figure 5: force-displacement plot for the two-dimensional single-edge notched shear fracture simulation, test 1a. The orange solid line refers to our vectorial FEM solution, the square markers denote the reference hybrid phase-field solution from Ambati et al. (2014) and the triangular markers denote anisotropic phase-field solution from Miehe et al. (2010b).

Iterations. Figure 6 plots the complete evolution of k_c observed while performing the full phase-field simulation, test 1 with level 1 mesh. The simulation was performed in parallel using 10 processes (MPI ranks) on the desktop PC from table 2. One can observe the effectiveness of preconditioning by noticing how k_c gets reduced when CG method is used alongside different preconditioners.

In particular, k_c with a simple Jacobi preconditioner, ranging from 593 to 920 reduces to a range of 152 to 202 when using a more robust block Jacobi preconditioner. Compared to the Jacobi and block Jacobi preconditioners, further drop in k_c range can be observed with the AMG preconditioner, the k_c iteration ranging now from 75 to 91.

For this test, the diagonal block matrices $\{\mathbf{M}_i\}_{i=1}^{N_p}$ of the block Jacobi preconditioner were factorized by using zero fill-in incomplete Cholesky (IC) decomposition. For conciseness we shall drop the words "zero fill-in" and refer this preconditioner in the text as "block Jacobi IC". Further, the AMG preconditioner was in fact the smoothed aggregate AMG preconditioner (Vaněk et al., 1996) with the Chebyshev smoother and block Jacobi IC inner solver for the coarse grids and the coarsest grid being factorized by complete LU. The reason for choosing such kind of preconditioners will be discussed in upcoming section 5.3.

Within fig. 6, a strong rise in CG iterations to converge is observed as the load passes from 5.3×10^{-3} to 6×10^{-3}, i.e., during the damage propagation phase (illustrated in gray color in fig. 6). The strong rise in CG iteration count k_c serves as an indicator that the conditioning of the assembled preconditioned linear systems ($\text{cond}_2(\mathbf{M}^{-1}\mathbf{A})$) must be changing drastically in the cracking zone.

In fig. 6, we also provide the data pertaining to the total CG iterations $\sum k_c$ needed to perform the complete phase-field simulations (1490 pseudo-time steps) with Jacobi, block Jacobi, and AMG preconditioners. Compared to the $\sim$ 1.1 million CG Jacobi iterations needed to completely execute the simulation, the block Jacobi (resp. AMG) completed the simulation in $\sim$ 1/4th (resp. $\sim$ 1/10th) number of Jacobi iterations.

Condition numbers. The speed of convergence for the CG method can be predicted by the knowledge of the condition number of the solved matrix $\mathbf{A}$ denoted by $\text{cond}_2(\mathbf{A}) \geq 1$. The convergence of the CG method relies on the eigenvalue distribution of $\mathbf{A}$, and $\text{cond}_2(\mathbf{A})$ provides a criterion for assessing the width of its

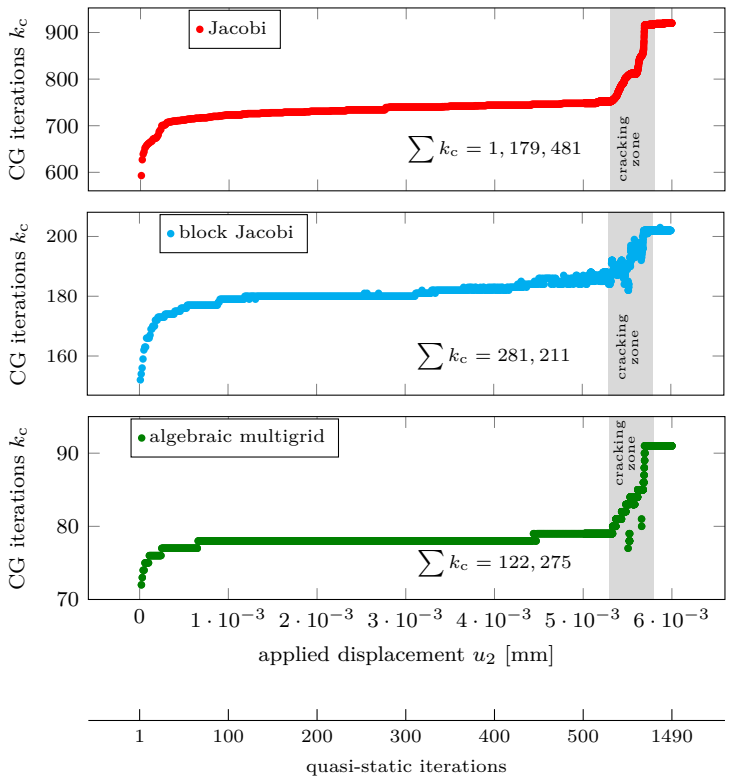


Figure 6: evolution of the CG iterations to converge k_c during execution of the full phase-field simulation (test 1 with level 1 mesh).

spectrum. Following Basermann et al. (1997), let us denote $\gamma := (\sqrt{\text{cond}_2(\mathbf{A})} - 1)/(\sqrt{\text{cond}_2(\mathbf{A})} + 1)$. The distance between the CG solution $\mathbf{x}_k$ at kth iterate from exact solution $\bar{\mathbf{x}}$ is bounded:

$$\|\mathbf{x}_k - \bar{\mathbf{x}}\|_2 \leq 2\gamma^k \sqrt{\text{cond}_2(\mathbf{A})} \|\mathbf{x}_0 - \bar{\mathbf{x}}\|_2.$$

As such, the lower the condition number the faster the convergence of CG, or in other words lower k_c.

Now, to investigate on the effect of preconditioning operators, fig. 7 presents the condition number evolution during the full phase-field simulations of test 1 with mesh level 1. The $\text{cond}_2(\mathbf{M}^{-1}\mathbf{A})$ is calculated by employing the Lanczos iteration algorithm for eigenvalue computation (Balay et al., 2019).

In practice, at each pseudo-time step, the Lanczos algorithm is used to calculate the eigenspectrum $\{\lambda_i\}_{i=1}^{N_{\text{eigen}}}$ of matrix $\mathbf{M}^{-1}\mathbf{A}$, with N_{eigen} denoting the total number of eigenvalues calculated. Then, we use the approximation: $\text{cond}_2(\mathbf{M}^{-1}\mathbf{A}) \approx \max|\lambda_i| / \min|\lambda_i|$. Note that the eigenspectrum $\{\lambda_i\}_{i=1}^{N_{\text{eigen}}}$ provided by the Lanczos algorithm cannot be used for complete eigenvalue analysis; however it is intended here to assist in understanding the convergence behavior of the CG and to quantify the efficiency of the preconditioning operators (Balay et al., 2019, Badri et al., 2019).

Complementary to the results from fig. 6, the condition number evolution provided in fig. 7 reveals that AMG preconditioning is able to reduce the condition number deterioration during cracking. Hence the problem is easier to solve when compared with other preconditioning options. As suspected by the rise in k_c from fig. 6, the plots from fig. 7 illustrates that $\text{cond}_2(\mathbf{M}^{-1}\mathbf{A})$ rises exponentially as the crack starts to propagate (highlighted in gray color). Further analysis of these data reveals that in comparison to the Jacobi preconditioner, the average $\text{cond}_2(\mathbf{M}^{-1}\mathbf{A})$ for the AMG (resp. the block Jacobi) is 1.8 (resp. 1.3) times smaller. This proves the superiority of the AMG preconditioner to "easily" solve the brittle fracture problems arising in the phase-field modeling.

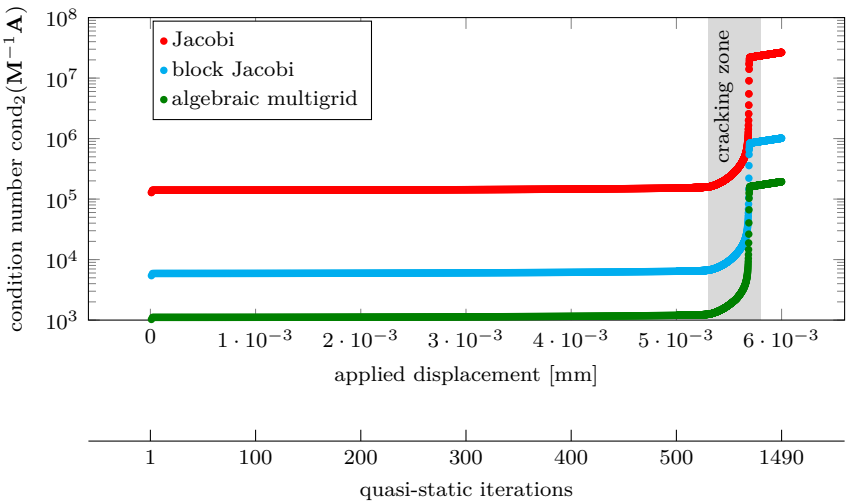


Figure 7: evolution of condition number $\mathrm{cond}_2(\mathbf{M}^{-1}\mathbf{A})$ during execution of the full phase-field simulation (test 1 with level 1 mesh).

The main aim of this subsection was to study condition number degradation caused by cracking, while the condition numbers were the ones estimated from the monolithic linear system assembled for the hybrid phase-field model. It should be noted that the magnitude deference between displacement and damage fields partly contributes to poor conditioning of the studied linear systems. Dealing with a dimensionless formulation (i.e., scaling the problem) could lead to linear systems with relatively different condition numbers (possibly lower). In this study we do not deal with a dimensionless formulation, however for such cases as well cracking should lead to rise in condition number (relative to the non-cracked state). As such, the conclusion that the AMG preconditioner for CG is relatively superior to the Jacobi or the block Jacobi preconditioner should apply to linear systems assembled from a dimensionless formulations too.

Influence of the mesh refinement. In order to gauge the sensitivity of the three different preconditioning operators with respect to the refining of the mesh (problem size), table 4 presents total CG iterations to converge k_c for a single pseudo-time step (9th) of the phase-field simulation test 1 with different mesh refinements. Since the highest order derivative for the equations of the phase-field model (eqs. (1) and (2)) looks like the Laplacian, as such, the condition number with the CG method will be $O(h)$. So k_c should increase as the square root of the problem size (number of elements).

Denoting the change in iteration count between simulations at mesh levels 1 and i by $Dk_{c,1\to i}$:

$$Dk_{c,1\to i} = k_{c,i}/k_{c,1} = \sqrt{N_{e,i}/N_{e,1}},$$

with $k_{c,i}$ and $N_{e,i}$ corresponding respectively to CG iterations to converge and number of elements for ith mesh level. In our numerical experiment, the mesh elements are quadrupled between mesh levels 1 and 2 (see table 3), therefore one observes approximately twice ($\sqrt{65536/16384} = \sqrt{4} = 2$) increase in iterations $Dk_{c,1\to 2}$ for the CG Jacobi method, since it is a weak preconditioner (see table 4 second row third column). This trend continues with the other mesh levels for the Jacobi preconditioner. It indicates that the Jacobi preconditioner is a weak preconditioner for the phase-field problems of fracture in the sense of not being able to control the rise in CG iteration count $Dk_{c,1\to i}$ between mesh refinements.

In contrast to what was observed for the Jacobi preconditioner, in table 4, we observe that the CG setup with the AMG preconditioning remains the least sensitive to mesh refinement. At the finest mesh refinement (level 5), for the AMG preconditioner we observe a $Dk_{c,1\to 5} = 10.63$ in comparison to $Dk_{c,1\to 5} = 14.43$

Table 4: total CG iterations to converge for different preconditioning options. Numbers in brackets indicate the rise in iteration count $Dk_{\mathrm{c},1\to i}$. $k_{\mathrm{c}}^{\mathrm{Jacobi}}$, $k_{\mathrm{c}}^{\mathrm{B\text{-}Jacobi}}$, and $k_{\mathrm{c}}^{\mathrm{AMG}}$ refer to the total number of CG iterations to converge k_{c} for Jacobi, block Jacobi, and AMG preconditioners respectively. Subscript "BJacobi" denotes block Jacobi.

level (i)	$N_{\mathrm{e},i}/N_{\mathrm{e},1}$	$k_{\mathrm{c}}^{\mathrm{Jacobi}}$	$k_{\mathrm{c}}^{\mathrm{BJacobi}}$	$k_{\mathrm{c}}^{\mathrm{AMG}}$
1	1	663 (1)	166 (1)	75 (1)
2	4	1,312 (1.98)	312 (1.88)	126 (1.68)
3	16	2,612 (3.94)	604 (3.64)	258 (3.45)
4	64	5,191 (7.83)	1,201 (7.24)	475 (6.34)
5	256	10,375 (15.65)	2,395 (14.43)	797 (10.63)

for the block Jacobi preconditioner, $Dk_{\mathrm{c},1\to5} = 15.65$ for the Jacobi preconditioner, and $Dk_{\mathrm{c},1\to5} = 16$ (theoretical) for no preconditioning. These results again highlight the superiority of the AMG preconditioner in comparison to the Jacobi or the block Jacobi preconditioners. Overall these results points out that for the AMG preconditioner CG convergence is least effected by mesh refinements. Although only a single pseudo-time step (9th one) was used here to showcase the effect of mesh refinement on different preconditioners, from the results presented in figs. 6 and 7, we know that similar conclusions should apply for other pseudo-time steps of the full phase-field simulation.

Overall in this subsection, the preconditioning analysis — based on CG iterations to converge k_{c}, total CG iterations needed for the full phase-field simulation $\sum k_{\mathrm{c}}$, sensitivity of k_{c} with mesh refinement, and condition number analysis — points towards AMG being an effective choice of preconditioner for solving brittle fracture problems using hybrid phase-field models in a vectorial FEM framework.

5.3. Three-dimensional numerical experiments: preconditioner tuning and scaling assessment

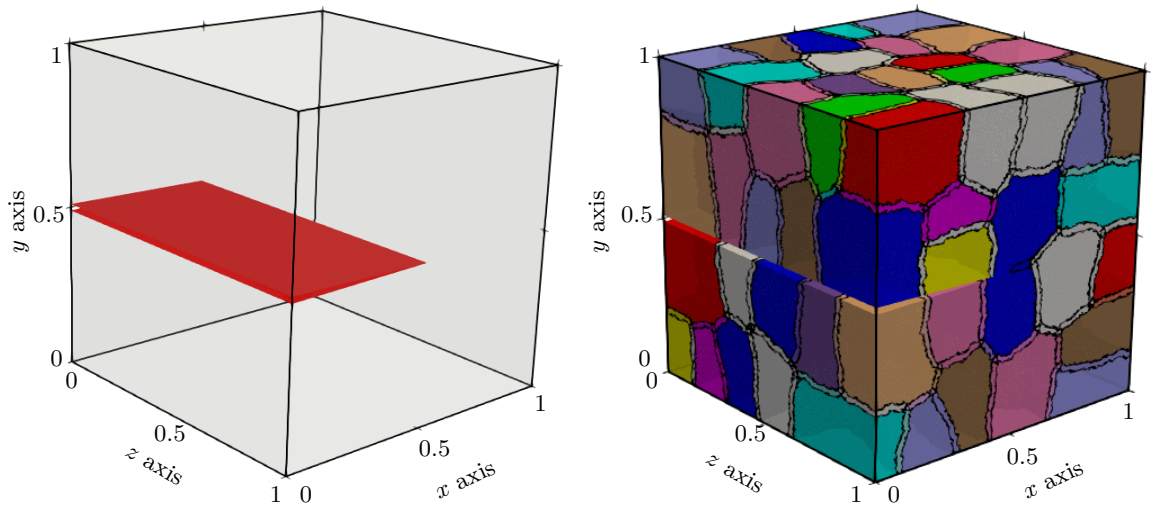
This subsection aims to test, tune, and compare the previously discussed preconditioning options for solving large-scale brittle fracture problems with the CG algorithm. In particular, two three-dimensional (3D) problems with 32,764,068 ($\sim$ 33 M) and 80,570,308 ($\sim$ 81 M) unknowns are used as numerical tests in this subsection.

Problem setting. The considered problem in here is a 3D simulation of tensile damage in a pre-cracked specimen (fig. 8a) obtained by a unit extrusion (along z direction) of the 2D domain considered in test 1. Similar to test 1, the bottom and top surfaces (xz-planes at $y = 0$ and $y = 1$) of the cubic geometry are respectively constrained and vertically loaded till failure. For the constrained plane $\partial\Omega_{\mathrm{D}}(x, y : y = 0, z)$, the displacements $\bar{u}_1 = \bar{u}_2 = \bar{u}_3 = 0$; and for loading plane $\partial\Omega_{\mathrm{D}}(x, y : y = 1, z)$, the displacement Dirichlet condition is applied on u_2 with an increment $\Delta\bar{u}_2 = 1 \cdot 10^{-5}$ mm.

The unstructured finite element mesh used for the $\sim$ 33 M (resp. $\sim$ 81 M) contained 8,191,017 (resp. 20,142,577) nodes and 47,759,360 (resp. 118,308,864) tetrahedral elements. From here-forth in the text, we denote the $\sim$ 33 M and $\sim$ 81 M problems by test 2A and test 2B, respectively. Figure 8b shows the partitioned mesh used for the test 2A. This partitioned mesh contains 96 subdomains and is used for parallel numerical simulations with 96 processes.

5.3.1. Preconditioner tuning

To identify the optimal preconditioning option for large-scale phase-field fracture simulations, the linear systems obtained by the vectorial FEM discretization of tests 2A and 2B were solved by using the CG method setup with Jacobi, block Jacobi, and AMG preconditioners. Each simulations was carried out using 384 processes on 8 Skylake nodes of the Inti supercomputer.



(a) domain Ω of test 2A and 2B.

(b) partitioned mesh $\{\Omega_i^h\}_{i=1}^{96}$ of test 2A.

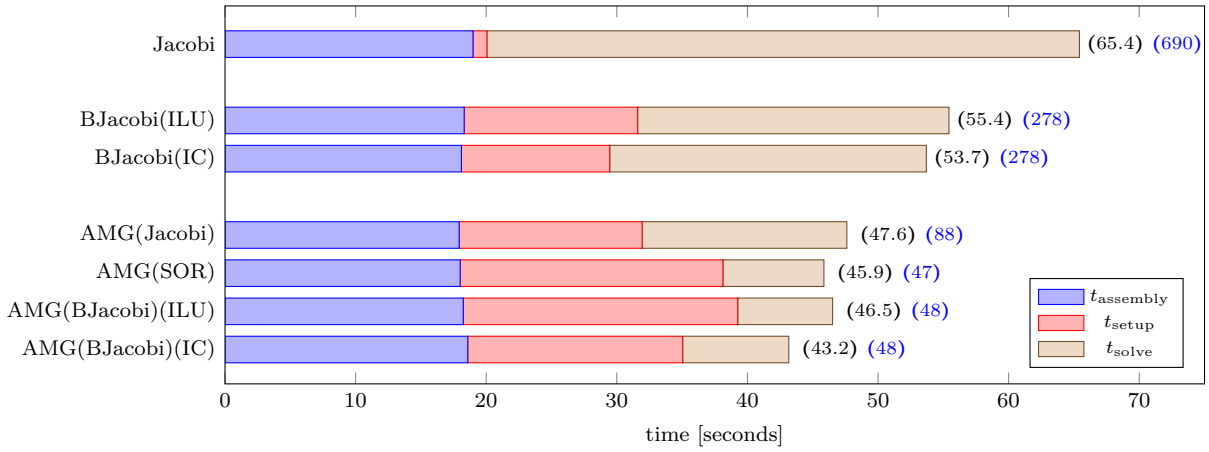
Figure 8: three-dimensional brittle cracking problem geometry and the partitioned finite element mesh $\{\Omega_i^h\}_{i=1}^{96}$. The pre-cracked surfaces in (a) are highlighted in red color .

Table 5: solution phase timings t_{solution} and CG iterations k_{c} observed by varying different tuning parameters of the AMG preconditioner.

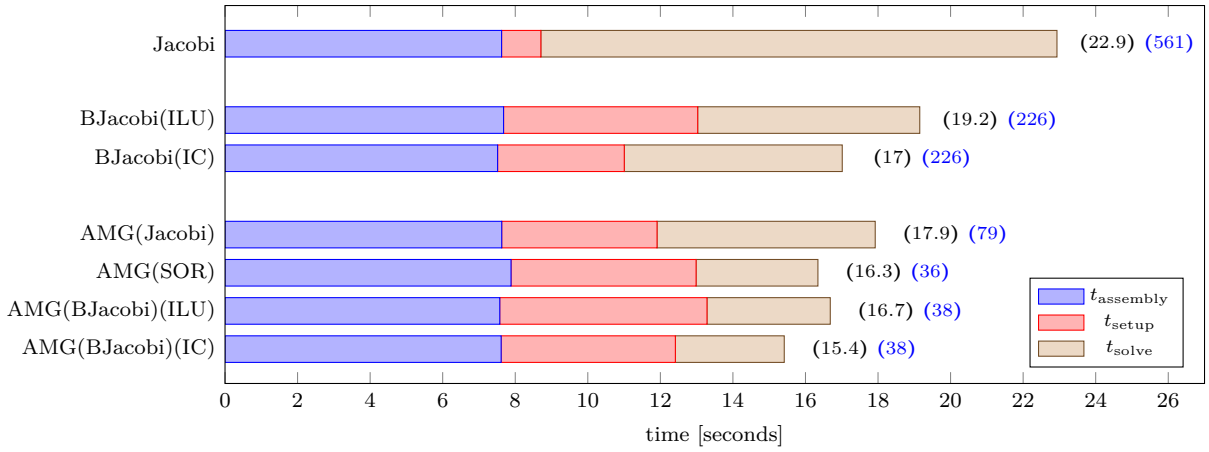
	max coarse space Krylov iterations					smoothed		smoother	
problem	1	2	3	4	5	yes	no	Chebyshev	Richardson
Timings									
test 2A	20.2	7.8	15.4	18.6	20.4	7.8	35.5	7.8	31.0
test 2B	62.2	24.5	77.1	83.9	95.7	24.5	108.4	24.5	97.3
Iterations									
test 2A	176	42	55	34	30	42	155	42	112
test 2B	214	48	108	89	84	48	187	48	214

CPU timings. In fig. 9, comparative CPU timings for the different preconditioning options are presented in the form of stacked bar plots. Each stacked bar within the plot represents t_{total} and has been color-coded with other CPU timings t_{assembly}, t_{setup}, and t_{solve}. The color-coding of these stacked bars help in analyzing the CPU cost of different phases of the full finite element solution. Note that for performance/tuning/scaling analysis of preconditioners presented in this subsection we shall only be using a single pseudo-time step (8th one) from the full brittle fracture mechanics simulations. Further, to avoid any hardware related discrepancies, all timings are averaged based on three simulation runs for a specific case. Additionally, fig. 9 also provides information of total CG iterations to convergence k_{c} for each preconditioning option.

Among the tested preconditioners, as expected, the CG setup with the Jacobi preconditioner yields high time to solution t_{total} and high number of CG iterations to converge k_{c}. Overall, the total solution time t_{total} of 22.93 (resp. 65.41) seconds was observed for the numerical simulation of test 2A (resp. test 2B) solved parallely using 384 processes. This t_{total} is decomposed into $t_{\text{assembly}} = 7.62$ (resp. 18.98) seconds and $t_{\text{solution}} = 15.31$ (resp. 46.43) seconds for the test 2A (resp. test 2B). Further, the Jacobi preconditioned CG took 561 (resp. 690) iterations to convergence for test 2A (resp. test 2B). Observe that the preconditioner setup cost t_{setup} is negligible (~ 1 % of the total solving time t_{total}). As the Jacobi preconditioning simply involves the use of diag($\mathbf{A}$) as the preconditioning operator $\mathbf{M}^{-1}$, t_{setup} is expected to be low.



(a) ∼ 81 M DOF problem, test 2B.



(b) ∼ 33 M DOF problem, test 2A.

Figure 9: stacked bar plots for CPU timings highlighting the effect of different preconditioners on overall solver timings for two three-dimensional brittle fracture phase-field problems test 2A and test 2B solved using 384 processes. In the figure "BJacobi" stand for block Jacobi. The numbers in the brackets next to the bar plots represent overall solver timings t_{total} in seconds (black) and the number of CG iterations k_{c} to convergence (blue).

The CG setup with the block Jacobi preconditioner performed faster (less t_{solution} and k_{c}) than the CG setup with the Jacobi preconditioner. Faster solution timings t_{solution} were observed when the 384 inner diagonal block matrices $\{\mathbf{M}_i\}_{i=1}^{384}$ of the block Jacobi preconditioner were solved using the IC factorization, in comparison to the zero fill-in incomplete LU factorization (ILU) being used as the block diagonal matrix solver.

Further, the performance of the block Jacobi preconditioners can be improved by increasing the fill-in level ("n") of the ILU(n) or IC(n) used as the block diagonal solver to obtain $\mathbf{M}^{-1}$. Note that with increase in n, ILU(n) (resp. IC(n)) approaches to complete LU (resp. Cholesky) factorization leading to a perfect preconditioner. However, since the goal of this article is to propose a preconditioner with good parallel performance for large-scale phase-field problems, we refrain from using the complete LU (resp. the complete Cholesky) factorization as diagonal block solvers to the block Jacobi preconditioner, and only use the zero fill-in versions of ILU and IC.

In comparison to the baseline run results from the Jacobi preconditioner, we observe that the t_{solution} of

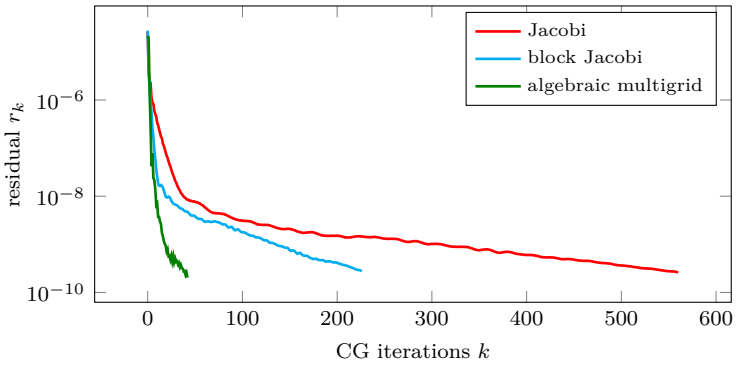


Figure 10: conjugate gradient method convergence history for the Jacobi, block Jacobi and algebraic multigrid preconditioners.

block Jacobi ILU is 1.33 (resp. 1.25) times lower than the t_{solution} of the Jacobi preconditioner for test 2A (resp. test 2B). We also observe that the t_{solution} of block Jacobi IC is 1.61 (resp. 1.31) times lower than the Jacobi preconditioner for test 2A (resp. test 2B).

The CG setup with the AMG preconditioner, when properly tuned, proves to be the fastest preconditioner among the other preconditioning options tested in this article. This statement is asserted by values of total solving time t_{total} and the CG iterations to k_{c} for the AMG preconditioner provided in fig. 9. Moreover, as the combined displacement-damage linear systems (from vectorial FEM) to be solved are symmetric, the smoothed aggregate AMG preconditioner outruns (both in t_{total} and k_{c}) the non-smoothed version (see table 5).

Among the smoothed aggregate, the Chebyshev smoother performed better (time-wise and iteration-wise) than the Richardson's smoother. To elaborate on this, in comparison to Richardson's smoother, the Chebyshev smoother uses the eigenvalue estimates for the smoothing procedure. Thanks to positive definite symmetric matrices provided by the vectorial FEM discretization, the smallest eigenvalues are easy to approximate by using CG as the eigenvalue estimator. As such, the Chebyshev smoother outperforms the Richardson's one.

Further, we tested which Krylov solver among the Jacobi, block Jacobi, and SOR performs the best as a coarse grid solver for the AMG preconditioner. Among them, the block Jacobi IC diagonal block solving outran the other two options. To further fine tune the AMG preconditioner, the coarse grid Krylov solver iterations were restricted to 1, 2, 3 , 4, and 5. It was revealed that 2 coarse grid solver iterations produce the fastest timings. In comparison to the baseline run results from the Jacobi preconditioner, we observe that the t_{solution} of the tuned AMG preconditioner is 2.0 (resp. 1.9) times lower than the t_{solution} of the Jacobi preconditioner for test 2B (resp. test 2A). Observe that in contrast to Jacobi preconditioner, where t_{setup} was negligible in comparison to t_{solution}, the same is not true for the AMG preconditioners. We observe almost 50 % of t_{solution} consists of the preconditioner setup cost, $t_{\text{setup}} \approx 0.5\, t_{\text{solution}}$.

The results provided for the tests 2A and 2B in this subsection identify the smoothed aggregate AMG with coarse grid block Jacobi IC as the optimal preconditioning option for solving the monolithic (vectorial) finite element linear systems that arises from the considered hybrid phase-field model. In comparison to other preconditioners that were tested in this subsection, the tuned AMG preconditioner exhibited low iteration counts, low solution timings, and faster convergence rates. Note that from here on in the text, we shall refer this tuned AMG preconditioner by simply "AMG preconditioner".

Convergence history. In fig. 10, we show the convergence history for the three different preconditioners applied to CG as it approaches convergence at k_{c}th iteration. Particularly, the relative residual r_k is plotted as a function of the CG iteration number $k \in [0, k_{\text{c}}]$ on a semilogarithmic plot. To obtain these plots, and for demonstration purposes, we simply used the test 2A solved on 384 processes. A nearly monotonic drop in

residual can be observed for the AMG preconditioned CG solver, while the same is not true for the Jacobi or the block Jacobi preconditioned CG solvers. The residual reduction rate of the AMG is clearly superior to that of the Jacobi or the block Jacobi preconditioned CG solvers. Similar trends were observed (not shown here) for the test 2B.

For both problems, we also approximate the residual reduction rate R_{r_k}, which quantifies how rapidly a preconditioner converges. For the test 2A (resp. test 2B), R_{r_k} was observed to be equal to 2.09 (resp. 2.09) with the AMG preconditioner, whereas it was observed equal to be 1.04 (resp. 1.03) and 1.11 (resp. 1.09) for the Jacobi and the block Jacobi preconditioners respectively. As such, with the AMG preconditioner, irrespective of the problem size, the residual approximately halves with each CG iterate as CG approaches to convergence. This is certainly a superior characteristic of the AMG preconditioner in caparison to the Jacobi and the block Jacobi preconditioners.

5.3.2. Scaling analysis

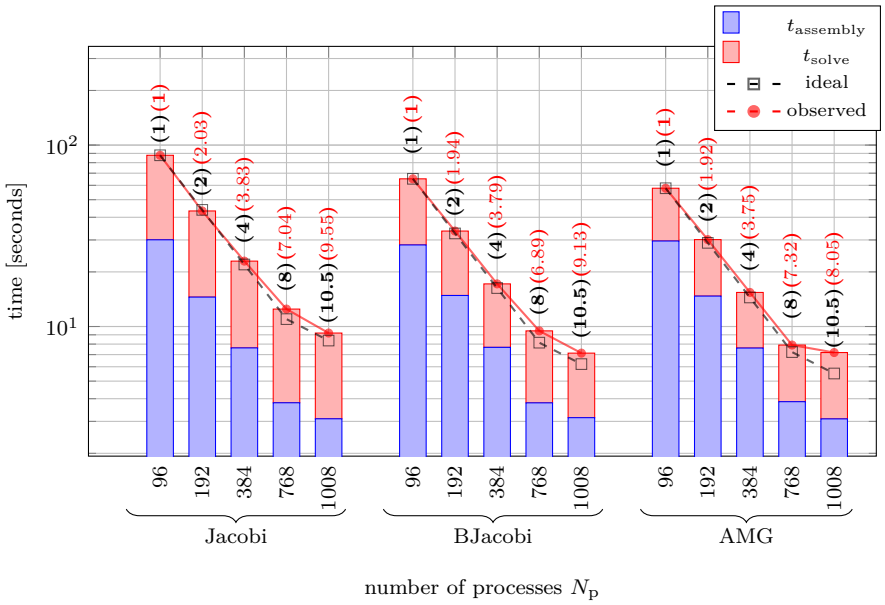
To analyze the parallelization efficiency of the proposed preconditioners, we now compare the strong parallel scaling performance of the Jacobi, block Jacobi IC, and tuned AMG preconditioners. The test 2A and 2B from the previous subsection, are used for the parallel scaling experiments which appear in this subsection. However, to avoid confusion, these tests are renamed as tests 3A and 3B. To perform strong scaling analysis, 96 to 1,008 processes (MPI ranks) on Inti supercomputer (see table 2) were used. On this machine, test runs with 96 processes correspond to 2 Skylake nodes ($N_\mathrm{p} = 2 \times 48 = 96$); similarly, 1,008 processes correspond to 21 Skylake nodes ($N_\mathrm{p} = 21 \times 48 = 1008$).

In fig. 11, CPU timings data provided by the scaling analysis are presented in the form of stacked bar plots. In particular, the CPU timings for linear system assembly and solving are highlighted. Additionally, the observed scaling for the different preconditioners is compared against the expected ideal scaling (theoretical). Within the plots, the ideal parallel scaling efficiency and the observed one are also highlighted.

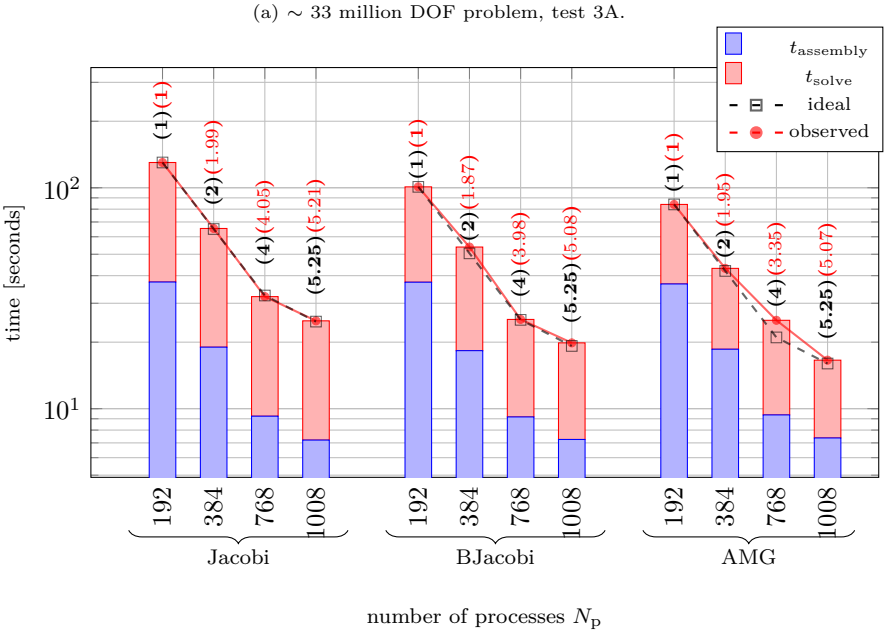
The scaling data suggests that all the considered preconditioning options lead to quasi-linear scaling behaviors when problem size is sufficiently large.

For test 3A, we observe excellent scaling efficiencies ($\sim$ 95 %) for the three tested preconditioners, when the number of processes is between 96 to 384. However, a slight decrease in parallel efficiency was observed when processor count increased above 384. The following scaling efficiencies were observed for N_p =1,008: $\sim$ 91 % for the CG Jacobi, $\sim$ 87 % for the CG block Jacobi IC, and $\sim$ 77 % for the CG AMG. In comparison to the parallel efficiency drop observed for the CG Jacobi, the efficiency drop observed for the CG AMG (resp. block Jacobi IC) preconditioner was significantly higher. This efficiency decrease is most likely caused by the higher MPI communication overhead of the CG AMG (resp. block Jacobi IC) solver in comparison to the CG Jacobi solver. Scaling-wise, all three preconditioning options have similar behavior. However, timing-wise, the CG AMG remained the fastest option. For instance, at N_p =96, the simulation executed in 28.2, 37.0, and 57.8 seconds (solving time t_solve) with the CG AMG, the CG block Jacobi IC, and the CG Jacobi preconditioners, respectively. This makes the CG AMG solver 2.05 (resp. 1.31) times faster than the CG Jacobi (resp. block Jacobi IC) solver. At N_p =1,008, the simulation solving times t_solve of 4.1, 3.8, and 6.1 seconds were reported for the CG AMG, the CG block Jacobi IC, and the CG Jacobi solvers, respectively. In this case, due to higher overhead of the CG AMG for this particular problem, the CG block Jacobi IC is the better option among the three preconditioners.

As expected, in comparison to the test 3A, better strong scaling characteristics were observed for the test 3B for the three tested preconditioners. Note that, due to the large problem size, scaling data was measured in between 192 to 1,008 processes. The following scaling efficiencies were observed for N_p =1,008: $\sim$ 99 % for the CG Jacobi, $\sim$ 97 % for the CG block Jacobi IC, and $\sim$ 96 % for the CG AMG. As observed for the test 3A, timing-wise, the CG AMG remained the fastest option among the three preconditioning options for the test 3B. For instance, at N_p =196, the simulation solving times t_solve of 47.4, 63.6, and 92.7 seconds were reported for the CG AMG, the CG block Jacobi IC, and the CG Jacobi, respectively. This makes the CG



(a) $\sim$ 33 million DOF problem, test 3A.

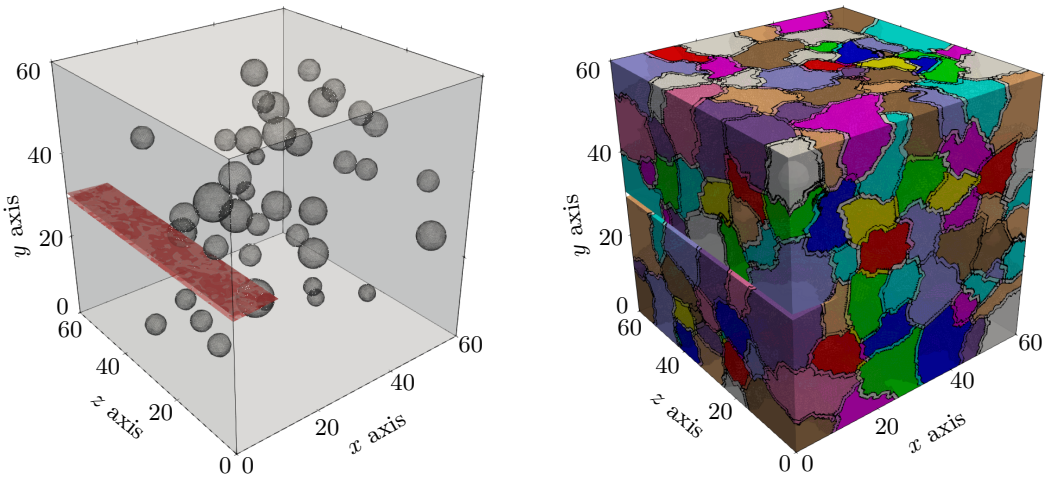


(b) $\sim$ 81 million DOF problem, test 3B.

Figure 11: bar plots depicting the strong scaling characteristics of different preconditioners up-till 1,008 cores for the three-dimensional brittle fracture problems, tests 3A and 3B, solved on Skylake nodes of Inti supercomputer. The numbers in the brackets next to the bar plots represent ideal speedup (black) and observed speedup (red). In the figure "BJacobi" stands for block Jacobi preconditioner.

AMG 1.96 and 1.34 times faster than the CG Jacobi and the CG block Jacobi IC. In contrast to test 3A, at N_{p} =1,008, the simulation solving times t_{solve} of 9.2, 12.6, and 17.7 seconds were reported for the CG AMG, the CG block Jacobi IC, and the CG Jacobi, respectively, making the CG AMG 1.92 and 1.36 times faster than the CG Jacobi and the CG block Jacobi IC.

Overall, all three preconditioner tested in this subsection for CG showed excellent parallel strong scaling



(a) domain Ω.

(b) partitioned mesh $\{\Omega_i^h\}_{i=1}^{1008}$.

Figure 12: large-scale perforated medium test domain Ω and the partitioned mesh $\{\Omega_i^h\}_{i=1}^{1008}$. Within (a) the initial crack surface is highlighted in red color and the different sized holes can also be observed.

characteristics, however, timing-wise CG AMG remains the fastest option.

5.4. Large-scale application: a three-dimensional perforated medium

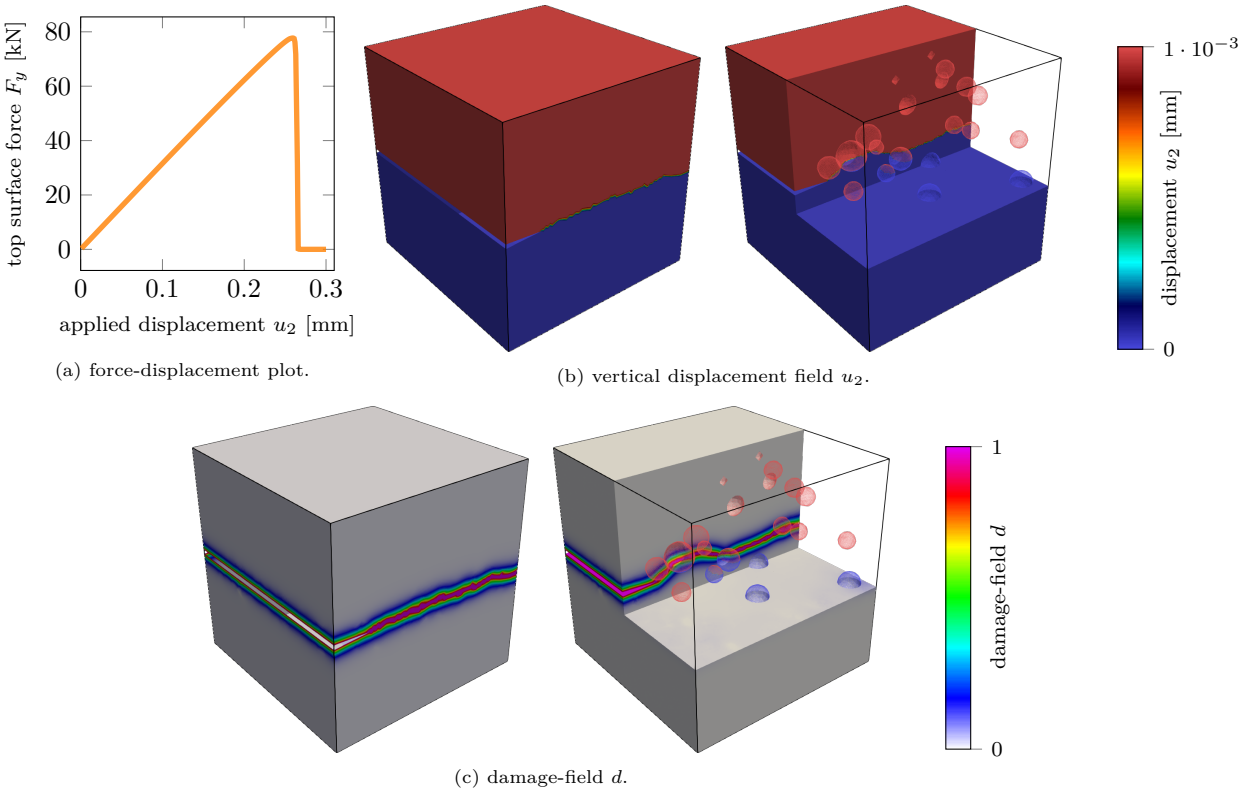
In this subsection, we present a mock application of perforated medium cracking. We demonstrate that by using the aforementioned parallel vectorial FEM formulation combined with the AMG preconditioner, we can "easily" perform the phase-field simulations of fracture propagation in a topologically complex medium, discretization of which leads to about 65,000,000 degrees of freedom.

Problem setting. A notched cubic specimen with the size of the side equal to 6 cm is submitted to tensile loading in the vertical direction (imposed via vertical displacements). The notch is placed at mid-height ($y = 3$ cm) of one side of the sample and is 1 cm in length. Its width is negligible if compared to the other dimensions of the computational domain.

To mimic a realistic perforated media, 44 spherical voids with different diameters are randomly arranged inside the specimen to induce a non-planar crack propagation. The generated geometry (fig. 12a) is such that the ratio between the solid phase and the voids is almost representative of concrete mortar. Mechanical parameters for performing the phase-field simulation are then chosen accordingly: $\lambda = 1.94$ kPa, $\mu = 2.45$ kPa, and $G_c = 2.28\ \mathrm{N\,mm^{-1}}$.

To induce damage initiation and evolution, the top surface of the numerical sample is loaded imposing a directly controlled vertical displacement increment of $\Delta\bar{u}_2 = 1 \cdot 10^{-3}$ mm, whereas the bottom face is fixed in all three directions $\bar{u}_1 = \bar{u}_2 = \bar{u}_3 = 0$. Remaining lateral faces are stress-free. Similar to the preceding tests, these conditions serve as Dirichlet conditions for our finite element formulations.

The fine meshing constraints of this test case lead to a problem with 64,456,128 degrees of freedom. A finite element mesh with 16,114,032 nodes and $98,295,595$ tetrahedral elements was used for solving this test with 1,008 processes on 21 Skylake nodes of Inti supercomputer. Figure 12b shows the partitioned mesh with 1,008 subdomains. To complete the full phase-field simulation, 865 total solves of the linear system were performed in less than 145 minutes (approximately two and a half hours). Considering that our solver possesses linear scaling characteristics, if this problem was solved on a sequential computer, hypothetically it would have taken $(145 \times 1008)/(60 \times 24) \approx 101$ days to solve. This highlights the importance of parallel



(a) force-displacement plot.

(b) vertical displacement field u_2.

(c) damage-field d.

Figure 13: results of the three-dimensional cracking of the perforated medium. Within the truncated visualizations (right ones in (b) and (c)) the spherical hole wire-frame meshes are colored according to the vertical displacement field u_2.

computations with effective preconditioning for solving large-scale realistic problems of fracture mechanics in reasonable times. Additionally, following the timing results from previous subsections, it is reasonable to say that CG setup with Jacobi and block Jacobi preconditioners would have taken more time and iterations to solve this particular test in comparison to the adopted AMG preconditioner.

Results. Without going into details of obtained results, briefly, the crack propagates from the notch to the right side of the specimen along the x axis. As a result, by the end of the simulation, the specimen is split into two halves. Figure 13 presents the displacement and the damage-fields after the crack is fully developed. Alongside these fields, the load-displacement plot of the top surface reaction force versus applied displacement has also been presented in fig. 13a. Typical tensile cracking behavior (mode I cracking) can be inferred from the curve, the reaction force peaks to a maximum value followed by rapid drop caused due to fast crack propagation. Notice from the displacement field (fig. 13b), for the whole upper half of the specimen $u_2 \approx 1 \cdot 10^{-3}$ (which is the imposed incremental displacement), while the lower half of the specimen remains fixed $u_2 = 0$. Complementary to this result, the damage-field shown in fig. 13c rightly suggests that the specimen is fully cracked. As such, we observe a free-body motion for the upper detached part of the specimen while the lower part remains static.

To highlight non-planar crack development, which was expected in this configuration, fig. 14 depicts the damage-field and crack surface development for three phases of the simulation, d is filtered for values close to unity in order to clearly identify the represented pseudo-crack. The color map in fig. 14-left helps to analyze the vertical non-planar deviation of this pseudo-crack (fracture deviation y_{d-d_0}). On the derived cracking surface, at a particular mesh node ($P = P(x, y, z)$), the fracture deviation $y_{d-d_0} \in \mathbb{R}$ is simply obtained by $y_{d-d_0} = y - 30$ mm. Results from fig. 14-left suggests that due to the presence of holes, the crack deviates

(compared to a planar crack) by approximately half a centimeter at certain places. As clearly illustrated by traversal cross-sections of fig. 14-right, despite what is obtained in the previous case, the damage band is no more "planar", but deviates from the median plane due to the interaction with holes. Holes attract the crack due to local stress concentrations, thus leading to a quite realistic crack profile. Being able to trace such complex cracking trajectory infers robustness of our monolithic vectorial finite element procedure for solving the hybrid phase-field model.

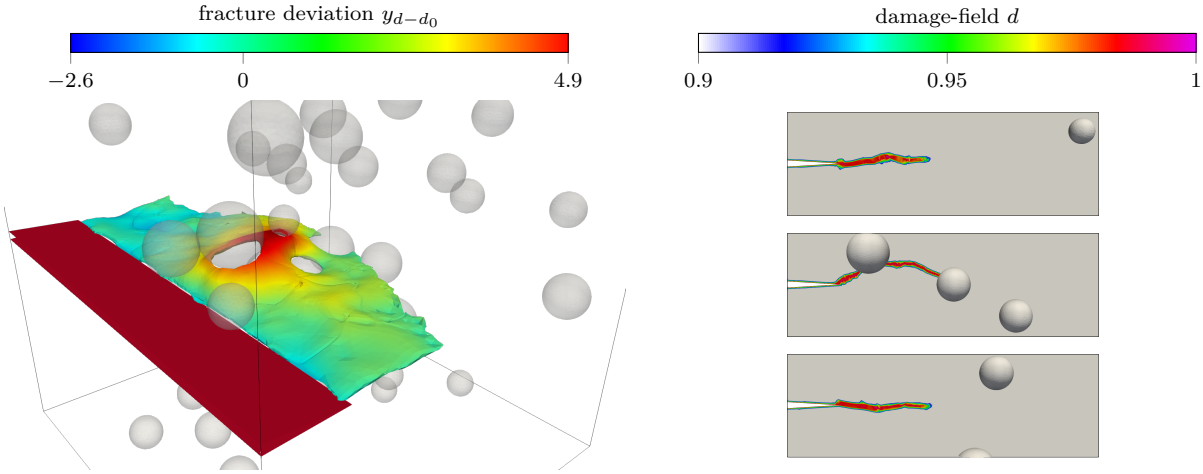
6. Conclusions

In the current paper, we presented a monolithic finite element scheme of setting up the hybrid phase-field model for solving large-scale brittle fracture mechanics problems with millions of unknowns. To achieve a monolithic scheme, the combined discretization of the coupled damage-field equation and the augmented linear momentum equation was handled by employing the vectorial finite elements. All the finite element unknowns (displacement and damage fields) were consequently approximated via a single vectorial test and trial functions. Among others, one key advantage of using the vectorial finite element discretization for the phase-field model is that it is straightforward to parallelize via domain-decomposition method. As such in this study, we used distributed-memory computing for solving problems with extremely fine meshes which are a prerequisite for standard brittle fracture mechanics problems. Moreover, this study shows that when the vectorial finite element discretization is combined with the Picard iterations for handling nonlinearity posed by the coupled equations of the phase-field model, it leads to a single positive definite symmetric matrix system. This allows the use of Krylov subspace-based conjugate gradient method for solving (memory efficiently) these linear systems.

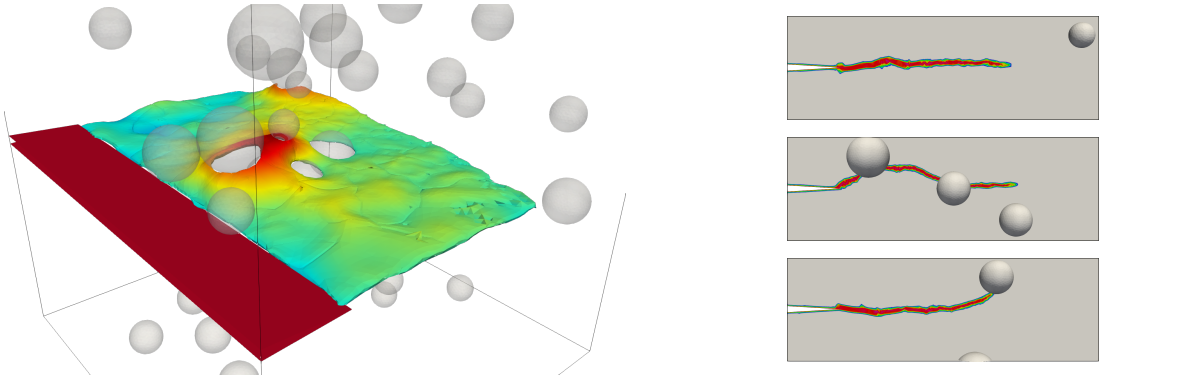
To improve convergence rates, consequently time to solution, of the conjugate gradient method applied to crack propagation problems with millions of unknowns, different preconditioning strategies were compared, analyzed, tuned, and discussed. Our analysis shows that the conjugate gradient method preconditioned with the tuned algebraic multigrid preconditioner is a robust and efficient scheme for solving the monolithic linear systems provided by the vectorial finite element discretization of the hybrid phase-field model for large-scale fracture mechanics problems.

By performing the condition number analysis on the monolithic linear systems for the brittle fracture mechanics problems, it was shown that crack initiation (during the phase-field simulations) causes the condition number to deteriorate (rise exponentially). This was consequently shown to impact the convergence behavior of the conjugate gradient solver, the iterations to converge grew drastically as the condition number to rose in the cracking zone. As such, the linear systems obtained for the hybrid phase-field problem within the cracking zone become more and more difficult to solve as the crack starts to move. Our numerical experiments prove that, in comparison to the conjugate gradient preconditioned with the Jacobi or block Jacobi preconditioners, the conjugate gradient preconditioned with the tuned algebraic multigrid preconditioner controls the rise in condition number during crack initiations. Hence, this latter preconditioner outruns the other two preconditioning options in the aspect of ease of solving a brittle fracture mechanics problem with the hybrid phase-field method. Further, numerical experiments also revealed that this tuned algebraic multigrid preconditioner remains least sensitive to mesh refinements. Considering that brittle fracture problems have a prerequisite of extremely fine meshing for replicating the cracking physics accurately, this quality makes the tune algebraic multigrid preconditioner an excellent choice over the Jacobi or block Jacobi preconditioners for the conjugate gradient method.

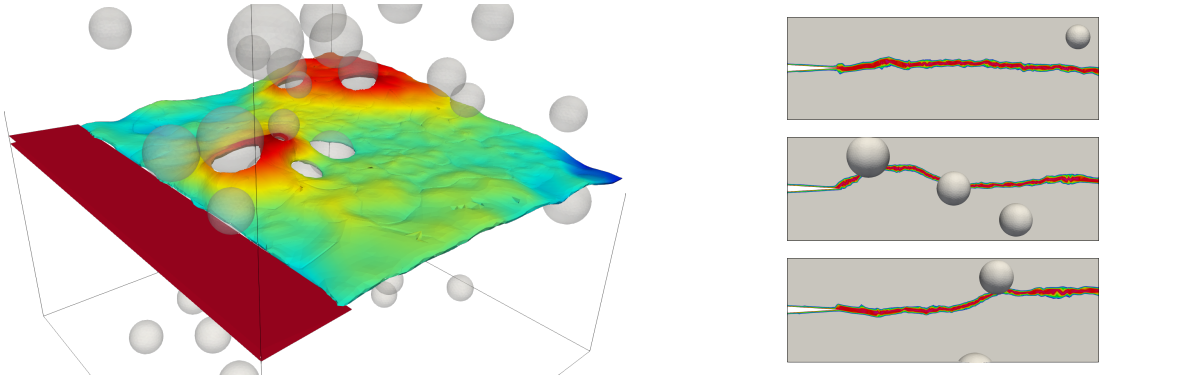
Results from two large-scale hybrid phase-field numerical simulations (with ~ 32 and ~ 81 million unknowns) performed for three-dimensional brittle fracture mechanics reveal that timing-wise, the tuned conjugate gradient algebraic multigrid preconditioner is approximately two (resp. 1.3) times faster than the conjugate gradient method setup with the Jacobi (resp. block Jacobi) preconditioner. Further, these large-scale tests were also used to show that the tuned algebraic multigrid preconditioner has the highest residual reduction rate (quality of rapidly decreasing the residual $||\mathbf{Ax} - \mathbf{b}||$ to reach faster convergence). It was equally



(a) pseudo time-step 256 ($\bar{u}_2 = 2.56 \times 10^{-1}$mm).



(b) pseudo time-step 261 ($\bar{u}_2 = 2.61 \times 10^{-1}$mm).



(c) pseudo time-step 266 ($\bar{u}_2 = 2.66 \times 10^{-1}$mm).

Figure 14: results depicting crack propagation within the three-dimensional perforated medium test. Left: truncated damage-field with fracture deviation color map; Right: different damage-field cross-sections (xy-planes) within the bulk of the material. For each (a), (b), and (c) the three xy-planes — top, middle, and bottom — are defined by top: $(x, y : 20 > y > 40, z : z = 10)$, middle: $(x, y : 20 > y > 40, z : z = 30)$, and bottom: $(x, y : 20 > y > 40, z : z = 50)$.

revealed that the algebraic multigrid preconditioner alongside the block Jacobi and Jacobi preconditioners possess excellent parallel strong scaling characteristics (more than 90 % efficiency) when these are applied to solve the large-scale phase-field fracture problems which involve linear systems with millions of degrees of freedom. However, timing-wise, the tuned algebraic multigrid preconditioner always remains the fastest option.

As an application case, a three-dimensional numerical simulation mimicking perforated media cracking which involved more than sixty-four million unknowns was "easily" solved by employing the vectorial finite element scheme. Notably, these are scale which would have otherwise not been possible to achieve if the tune preconditioning options for the conjugate gradient method was not used, which efficiently uses the high-performance computing resources.

Acknowledgments

The authors acknowledge the financial support of the Cross-Disciplinary Program on Numerical Simulation of CEA, the French Alternative Energies and Atomic Energy Commission. We thank TGCC-CEA, Bruyères-Le-Châtel, France, for providing us compute time on the Inti supercomputer. We also thank Dr. Pierre Jolivet from IRIT, CNRS, Toulouse, France, for fruitful discussions on preconditioning/parallelization. The authors declare no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

J. Ahrens, B. Geveci, C. Law, C. Hansen, and C. Johnson. ParaView: An end-user tool for large-data visualization. *The Visualization Handbook*, 717, 2005.

R. Alessi, J.-J. Marigo, and S. Vidoli. Gradient damage models coupled with plasticity: Variational formulation and main properties. *Mechanics of Materials*, 80:351–367, 2015.

J. Alexander and J. A. Yorke. The homotopy continuation method: numerically implementable topological procedures. *Transactions of the American Mathematical Society*, 242:271–284, 1978.

M. Ambati, T. Gerasimov, and L. D. Lorenzis. A review on phase-field models of brittle fracture and a new fast hybrid formulation. *Computational Mechanics*, 55(2):383–405, 2014.

M. Ambati, T. Gerasimov, and L. De Lorenzis. Phase-field modeling of ductile fracture. *Computational Mechanics*, 55(5): 1017–1040, 2015.

M. Ambati, R. Kruse, and L. De Lorenzis. A phase-field model for ductile fracture at finite strains and its experimental verification. *Computational Mechanics*, 57(1):149–167, 2016.

H. Amor, J.-J. Marigo, and C. Maurini. Regularized formulation of the variational brittle fracture with unilateral contact: Numerical experiments. *Journal of the Mechanics and Physics of Solids*, 57(8):1209–1229, 2009.

M. A. Badri, P. Jolivet, B. Rousseau, S. Le Corre, H. Digonnet, and Y. Favennec. Vectorial finite elements for solving the radiative transfer equation. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 212:59–74, 2018.

M. A. Badri, P. Jolivet, B. Rousseau, and Y. Favennec. Preconditioned krylov subspace methods for solving radiative transfer problems with scattering and reflection. *Computers & Mathematics with Applications*, 77(6):1453–1465, 2019.

A. H. Baker, R. D. Falgout, T. V. Kolev, and U. M. Yang. Multigrid smoothers for ultraparallel computing. *SIAM Journal on Scientific Computing*, 33(5):2864–2887, 2011a.

A. H. Baker, R. D. Falgout, T. V. Kolev, and U. M. Yang. Multigrid smoothers for ultra-parallel computing: Additional theory and discussion. Technical report, Lawrence Livermore National Lab.(LLNL), Livermore, CA (United States), 2011b.

S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, D. Karpeyev, D. Kaushik, M. Knepley, D. May, L. Curfman McInnes, R. Mills, T. Munson, K. Rupp, P. Sanan, B. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc Users Manual. Technical report, Argonne National Laboratory, 2019.

A. Basermann, B. Reichel, and C. Schelthoff. Preconditioned CG methods for sparse matrices on massively parallel machines. *Parallel Computing*, 23(3):381–398, 1997.

Z. P. Bažant and M. Jirásek. Nonlocal Integral Formulations of Plasticity and Damage: Survey of Progress. *Journal of Engineering Mechanics*, 128(11):1119–1149, 2002.

M. Benzi. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics*, 182(2):418–477, 2002.

C. Bilgen, A. Kopaničáková, R. Krause, and K. Weinberg. A phase-field approach to conchoidal fracture. *Meccanica*, 53(6): 1203–1219, 2017.

M. J. Borden, C. V. Verhoosel, M. A. Scott, T. J. Hughes, and C. M. Landis. A phase-field description of dynamic brittle fracture. *Computer Methods in Applied Mechanics and Engineering*, 217:77–95, 2012.

M. J. Borden, T. J. Hughes, C. M. Landis, A. Anvari, and I. J. Lee. Phase-field formulation for ductile fracture. In *Advances in Computational Plasticity*, pages 45–70. Springer, 2018.

B. Bourdin. Numerical implementation of the variational formulation for quasi-static brittle fracture. *Interfaces and Free Boundaries*, 9(3):411–430, 2007.

B. Bourdin, G. Francfort, and J.-J. Marigo. Numerical experiments in revisited brittle fracture. *Journal of the Mechanics and Physics of Solids*, 48(4):797–826, 2000.

B. Bourdin, G. A. Francfort, and J.-J. Marigo. The variational approach to fracture. *Journal of Elasticity*, 91(1-3):5–148, 2008.

R. de Borst and C. V. Verhoosel. Gradient damage vs phase-field approaches for fracture: Similarities and differences. *Computer Methods in Applied Mechanics and Engineering*, 312:78–94, 2016.

D. H. Doan, T. Q. Bui, N. D. Duc, and K. Fushinobu. Hybrid phase field simulation of dynamic crack propagation in functionally graded glass-filled epoxy. *Composites Part B: Engineering*, 99:266–276, 2016.

T. Dupont, R. P. Kendall, and H. Rachford, Jr. An approximate factorization procedure for solving self-adjoint elliptic difference equations. *SIAM Journal on Numerical Analysis*, 5(3):559–573, 1968.

P. E. Farrell and C. Maurini. Linear and nonlinear solvers for variational phase-field models of brittle fracture. *International Journal for Numerical Methods in Engineering*, 109:648–667, 2017.

G. A. Francfort and J.-J. Marigo. Revisiting brittle fracture as an energy minimization problem. *Journal of the Mechanics and Physics of Solids*, 46(8):1319–1342, 1998.

M. Frémond and B. Nedjar. Damage, gradient of damage and principle of virtual power. *International Journal of Solids and Structures*, 33(8):1083–1103, 1996.

T. Gerasimov and L. De Lorenzis. A line search assisted monolithic approach for phase-field computing of brittle fracture. *Computer Methods in Applied Mechanics and Engineering*, 312:276–303, 2016.

C. Geuzaine and J. F. Remacle. Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities. *International Journal for Numerical Methods in Engineering*, 79(11):1309–1331, 2009.

C. Giry, F. Dufour, and J. Mazars. Stress-based nonlocal damage model. *International Journal of Solids and Structures*, 48 (25):3431–3443, 2011.

A. Grama, V. Kumar, A. Gupta, and G. Karypis. *Introduction to parallel computing*. Pearson Education, 2003.

W. D. Gropp, E. Lusk, and A. Skjellum. *Using MPI: portable parallel programming with the message-passing interface*, volume 1. MIT press, 1999.

F. Hecht. New development in FreeFem++. *Journal of Numerical Mathematics*, 20(3-4):251–266, 2012.
T. Heister and T. Wick. Parallel solution, adaptivity, computational convergence, and open-source code of 2D and 3D pressurized phase-field fracture problems. In *PAMM: Proceedings in Applied Mathematics and Mechanics*, volume 18, page e201800353. Wiley Online Library, 2018.
M. R. Hestenes and E. Stiefel. *Methods of conjugate gradients for solving linear systems*, volume 49. NBS Washington, DC, 1952.
Hirshikesh, S. Natarajan, and R. K. Annabattula. A FEniCS implementation of the phase field method for quasi-static brittle fracture. *Frontiers of Structural and Civil Engineering*, 13(2):380–396, 2018.
Hirshikesh, A. Pramod, R. Annabattula, E. Ooi, C. Song, and S. Natarajan. Adaptive phase-field modeling of brittle fracture using the scaled boundary finite element method. *Computer Methods in Applied Mechanics and Engineering*, 355:284–307, 2019.
M. Hofacker and C. Miehe. Continuum phase field modeling of dynamic fracture: variational principles and staggered FE implementation. *International Journal of Fracture*, 178(1-2):113–129, 2012.
H. Jeong, S. Signetti, T.-S. Han, and S. Ryu. Phase field modeling of crack propagation under combined shear and tensile loading with hybrid formulation. *Computational Materials Science*, 155:483–492, 2018.
D. Jodlbauer, U. Langer, and T. Wick. Matrix-free multigrid solvers for phase-field fracture problems. *arXiv e-prints*, 54: arXiv:1902.08112, 2019.
J. E. Jones and P. S. Vassilevski. AMGe based on element agglomeration. *SIAM Journal on Scientific Computing*, 23(1): 109–133, 2001.
G. Karypis, K. Schloegel, and V. Kumar. Parmetis: Parallel graph partitioning and sparse matrix ordering library. *Version 1.0, Dept. of Computer Science, University of Minnesota*, page 22, 1997.
D. S. Kershaw. The incomplete Cholesky–conjugate gradient method for the iterative solution of systems of linear equations. *Journal of Computational Physics*, 26(1):43–65, 1978.
A. Kopaničáková and R. Krause. A recursive multilevel trust region method with application to fully monolithic phase-field models of brittle fracture. *Computer Methods in Applied Mechanics and Engineering*, 360:112720, 2020.
C. Kuhn and R. Müller. Phase field simulation of thermomechanical fracture. In *PAMM: Proceedings in Applied Mathematics and Mechanics*, volume 9, pages 191–192. Wiley Online Library, 2009.
G. Liu, Q. Li, M. A. Msekh, and Z. Zuo. Abaqus implementation of monolithic and staggered schemes for quasi-static and dynamic fracture phase-field model. *Computational Materials Science*, 121:35–47, 2016.
J. Málek and Z. Strakos. *Preconditioning and the conjugate gradient method in the context of solving PDEs*, volume 1. SIAM, 2014.
T. K. Mandal, V. P. Nguyen, and A. Heidarpour. Phase field and gradient enhanced damage models for quasi-brittle failure: A numerical comparative study. *Engineering Fracture Mechanics*, 207:48–67, 2019.
S. May, J. Vignollet, and R. De Borst. A numerical assessment of phase-field models for brittle and cohesive fracture: γ-convergence and stress oscillations. *European Journal of Mechanics-A/Solids*, 52:72–84, 2015.
C. Miehe and L.-M. Schänzel. Phase field modeling of fracture in rubbery polymers. Part I: Finite elasticity coupled with brittle failure. *Journal of the Mechanics and Physics of Solids*, 65:93–113, 2014.
C. Miehe, M. Hofacker, and F. Welschinger. A phase field model for rate-independent crack propagation: Robust algorithmic implementation based on operator splits. *Computer Methods in Applied Mechanics and Engineering*, 199(45-48):2765–2778, 2010a.
C. Miehe, F. Welschinger, and M. Hofacker. Thermodynamically consistent phase-field models of fracture: Variational principles and multi-field FE implementations. *International Journal for Numerical Methods in Engineering*, 83(10):1273–1311, 2010b.
C. Miehe, M. Hofacker, L.-M. Schänzel, and F. Aldakheel. Phase field modeling of fracture in multi-physics problems. Part II. Coupled brittle-to-ductile failure criteria and crack propagation in thermo-elastic–plastic solids. *Computer Methods in Applied Mechanics and Engineering*, 294:486–522, 2015.
N. Moës, C. Stolz, P.-E. Bernard, and N. Chevaugeon. A level set based model for damage growth: The thick level set approach. *International Journal for Numerical Methods in Engineering*, 86(3):358–380, 2011.
G. Molnár and A. Gravouil. 2D and 3D Abaqus implementation of a robust staggered phase-field solution for modeling brittle fracture. *Finite Elements in Analysis and Design*, 130:27–38, 2017.
T. T. Nguyen, J. Yvonnet, M. Bornert, C. Chateau, K. Sab, R. Romani, and R. Le Roy. On the choice of parameters in the phase field method for simulating crack initiation with experimental validation. *International Journal of Fracture*, 197(2): 213–226, 2016.
C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.
R. Peerlings, M. Geers, R. de Borst, and W. Brekelmans. A critical comparison of nonlocal and gradient-enhanced softening continua. *International Journal of Solids and Structures*, 38(44-45):7723–7746, 2001.
K. H. Pham, K. Ravi-Chandar, and C. M. Landis. Experimental validation of a phase-field model for fracture. *International Journal of Fracture*, 205(1):83–101, 2017.
G. Pijaudier-Cabot and Z. P. Bažant. Nonlocal damage theory. *Journal of Engineering Mechanics*, 113(10):1512–1533, 1987.
G. Rastiello, C. Giry, F. Gatuingt, and R. Desmorat. From diffuse damage to strain localization from an Eikonal Non-Local (ENL) Continuum Damage model with evolving internal length. *Computer Methods in Applied Mechanics and Engineering*, 331:650–674, 2018.
J. W. Ruge and K. Stüben. Algebraic multigrid. In *Multigrid methods*, pages 73–130. SIAM, 1987.
Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

K. Stüben. A review of algebraic multigrid. In *Numerical Analysis: Historical Developments in the 20th Century*, pages 331–359. Elsevier, 2001.

P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56(3):179–196, 1996.

C. V. Verhoosel and R. de Borst. A phase-field model for cohesive fracture. *International Journal for Numerical Methods in Engineering*, 96(1):43–62, 2013.

J. Vignollet, S. May, R. De Borst, and C. V. Verhoosel. Phase-field models for brittle and cohesive fracture. *Meccanica*, 49 (11):2587–2601, 2014.

T. Wick. Modified Newton methods for solving fully monolithic phase-field quasi-static brittle fracture propagation. *Computer Methods in Applied Mechanics and Engineering*, 325:577–611, 2017.

J.-Y. Wu. A unified phase-field theory for the mechanics of damage and quasi-brittle failure. *Journal of the Mechanics and Physics of Solids*, 103:72–99, 2017.

J.-Y. Wu and V. P. Nguyen. A length scale insensitive phase-field damage model for brittle fracture. *Journal of the Mechanics and Physics of Solids*, 119:20–42, 2018.

J.-Y. Wu, V. P. Nguyen, C. T. Nguyen, D. Sutula, S. Bordas, and S. Sinaie. Phase field modeling of fracture. *Advances in Applied Mechancis: Multi-scale Theory and Computation*, 52, 2018.

J. Xu and L. Zikatanov. Algebraic multigrid methods. *Acta Numerica*, 26:591–721, 2017.