Available online at www.sciencedirect.com
ScienceDirect
Comput. Methods Appl. Mech. Engrg. 395 (2022) 114937
Computer methods in applied mechanics and engineering
www.elsevier.com/locate/cma

An adapted deflated conjugate gradient solver for robust extended/generalised finite element solutions of large scale, 3D crack propagation problems

Konstantinos Agathos[a,*], Tim Dodwell[a,b], Eleni Chatzi[c], Stéphane P.A. Bordas[d,e]

[a] College of Engineering Mathematics and Physical Sciences, University of Exeter, UK
[b] The Alan Turing Institute, UK
[c] Department of Civil, Environmental, and Geomatic Engineering, ETH Zürich, Switzerland
[d] Research Unit in Engineering Science, Luxembourg University, Luxembourg
[e] Institute of Theoretical, Applied and Computational Mechanics, Cardiff University, UK

Received 17 November 2021; received in revised form 4 February 2022; accepted 22 March 2022
Available online 28 April 2022

Abstract

An adapted deflation preconditioner is employed to accelerate the solution of linear systems resulting from the discretisation of fracture mechanics problems with well-conditioned extended/generalised finite elements. The deflation space typically used for linear elasticity problems is enriched with additional vectors, accounting for the enrichment functions used, thus effectively removing low frequency components of the error. To further improve performance, deflation is combined, in a multiplicative way, with a block-Jacobi preconditioner, which removes high frequency components of the error as well as near-linear dependencies introduced by enrichment. The resulting scheme is tested on a series of non-planar crack propagation problems and compared to alternative linear solvers in terms of performance.

Keywords: XFEM; GFEM; Deflated CG; Multigrid; Domain decomposition; Crack propagation

1. Introduction

The eXtended and Generalised Finite Element Methods (XFEM and GFEM respectively) [1,2], form a class of multiscale finite element methods that have significantly contributed to the increase of the level of automation and computational efficiency of crack propagation simulations. These approaches substantially reduce, or even eliminate, the need for re-meshing, allowing small-scale features to be captured efficiently and accurately at a coarse, marco-scale representation. This is accomplished by means of Partition of Unity (PoU) enrichment [3], whereby polynomial approximation spaces of the Finite Element Method (FEM) are supplemented with appropriate additional enrichment functions. Importantly, these enrichment functions encompass known features or micro-scale information of the system. In fracture mechanics, features such as discontinuities and singularities can be represented independently

* Corresponding author.
E-mail address: k.agathos@exeter.ac.uk (K. Agathos).

https://doi.org/10.1016/j.cma.2022.114937

of the underlying Finite Element (FE) mesh, offering significant computational advantages. This has contributed to the popularity of the methods, and has led to their use in a vast array of applications, now documented in numerous review papers [4–7].

The advantages of the method do not come without a price. Among other challenges, the method can produce ill-conditioned systems of equations, as a result of enrichment functions being almost linearly dependent with each other or with the FE basis. In the case of large scale, three dimensional simulations, for which parallel iterative solvers are a necessity, this ill-conditioning, without special attention, leads to stagnation (often complete failure) of the solver. To mitigate these issues, several approaches have been proposed, seeking to resolve the near-linear dependencies introduced by the enrichment functions. For the discontinuous functions used to represent solutions along crack faces [1], strategies include the elimination of the pathological enriched degrees of freedom [8], and particular stabilisation techniques either at the element [9] or system level [10]. For enrichment with asymptotic functions, typically used to represent singularities occurring at crack tips/fronts, alternative enrichment functions have been proposed [11,12], alongside enrichment modification strategies, such as degree of freedom gathering [13–15], the so-called stable GFEM [16–18] and orthogonalisation [19,20].

Most of the above strategies seek to improve the conditioning of the resultant system of equations by acting directly on the formulation of the XFEM/GFEM methodology and the choice, selection or representation of the enrichment functions. Stepping back from this specific XFEM/GFEM setting, to the more general challenge of parallel iterative solvers for ill-conditioned systems of equations, a significant amount of work exists on robust preconditioners for parallel iterative solvers. For a general ill-conditioned system $\mathbf{Ku} = \mathbf{f}$, the idea of (left) preconditioning is to determine an operator $\mathbf{M}^{-1}$ for which $\mathbf{M}^{-1}\mathbf{K}$ is well-conditioned and $\mathbf{M}^{-1}$ is efficient to construct; allowing a parallel iterative solver, e.g. Conjujate Gradient (pCG), to converge quickly. The most widely used preconditioners for iterative solvers in both commercial and scientific FE codes are Algebraic Multigrid (AMG) methods [21]. They have demonstrated excellent scalability for a broad class of problems over thousands of processors, and have the advantage of operating exclusively on the matrix equations, so that they can be applied as a 'black-box'. As a preconditioner, AMG constructs the matrix $\mathbf{M}$ by repeatedly coarsening the full matrix $\mathbf{K}$ through recursive aggregation over the degrees of freedom. The aggregation process is algebraic and based on the fact that the solution at two neighbouring nodes will be similar if they are 'strongly connected'. The success of an AMG preconditioner depends on this aggregation process. In the vicinity of a crack, and with large simulation divided into connected subdomains, this aggregation can be difficult to design properly. A widely used alternative is additive Schwarz based preconditioners. Single level versions utilise independent solves of the local Dirichlet problem. These are computed in parallel, using robust direct solvers, with the aim of resolving the ill-conditioning of the problem at a local length scale. These can be particularly effective, yet are known to lack robustness in the case of large numbers of subdomains or with respect to large contrast in coefficients. A problem ever present in crack propagation problems. The problems in this case are linked to the fact that the localised solves do not resolve the global low energy modes of the system, meaning the ill-conditioning persists. To overcome this issue, preconditioners can be augmented with an additional step, which involves either deflation methods [22], or second level coarse-solvers. In the former case, the deflection vectors/coarse basis for the problem can be identified through an analytical solution (e.g. ZEM [22]), while in the latter, optimal approximations the global low energy modes are identified via local eigenproblems, see for example methods involving Generalised Eigenproblems in the Overlap GenEO [23–25]. It is interesting to note that construction of robust coarse spaces, via GenEO type methods bears strong ties to Generalised Finite Element Methods, as studied by Ma et al. [26]. In some respect, this optimal basis is an optimal presentation of enrichment functions, which does not suffer from near-linear dependencies. Yet with this robustness comes the additional computational burden associated to solution of the local eigenvalue problems, which in many cases is an overkill.

Several techniques have also been proposed to improve the performance of iterative solvers for crack propagation problems discretised using the XFEM/GFEM. The preconditioners proposed by Béchet et al. [27] and Menk and Bordas [28] employ a Cholesky decomposition to remove near-linear dependencies introduced by enrichment. The application of AMG preconditioners to systems produced by the XFEM/GFEM has also been proposed in several works. However, as shown, for instance, in Berger-Vergiat et al. [29], it cannot be performed directly since it results in sub-optimal performance. This is attributed to the fact that discontinuities present along crack faces are not accounted for in the construction of prolongation/restriction operators. To overcome this limitation, some works [29–31] employ Schwartz preconditioners to decompose the problem domain into a healthy and a cracked part, for

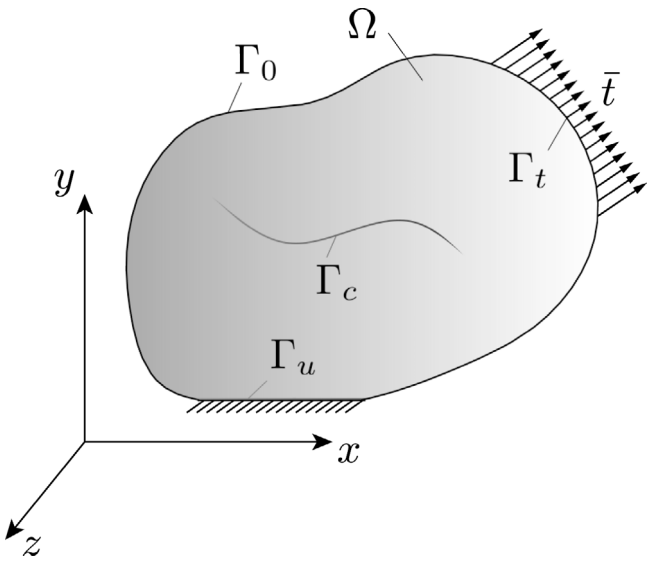


Fig. 1. Cracked body and boundary conditions.

which different solvers, such as AMG or LU, can be used. It should be noted that this type of decomposition has also been exploited for other purposes in the XFEM/GFEM literature, for instance in the seminal work of Bordas and Moran [32,33] it is used to enable the combination of commercial and research codes, while in the global–local GFEM [34] it is employed to construct numerically derived enrichment functions. Other works [35,36] modify prolongation/restriction operators to account for discontinuities along crack surfaces. In a more recent work [37], the block structure of linear systems produced by the XFEM/GFEM is exploited to derive a modified system, where AMG and reanalysis techniques can be applied to accelerate the solution. This block structure is also exploited in the work of Fillmore et al. [38] through specialised block-Jacobi and block-Gauss–Seidel preconditioners.

In this work, a preconditioner based on adapted deflation [39] is proposed for crack propagation problems, discretised with the XFEM/GFEM. The deflation space is constructed based on domain partitioning [40], while additional deflation vectors are introduced to account for the discontinuities introduced by the cracks. The approach is combined with a block-Jacobi preconditioner, which offers several advantages such as high efficiency, straightforward parallelisation, and efficient update for crack propagation problems. Whilst the main target application is non-planar crack propagation, the approach would also be suitable for other types of problems, for instance inverse problems [41], where repeated solutions of linear systems with slight variations, introduced by different crack configurations, are required.

The remainder of the paper is structured as follows: after a brief presentation of the elasticity equations and adopted discretisation scheme in Section 2, the different components of the proposed preconditioner are described in Section 3, and tested in Section 4 through a series of examples involving crack propagation in models of components of increasing geometrical complexity and size. Finally, in Section 5, the results are summarised and conclusions are drawn.

2. Problem statement and discretisation

2.1. Weak form

For linear elastic fracture mechanics, we consider the linear elasticity problem for a cracked domain Ω, whose boundary can be decomposed, as illustrated in Fig. 1, in a part where free surface conditions apply (Γ_0), a part where displacements $\bar{\boldsymbol{u}}$ are imposed as Dirichlet conditions (Γ_u), a part where surface tractions $\bar{\boldsymbol{t}}$ are applied (Γ_t) and the crack faces Γ_c, where free surface conditions also apply. The weak form of the problem, assuming a linear elastic constitutive equation, can be written as:

Find $\boldsymbol{u} \in \mathcal{U}$ such that $\forall \boldsymbol{v} \in \mathcal{V}^0$:

$$\int_{\Omega} \boldsymbol{\epsilon}(\boldsymbol{u}) : \boldsymbol{D} : \boldsymbol{\epsilon}(\boldsymbol{v})\, d\Omega = \int_{\Omega} \boldsymbol{b} \cdot \boldsymbol{v}\, d\Omega + \int_{\Gamma_t} \bar{\boldsymbol{t}} \cdot \boldsymbol{v}\, d\Gamma \tag{1}$$

with :

$$\mathcal{U} = \left\{ \boldsymbol{u} | \boldsymbol{u} \in \left(H^1(\Omega)\right)^3, \boldsymbol{u} = \bar{\boldsymbol{u}} \text{ on } \Gamma_u \right\} \tag{2}$$

and

$$\mathcal{V}^0 = \left\{ \boldsymbol{v} | \boldsymbol{v} \in \left(H^1(\Omega)\right)^3, \boldsymbol{v} = \boldsymbol{0} \text{ on } \Gamma_u \right\} \tag{3}$$

In Eq. (1), $\boldsymbol{u}$ is the displacement field, $\boldsymbol{v}$ is the virtual displacement field, $\boldsymbol{D}$ is the Hooke tensor, $\boldsymbol{\sigma}$ is the Cauchy stress tensor and $\boldsymbol{\epsilon}$ is the linear strain tensor.

2.2. Crack representation

As typically done within the XFEM/GFEM framework, cracks are implicitly represented using the level set method. To this end, two signed distance functions are defined:

- The normal level set ϕ, defined as the signed distance from the crack surface:

$$\phi(\boldsymbol{x}) = \min_{\bar{\boldsymbol{x}} \in \Gamma_c} \|\boldsymbol{x} - \bar{\boldsymbol{x}}\| \operatorname{sign}\left(\boldsymbol{n}^+ \cdot (\boldsymbol{x} - \bar{\boldsymbol{x}})\right) \tag{4}$$

  where $\boldsymbol{n}^+$ is the outward normal to the crack surface and $\operatorname{sign}(\square)$ is the sign function.
- The tangential level set ψ, defined as the signed distance function satisfying the conditions:

$$\nabla\phi \cdot \nabla\psi = 0 \tag{5a}$$

$$\left.\begin{array}{l} \phi(\boldsymbol{x}) = 0 \\ \psi(\boldsymbol{x}) = 0 \end{array}\right\} \forall \boldsymbol{x} \in \Gamma_f \tag{5b}$$

  where Γ_f are the crack front/tips.

Based on these functions, a polar coordinate system, with its origin at the crack front, can be defined:

$$r = \sqrt{\phi^2 + \psi^2}, \qquad \theta = \arctan\left(\frac{\phi}{\psi}\right) \tag{6}$$

In general, the integration of several evolution equations is required to update level set descriptions of propagating cracks [42], a procedure that can be both complex and computationally demanding. Herein, the vector level set method [43–45] is employed, which allows to update the crack description using only geometrical operations, thus simplifying the process.

2.3. Discretisation

The weak form of Eq. (1) is discretised using the standard XFEM displacement approximation with shifted jump enrichment functions and quasi-orthogonalised tip enrichment functions:

$$\boldsymbol{u}^h(\boldsymbol{x}) = \sum_{\forall I \in \mathcal{N}} N_I(\boldsymbol{x})\, \boldsymbol{u}_I + \sum_{\forall J \in \mathcal{N}^j} N_J(\boldsymbol{x})\, H_J(\boldsymbol{x})\, \boldsymbol{b}_J + \sum_{\forall T \in \mathcal{N}^t} \sum_{\forall j} N_T(\boldsymbol{x})\, \bar{F}_{Tj}(\boldsymbol{x})\, \boldsymbol{c}_{Tj} \tag{7}$$

where N_I are the FE interpolation functions, $\boldsymbol{u}_I$ are FE degrees of freedom (dofs), and $\boldsymbol{b}_J$, $\boldsymbol{c}_{TJ}$ are the enriched degrees of freedom. The shifted jump enrichment functions are defined as:

$$H_J(\boldsymbol{x}) = H(\boldsymbol{x}) - H(\boldsymbol{x}_J) \tag{8}$$

where $\boldsymbol{x}_J$ are the nodal coordinates of node J and:

$$H(\boldsymbol{x}) = H(\phi(\boldsymbol{x})) = \begin{cases} 1 & \text{for } \phi \geq 0 \\ -1 & \text{for } \phi < 0 \end{cases} \tag{9}$$

The quasi-orthogonalised tip enrichment functions are defined as in one of the previous works of the authoring team [19]:

$$\begin{aligned}\bar{\boldsymbol{F}}_I(r,\theta) = \{ & \sqrt{r}\sin\frac{\theta}{2} - \sqrt{r_I}\sin\frac{\theta_I}{2}, \\ & \sqrt{r}\cos\frac{\theta}{2} - \sqrt{r_I}\cos\frac{\theta_I}{2}, \\ & 2\sqrt{r}\cos\left(\frac{\theta}{2}+\theta_I\right)\sin\left(\frac{\theta-\theta_I}{2}\right), \\ & -2\sqrt{r}\sin^2\left(\frac{\theta-\theta_I}{2}\right)\left[c_{I34}\cos\left(\frac{\theta}{2}+\theta_I\right)+\sin\left(\frac{\theta}{2}+\theta_I\right)\right]\Big\}\end{aligned} \tag{10}$$

where r_I, θ_I are the nodal values of the radius r and angle θ and:

$$c_{I34} = -\frac{7\sin(\theta_I) + 11\sin(2\theta_I) + 65\sin(3\theta_I)}{36\left[1+\cos(3\theta_I)\right]} \tag{11}$$

It should be noted that index j in Eq. (7) refers to the different components of $\bar{\boldsymbol{F}}_I$

The nodal sets in Eq. (7) are defined as:

$\mathcal{N}$ is the set of all nodes in the FE mesh.

$\mathcal{N}^j$ is the set of jump enriched nodes. This nodal set includes all nodes whose support is split in two by the crack.

$\mathcal{N}^t$ is the set of tip enriched nodes. This nodal set includes all nodes whose support includes the crack front or nodes with $r_I \leq r_e$, with r_e a user defined enrichment radius.

The above choice of enrichment functions, combined with linear tetrahedral elements, was shown to lead well conditioned system matrices [19] while recovering linear convergence rates and is employed herein due to its simplicity. However, more accurate, well conditioned schemes are also available [18,46], and could be used in conjunction to the preconditioner presented in what follows.

2.4. Linear system of equations

Substituting the displacement approximation of Eq. (7) into the weak form of Eq. (1) and carrying out the integrations, yields a system of equations with the following block structure:

$$\underbrace{\begin{bmatrix}\boldsymbol{K}_{\boldsymbol{FF}} & \boldsymbol{K}_{\boldsymbol{FX}} \\ \boldsymbol{K}_{\boldsymbol{FX}}^T & \boldsymbol{K}_{\boldsymbol{XX}}\end{bmatrix}}_{\boldsymbol{K}} \underbrace{\begin{bmatrix}\boldsymbol{u}_{\boldsymbol{F}} \\ \boldsymbol{u}_{\boldsymbol{X}}\end{bmatrix}}_{\boldsymbol{u}} = \underbrace{\begin{bmatrix}\boldsymbol{f}_{\boldsymbol{F}} \\ \boldsymbol{f}_{\boldsymbol{X}}\end{bmatrix}}_{\boldsymbol{f}} \tag{12}$$

where $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ is the stiffness matrix, $\boldsymbol{u} \in \mathbb{R}^n$ is the vector of unknowns, $\boldsymbol{f} \in \mathbb{R}^n$ is the load vector, and n is the total number of unknowns. Moreover, $\boldsymbol{u}_{\boldsymbol{F}} \in \mathbb{R}^{n_F}$ contains all the standard degrees of freedom, corresponding to $\boldsymbol{u}_I$ from Eq. (7), $\boldsymbol{u}_{\boldsymbol{X}} \in \mathbb{R}^{n_X}$ contains all the enriched degrees of freedom, corresponding to $\boldsymbol{b}_J$ and $\boldsymbol{c}_{Tj}$ from Eq. (7), $\boldsymbol{K}_{\boldsymbol{FF}} \in \mathbb{R}^{n_F\times n_F}$, $\boldsymbol{K}_{\boldsymbol{FX}} \in \mathbb{R}^{n_F\times n_X}$, $\boldsymbol{K}_{\boldsymbol{X}} \in \mathbb{R}^{n_X\times n_X}$ are the parts of the stiffness matrix corresponding to the standard and enriched degrees of freedom, as well as their interaction, $\boldsymbol{f}_{\boldsymbol{F}} \in \mathbb{R}^{n_F}$, $\boldsymbol{f}_{\boldsymbol{X}} \in \mathbb{R}^{n_X}$ are the parts of the load vector corresponding to the standard and enriched degrees of freedom, n_F is the number of standard degrees of freedom, n_X is the number of enriched degrees of freedom. From the above it should be obvious that $n = n_F + n_X$.

While Eq. (12) does not necessarily reflect the order in which equations are stored in practice, it helps gain some insight regarding the structure of the resulting systems. For instance, $\boldsymbol{K}_{\boldsymbol{FF}}$ will typically contain more than 90% of the total entries in the stiffness matrix and will remain constant during a crack propagation simulation, with only the other three blocks changing as cracks propagate and additional nodes are enriched.

3. Adapted deflation preconditioner

3.1. The conjugate gradient method

For two dimensional or small three dimensional problems, the linear system of (12) can be solved efficiently using direct solvers. However, for larger three dimensional problems, where direct solvers can be inefficient due to excessive memory requirements, iterative solvers become an attractive alternative. For symmetric systems, the most widely used alternative is the conjugate gradient method. As shown in algorithm 1, the method iteratively improves upon an initial solution $\boldsymbol{u}_0$ by adding vectors $\boldsymbol{p}_i$, which are conjugate with respect to $\boldsymbol{K}$. Since the number of iterations required to obtain an accurate solution depends on the conditioning of $\boldsymbol{K}$, a preconditioner $\boldsymbol{M}$ is typically employed, the choice of which can significantly affect the efficiency of the solver. In what follows, a deflation based preconditioner, specifically tailored for systems of the form of (12) is presented.

Algorithm 1: The conjugate gradient method.

Data: $\boldsymbol{K}$, $\boldsymbol{f}$, $\boldsymbol{u}_0$, $\boldsymbol{M}$
Result: $\boldsymbol{u}_i$
$\boldsymbol{r}_0 = \boldsymbol{f} - \boldsymbol{K}\boldsymbol{u}_0$;
$z_0 = \boldsymbol{M}^{-1}\boldsymbol{r}_0$;
$\boldsymbol{p}_0 = z_0$
for $i = 0, 1, 2, \ldots$ **do**
  $\alpha_i = \dfrac{\boldsymbol{r}_i^T z_i}{\boldsymbol{p}_i^T \boldsymbol{K}\boldsymbol{p}_i}$;
  $\boldsymbol{u}_{i+1} = \boldsymbol{u}_i + \alpha_i \boldsymbol{p}_i$
  $\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \alpha_i \boldsymbol{K}\boldsymbol{p}_i$
  $z_{i+1} = \boldsymbol{M}^{-1}\boldsymbol{r}_{i+1}$
  $\beta_i = \dfrac{\boldsymbol{r}_{i+1}^T z_{i+1}}{\boldsymbol{r}_i^T z_i}$
  $\boldsymbol{p}_{i+1} = z_{i+1} + \beta_i \boldsymbol{p}_i$
end

3.2. Deflation

Given a basis for a subspace of $\mathbb{R}^{n\times n}$, deflation [47,48] aims at accelerating the iterative solution of linear systems, such as the one of Eq. (12), by solving a modified system, of the form:

$$\hat{\boldsymbol{K}}\hat{\boldsymbol{u}} = \hat{\boldsymbol{f}} \tag{13}$$

where:

$$\hat{\boldsymbol{K}} = \boldsymbol{P}\boldsymbol{K} \tag{14a}$$

$$\hat{\boldsymbol{f}} = \boldsymbol{P}\boldsymbol{f} \tag{14b}$$

$$\boldsymbol{P} = \boldsymbol{I} - \boldsymbol{K}\boldsymbol{Q} \tag{14c}$$

$$\boldsymbol{Q} = \boldsymbol{W}\left(\boldsymbol{W}^T\boldsymbol{K}\boldsymbol{W}\right)^{-1}\boldsymbol{W}^T \tag{14d}$$

with $\boldsymbol{W} \in \mathbb{R}^{n\times k}$ a matrix containing a basis of a subspace of $\mathbb{R}^n$, with $k \ll n$. It can be shown that if $\boldsymbol{W}$ is full rank, and $\boldsymbol{K}$ is symmetric positive definite (SPD), the system of Eq. (13) can be solved using the conjugate gradient (CG) method, while the solution for the original system can be recovered as:

$$\boldsymbol{u} = \boldsymbol{Q}\boldsymbol{f} + \boldsymbol{P}^T\hat{\boldsymbol{u}} \tag{15}$$

In Eq. (13), $\boldsymbol{P}$ projects the subspace defined by $\boldsymbol{W}$ out of the original system. While it can be shown that any full rank matrix will result in some reduction of the number of iterations required by the CG method, typically, columns of $\boldsymbol{W}$ are selected to approximate eigenvectors corresponding to the smallest eigenvalues of $\boldsymbol{K}$. Since the projection of Eq. (13) essentially removes the corresponding eigenvalues of $\boldsymbol{K}$, such a choice effectively reduces its condition number, resulting in a reduced number of CG iterations. More details about the construction of $\boldsymbol{W}$ for the problems considered herein are given in Section 3.3.

Deflation is also related to multi-grid methods, more specifically, $\boldsymbol{W}^T$ and $\boldsymbol{W}$ can be seen as prolongation and restriction operators respectively [49]. Then, $\left(\boldsymbol{W}^T\boldsymbol{K}\boldsymbol{W}\right)$ can be viewed as a coarse grid matrix and the application of $\boldsymbol{Q}$ to the residual during the iterative solution of the system represents a coarse grid correction, aiming at removing low frequency components of the error.

Deflation, viewed either as a projection or a coarse grid correction, can be combined to preconditioning in an additive or multiplicative way, resulting in multiple alternatives. For instance, applying deflation to a system preconditioned with an arbitrary SPD preconditioner $\boldsymbol{M}^{-1}$ results in the deflated CG algorithm introduced by Nicolaides [47].

In Tang et al. [49], a comparison of several alternatives was conducted, and it was concluded that a multiplicative combination of a general preconditioner with the coarse grid correction $\boldsymbol{Q}$, as discussed, for instance, in Smith et al. [39], results in the most robust and efficient method. The combination, termed "Adapted Deflation Variant 2" (A-DEF2) results in a preconditioner of the form:

$$\boldsymbol{M}_{\text{A-DEF2}} = \boldsymbol{P}^T\boldsymbol{M}^{-1} + \boldsymbol{Q} \tag{16}$$

where $\boldsymbol{M}^{-1}$ can be an arbitrary preconditioner. The operator is not SPD, however it was shown to be equivalent to other symmetric operators, including the one corresponding to the deflated CG [47], provided that the following initial solution is used:

$$\boldsymbol{u_0} = \boldsymbol{Q}\boldsymbol{f} + \boldsymbol{P}^T\boldsymbol{u}_{-1} \tag{17}$$

where $\boldsymbol{u}_{-1}$ is some arbitrary initial solution vector.

In algorithm 2, the efficient computation of this initial solution is described, while in algorithm 3 the steps required to efficiently apply A-DEF2 as a preconditioner, given a residual $\boldsymbol{r}$ are described. It should be mentioned that, both of these can be used in conjunction to the standard CG method, described in algorithm 1.

Algorithm 2: Initial solution to be used with the A-DEF2 preconditioner.

Data: $\boldsymbol{K}$, $\boldsymbol{W}$, $\boldsymbol{f}$, $\boldsymbol{u}_{-1}$
Result: $\boldsymbol{u}_0$
1 Solve: $\boldsymbol{W}^T\boldsymbol{K}\boldsymbol{W}\boldsymbol{\lambda} = \boldsymbol{W}^T\boldsymbol{K}\boldsymbol{u}_{-1}$;
2 Solve: $\boldsymbol{W}^T\boldsymbol{K}\boldsymbol{W}\boldsymbol{\mu} = \boldsymbol{W}^T\boldsymbol{f}$;
3 $\boldsymbol{u}_0 = \boldsymbol{u}_{-1} + \boldsymbol{W}\left(\boldsymbol{\mu} - \boldsymbol{\lambda}\right)$;

Algorithm 3: A-DEF2 preconditioner.

Data: $\boldsymbol{K}$, $\boldsymbol{M}$, $\boldsymbol{W}$, $\boldsymbol{r}$
Result: z
1 $\boldsymbol{y} = \boldsymbol{M}^{-1}\boldsymbol{r}$;
2 Solve: $\boldsymbol{W}^T\boldsymbol{K}\boldsymbol{W}\boldsymbol{\mu} = \boldsymbol{W}^T\boldsymbol{K}\boldsymbol{y}$;
3 Solve: $\boldsymbol{W}^T\boldsymbol{K}\boldsymbol{W}\boldsymbol{\lambda} = \boldsymbol{W}^T\boldsymbol{r}$;
4 $z = \boldsymbol{y} + \boldsymbol{W}\left(\boldsymbol{\lambda} - \boldsymbol{\mu}\right)$;

3.3. Deflation space

The choice of deflation space is critical to the performance of deflation-based preconditioners. To this end, different approaches have been proposed to efficiently approximate lower eigenvalues of linear systems resulting from the discretisation of different problems. Extending the categorisation of Diaz-Cortes et al. [50], the following techniques can be identified for constructing deflation spaces:

Recycling deflation In this technique, the most effective vectors of Krylov-subspaces used in previous solutions are identified and re-used as deflation vectors [51].

Subdomain deflation Using this approach, the problem domain is divided into subdomains and vectors corresponding to a constant solution in each subdomain are used to form the deflation space [47].

Multigrid and multilevel deflation As mentioned in Section 3.2, deflation spaces can also be seen as prolongation/restriction operators [49]. Thus prolongation/restriction operators constructed for multigrid/multilevel methods can also be used as deflation vectors.

POD based deflation Using this approach, solutions from problems similar to the problem to be solved are compressed using the Proper Orthogonal Decomposition (POD) and used as deflation vectors [50]. It should be mentioned that the POD can also be used in combination to Krylov solvers in reduced order modelling methods, such as the ones proposed by Kerfriden et al. [52,53], where a limited number of CG iterations are employed to enrich lower dimensional spaces used for fracture problems.

Herein we employ subdomain deflation, while we enrich the deflation space with additional vectors to account for cracks.

3.3.1. Deflation vectors for linear elasticity

For general linear elasticity problems, subdomain-based deflation spaces can be constructed by considering rigid body translations and rotations of each subdomain [40]. This approach results in deflation matrices assuming the following values for each node of the subdomain of interest:

$$\boldsymbol{W}_I = \begin{bmatrix} 1 & 0 & 0 & 0 & z_I & -y_I \\ 0 & 1 & 0 & -z_I & 0 & x_I \\ 0 & 0 & 1 & y_I & -x_I & 0 \end{bmatrix} \tag{18}$$

where x_I, y_I, z_I are the spatial coordinates of node I.

To render deflation effective, the problem domain should be divided evenly and subdomains should also be contiguous. In our implementation, METIS [54] is used for this task, which offers additional features, that, as described in Section 3.4, can be used to further accelerate the solution.

3.3.2. Enriched deflation vectors for XFEM/GFEM

For enriched approximations, we introduce a set of additional deflation vectors to account for the presence of the crack. These additional modes are only defined in subdomains containing enriched nodes and are derived by considering modes of deformation corresponding to the enrichment functions. For instance, for the modified Heaviside enrichment function the following modes are considered:

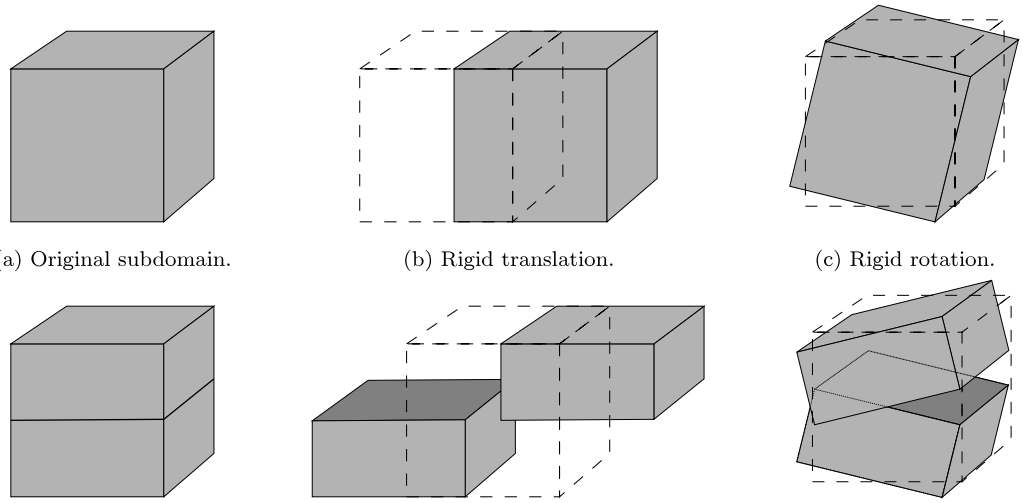
$$\boldsymbol{u}^h(\boldsymbol{x}_I) = \begin{bmatrix} H(\boldsymbol{x}_I) \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{u}^h(\boldsymbol{x}_I) = \begin{bmatrix} 0 \\ H(\boldsymbol{x}_I) \\ 0 \end{bmatrix}, \boldsymbol{u}^h(\boldsymbol{x}_I) = \begin{bmatrix} 0 \\ 0 \\ H(\boldsymbol{x}_I) \end{bmatrix} \tag{19}$$

$$\boldsymbol{u}^h(\boldsymbol{x}_I) = \begin{bmatrix} 0 \\ -H(\boldsymbol{x}_I)\, z_I \\ H(\boldsymbol{x}_I)\, y_I \end{bmatrix}, \boldsymbol{u}^h(\boldsymbol{x}_I) = \begin{bmatrix} H(\boldsymbol{x}_I)\, z_I \\ 0 \\ -H(\boldsymbol{x}_I)\, x_I \end{bmatrix}, \boldsymbol{u}^h(\boldsymbol{x}_I) = \begin{bmatrix} -H(\boldsymbol{x}_I)\, y_I \\ H(\boldsymbol{x}_I)\, x_I \\ 0 \end{bmatrix} \tag{20}$$

These modes, in accordance to the modes of Eq. (18) correspond to rigid translation and rotation of the two parts of the subdomain, lying above and below the crack, as illustrated in Fig. 2.

In order to derive the corresponding nodal values of the resulting deflation vectors, we start with the first mode considered, as well as the displacement approximation of Eq. (7). Then, we observe that the following values for the standard and enriched degrees of freedom satisfy the first of Eq. (19):

$$\boldsymbol{u}_I = \begin{bmatrix} H(\boldsymbol{x}_I) \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{b}_I = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{21}$$



(a) Original subdomain. (b) Rigid translation. (c) Rigid rotation.

(d) Cracked subdomain. (e) Rigid translation. (f) Rigid rotation.

Fig. 2. Standard and enriched deflation modes for a subdomain.

Similarly, the nodal values for the remaining modes can be derived, leading to the following deflation matrices:

$$\boldsymbol{W}_I^{\mathrm{enr}_u} = \begin{bmatrix} H(\boldsymbol{x}_I) & 0 & 0 & 0 & H(\boldsymbol{x}_I)\, z_I & -H(\boldsymbol{x}_I)\, y_I \\ 0 & H(\boldsymbol{x}_I) & 0 & -H(\boldsymbol{x}_I)\, z_I & 0 & H(\boldsymbol{x}_I)\, x_I \\ 0 & 0 & H(\boldsymbol{x}_I) & H(\boldsymbol{x}_I)\, y_I & -H(\boldsymbol{x}_I)\, x_I & 0 \end{bmatrix} \tag{22}$$

$$\boldsymbol{W}_I^{\mathrm{enr}_b} = \begin{bmatrix} 1 & 0 & 0 & 0 & z_I & -y_I \\ 0 & 1 & 0 & -z_I & 0 & x_I \\ 0 & 0 & 1 & y_I & -x_I & 0 \end{bmatrix} \tag{23}$$

corresponding to the standard and enriched degrees of freedom respectively.

The deflation vectors for the tip enriched degrees of freedom are derived in the same way.

3.4. Preconditioning and mesh partitioning

As shown in Section 3.2, deflation can be combined with a preconditioner to further increase efficiency. Since lower eigenvalues of the system matrix are removed through deflation, the preconditioner should ideally remove higher eigenvalues to effectively reduce the condition number. If viewed as a multi-grid method, then the preconditioner assumes the role of a smoother, aiming at removing high frequency components of the error. Herein, this task is performed by a block-Jacobi preconditioner, which, additionally, can be computed completely in parallel. Block-Jacobi preconditioners approximate the system matrix, and subsequently its inverse, by a block diagonal matrix, obtained by removing off-diagonal blocks from the original matrix:

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{K}_{11} & \boldsymbol{K}_{12} & \cdots & \boldsymbol{K}_{1n_b} \\ \boldsymbol{K}_{21} & \boldsymbol{K}_{22} & \cdots & \boldsymbol{K}_{2n_b} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{K}_{n_b1} & \boldsymbol{K}_{n_b2} & \cdots & \boldsymbol{K}_{n_bn_b} \end{bmatrix} \approx \begin{bmatrix} \boldsymbol{K}_{11} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{K}_{22} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{K}_{n_bn_b} \end{bmatrix} \tag{24}$$

where n_b is the number of blocks and blocks $\boldsymbol{K}_{ij}$ can be obtained by grouping together sets of unknowns. Since $\boldsymbol{K}$ is, in practice, sparse, the specific way in which unknowns are grouped can affect the performance of the preconditioner dramatically. More specifically, grouping dofs of neighbouring nodes together results in most of the off-diagonal blocks of $\boldsymbol{K}$ being zero, rendering the approximation of Eq. (24) more accurate. Therefore, groups can be selected to minimise the number of non-zero off-diagonal terms. To maximise the effectiveness the block Jacobi preconditioner, METIS [54] is used to partition the mesh into subdomains, while minimising interactions between subdomains.

Regarding subdomains containing enriched nodes, two alternatives would be possible. The first would consist of constructing different blocks for the standard and enriched part of the stiffness matrix, and would resemble the approach used in some existing works dealing with preconditioning for XFEM/GFEM [28,38]. This approach would have the advantage of keeping the part of the preconditioner corresponding to the standard part of the approximation unaltered during a crack propagation simulation. The second alternative, which is chosen here, consists of modifying blocks of enriched subdomains to include enriched degrees of freedom, leading to the following structure:

$$\boldsymbol{K} \approx \begin{bmatrix} \boldsymbol{K}_{11} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{K}_{22} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{K}_{FFii} & \boldsymbol{K}_{FXii} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{K}_{FXii}^{T} & \boldsymbol{K}_{XXii} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{K}_{n_b n_b} \end{bmatrix} \tag{25}$$

where $\boldsymbol{K}_{FFii}, \boldsymbol{K}_{XXii}, \boldsymbol{K}_{FXii}$ are the standard, enriched and interaction parts of block ii of the stiffness matrix. In the above, it is assumed that subdomain i, to which block ii corresponds, contains enriched degrees of freedom, while subdomains 1,2 and n_b do not.

The above choice can be justified by the fact that including enriched degrees of freedom in each block allows the preconditioner to remove near-linear dependencies between the standard and enriched part of the approximation, significantly decreasing the condition number of the initial matrix. Furthermore, the computational overhead for updating enriched blocks of the preconditioner in crack propagation simulations is small since, as also mentioned in Section 2.4, the number of enriched degrees of freedom, and as a result enriched subdomains, is only a small fraction of the total number of unknowns, and subdomains respectively.

4. Numerical examples

Implementation. The proposed approach is implemented in our in-house C++ code using the Eigen [55] library for linear algebra operations. The direct solutions employed in Algorithms 2 and 3 are carried out with Pardiso [56–58], a state of the art parallel direct solver from Intel, while the direct solutions required for the block Jacobi preconditioner are carried out using the Cholesky solver offered by Eigen [55], which, for small systems, as the ones arising within the preconditioner, is more efficient than Pardiso. The only parts of the solver/preconditioner that are implemented in parallel, are the matrix–vector multiplications involved in the CG algorithm, for which the shared memory parallelisation offered by Eigen is employed, and the block Jacobi preconditioner, for which, OpenMP is employed to carry out factorisations and backward substitutions in parallel. The relative tolerance for the CG solver is set to 10^{-8}.

All meshes are generated using gmsh [59] and results are visualised using Paraview [60,61]. It should be noted that adopted meshes are not always the optimal choice for the problem tested, rather they are aimed at illustrating the performance of the proposed approach for systems of varying size. Optimised meshes can be obtained through local refinement at the vicinity of the crack or even through adaptive refinement based on error estimators [62–64].

All of the following examples were ran on a workstation equipped with an Intel Xeon E3-1275 quad core processor, running at 3.80 GHz, and 32 GB of memory.

Crack propagation. Crack propagation lengths are computed using a Paris law, as described in Agathos et al. [45] and crack propagation angles are computed using the maximum circumferential stress criterion. Stress Intensity Factors (SIFs) and energy release rates, are computed using the interaction integral method, as described in Agathos et al. [15].

4.1. Cracked beam under three point bending

The first example involves crack propagation in a cracked beam under three point bending, which has already been the subject of numerical [44,65] and experimental studies [66]. As illustrated in Fig. 3, the crack is not normal to the axis of the beam, thus resulting in non-planar crack propagation. The geometrical parameters of the problem

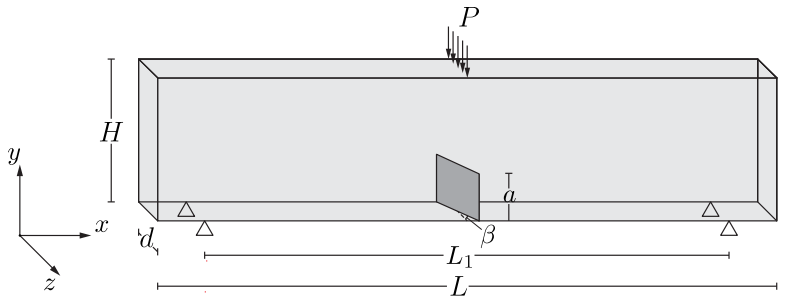
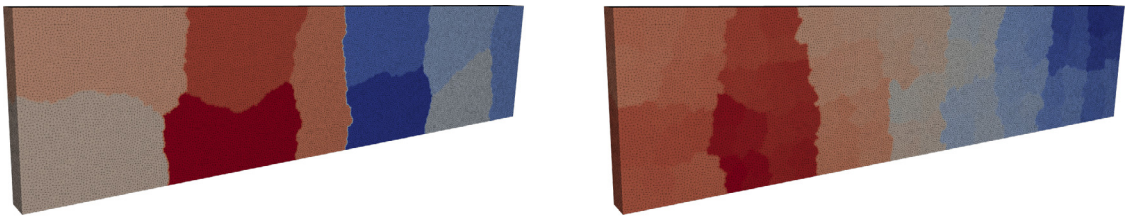


Fig. 3. Cracked beam under three point bending. The geometrical parameters of the problem are $L = 260$ mm, $L_1 = 240$ mm, $H = 60$ mm, $d = 10$ mm, $\alpha = 20$ mm, $\beta = 45°$.



(a) 10 subdomains. (b) 1,000 subdomains.

Fig. 4. Cracked beam under three point bending. Subdomains used for constructing the deflation space for two different cases for the mesh with $h = 1$ mm. Different colours correspond to different subdomains. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 1
Cracked beam under three point bending. Mesh statistics for three different refinement levels, corresponding to different mesh sizes h. The number of enriched dofs refers to the initial configuration, before any crack propagation occurs.

	$h = 2$ mm	$h = 1$ mm	$h = 0.5$ mm
Elements	125,179	676,949	4,036,623
Nodes	26,405	134,808	753,639
dofs	79,215	404,424	2,260,917
Enriched dofs	846	5535	26,550

are $L = 260$ mm, $L_1 = 240$ mm, $H = 60$ mm, $d = 10$ mm, $\alpha = 20$ mm, $\beta = 0°, 45°$, while the material parameters are $E = 2.1 \times 10^5$ N/mm^2, $\nu = 0.3$ and the applied load $P = 2$ kN.

Three different levels of mesh refinement are used, the statistics of which are provided in Table 1. Moreover, in Fig. 4, the mesh with mesh size $h = 1$ mm is illustrated along with the corresponding subdomains for two different cases. For all cases, a geometrical enrichment strategy is used with an enrichment radius $r_e = 3$ mm.

4.1.1. Effectiveness and robustness of the proposed preconditioning scheme and deflation space for enriched approximations

As a first test, in Figs. 5 and 6, the effectiveness of both the proposed preconditioner and enriched deflation space are assessed in terms of numbers or iterations and wall time for the second level of mesh refinement ($h = 1$ mm) used and different numbers of subdomains. More specifically, results of Fig. 5, refer to a beam without a crack, while in Fig. 6 the proposed approach is tested for the beam with the crack depicted in Fig. 3, which introduces enriched degrees of freedom. As a baseline for the comparisons, results were also obtained using a Jacobi preconditioner and Pardiso, which might often be the preferred option for problems of this size. For the Jacobi preconditioner, the single subdomain case corresponds to no deflation, while for the block Jacobi preconditioner, this case was not included since it would correspond to a solution with a direct solver.

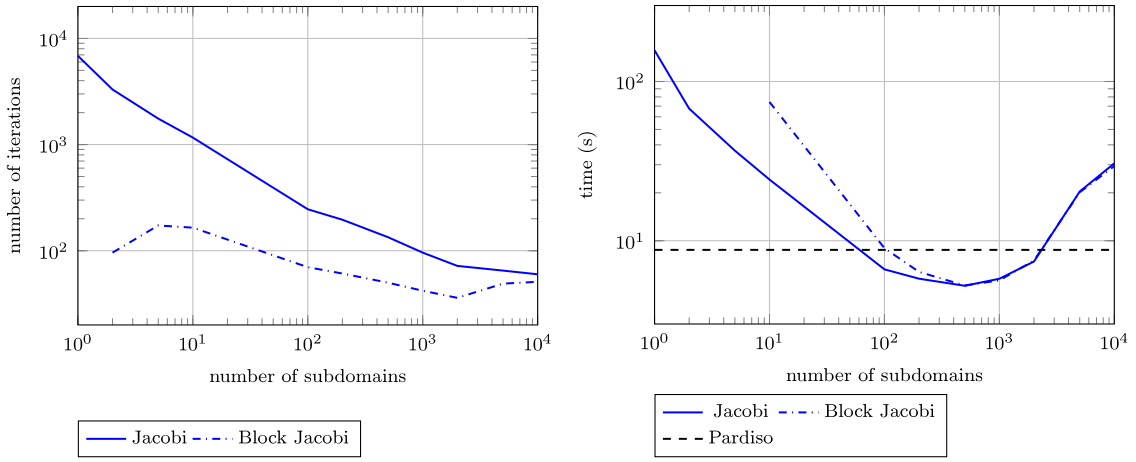


Fig. 5. Cracked beam under three point bending. Performance of deflation in conjunction to a Jacobi and a block Jacobi preconditioner for the beam without a crack.

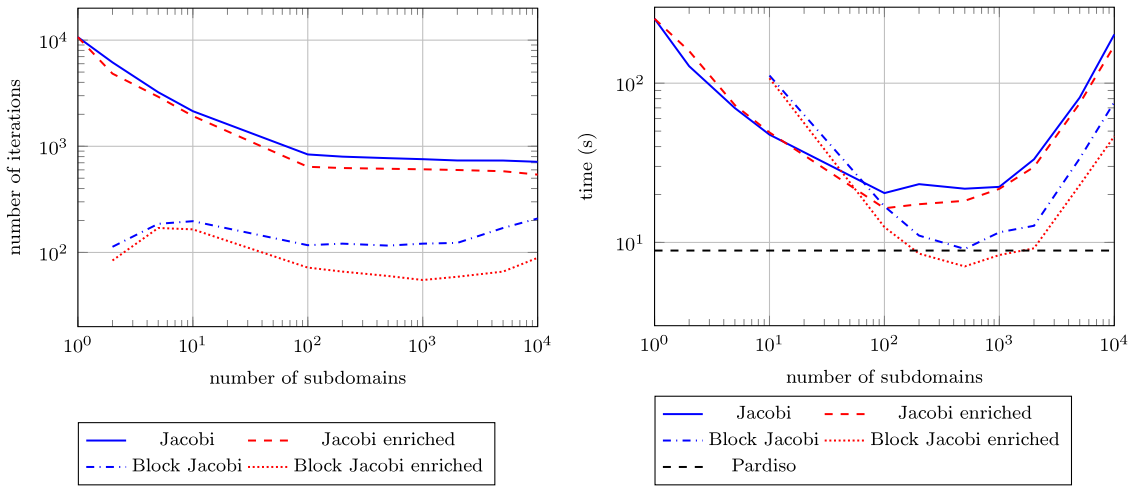


Fig. 6. Cracked beam under three point bending. Performance of deflation in conjunction to a Jacobi and the proposed block Jacobi preconditioner with the standard and enriched deflation space.

Regarding the numbers of iterations, the Jacobi preconditioner results in the typical behaviour observed for deflated CG solvers, with iterations decreasing as the number of subdomains increases. However, it should be noted that, for the enriched case (Fig. 6), this decrease is not as substantial as a result of near-linear dependencies introduced by enrichment. The block Jacobi preconditioner results in a significant reduction in the number of iterations, especially in the enriched case, since it is much more effective in removing high frequency components of the error in each subdomain by accounting for interactions between different dofs. Moreover, the dependence of the required iterations on the number of subdomains is less significant, but more complex, consisting of an initial increase (2–10 subdomains), followed by a decrease (10–1000 subdomains) and then a second increase (1000–10,000 subdomains). This can be attributed to the effect of the decreasing block size as the number of subdomains increase. More specifically, for small numbers of subdomains, blocks are large, and the preconditioner is a very accurate approximation of the system matrix, leading to a reduced number of iterations. As a limit case, if only one subdomain was to be used, the solver would converge in a single iteration. On the other hand, as the number of subdomains increases, blocks become smaller and the block preconditioner becomes less and less accurate, the limit case being the Jacobi preconditioner. This trend is confirmed in both Figs. 6 and 5, where the numbers of iterations for the two cases tend towards the same value as subdomains increase. Between the two aforementioned

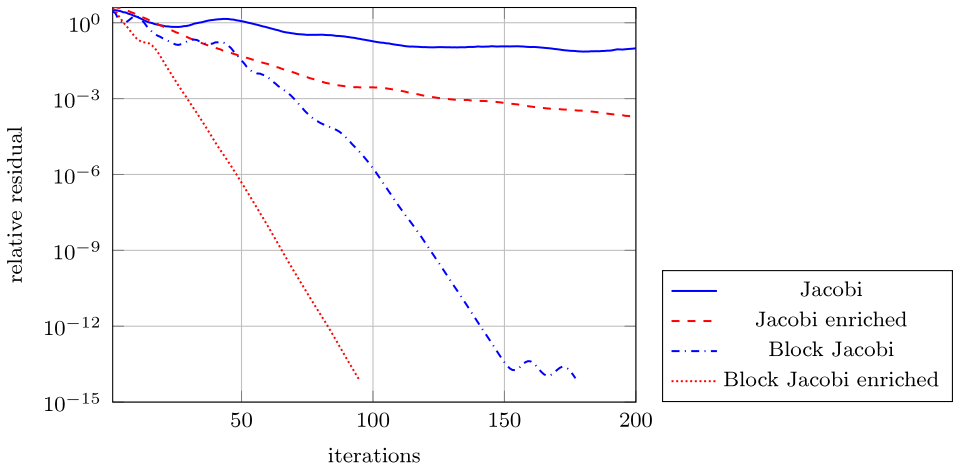


Fig. 7. Cracked beam under three point bending. Relative residual of the CG solver for different numbers of iterations for all considered cases and 500 subdomains.

regimes, the effect of preconditioning is combined to the effect of deflation leading to a decrease in the number of iterations.

In terms of wall time, in all cases it initially decreases as the number of subdomains increase, but tends to increase again after a certain point, as a result of iterations becoming more expensive for very large numbers of subdomains and the number of required iterations increasing in the case of the block Jacobi preconditioner. For the block Jacobi preconditioner, the initially increased computational time is attributed largely to the high cost of factorising blocks of the system matrix and performing backward substitutions. In fact, the peak in performance occurs for a block size, for which the direct solver used for these factorisations is the most efficient. Furthermore, while in the standard FE case (no enrichment, Fig. 5) both preconditioners offer comparable performance, in the enriched case (Fig. 6) the block Jacobi preconditioner is significantly more efficient, since it requires fewer iterations. This reduction comes as a result of near-linear dependencies between enriched and standard dofs being removed within each block.

The use of an enriched deflation space always provides an additional improvement in terms of both iterations and wall time. When used in combination to the block Jacobi preconditioner, this improvement is more substantial. In terms of computational time, when used with an optimal number of subdomains, the proposed approach can be more efficient even than the direct solver, while compared to the baseline Jacobi preconditioned CG (1 subdomain case), it can be over 35 times as fast.

In Fig. 7, the relative residual of the solution is given for different numbers of CG iterations for all the alternatives considered and 500 subdomains. The proposed combination of enriched deflation vectors and block Jacobi preconditioning achieves the best performance, allowing to efficiently solve the problem even for very low tolerances.

Although our current implementation is not optimised for parallel performance and scalability, in Fig. 8 a comparison is performed between the proposed approach and Pardiso, in terms of wall time, for different numbers of CPU threads. For the proposed approach 500 subdomains were used, which were found to yield the best performance. Two timings are provided, the first includes the time for mesh partitioning, preconditioner set up and solution, while, in the second, mesh partitioning, which is performed serially, is not included. As can be seen, the proposed approach is more efficient, while scaling, when mesh partitioning is excluded, is similar. When the time needed for mesh partitioning is taken into account, scaling slightly deteriorates, however this could be improved by employing a parallel mesh partitioner.

4.1.2. Performance of the proposed scheme for different levels of mesh refinement

Next, the performance of the proposed scheme is tested for different mesh densities to provide some insight regarding the scalability of the approach. The results are given in Fig. 9 in terms of iterations for different numbers of unknowns. Each point in the figure corresponds to one of the meshes of Table 1, while different lines correspond

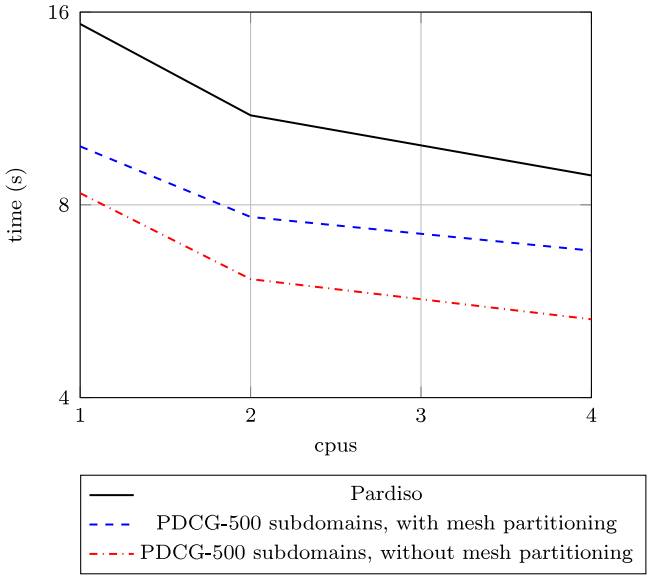


Fig. 8. Cracked beam under three point bending. Wall time for different number of cpu threads for Pardiso and the proposed approach using 500 subdomains. For the proposed approach, timings including and excluding mesh partitioning, which is performed serially, are provided.

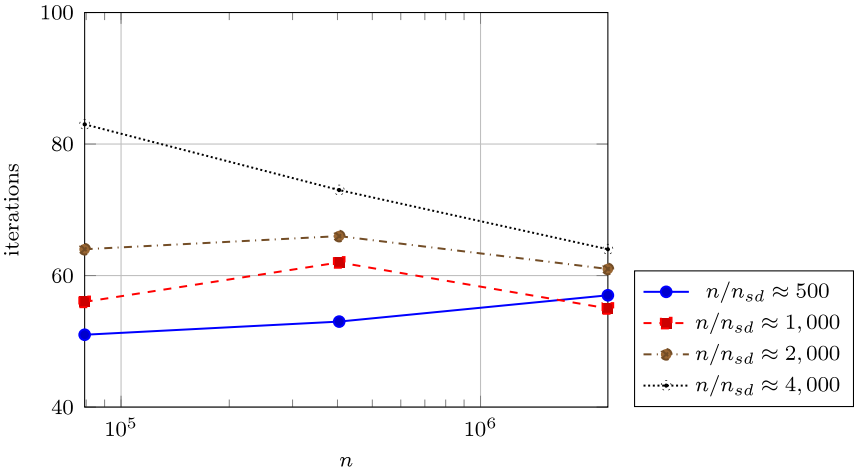
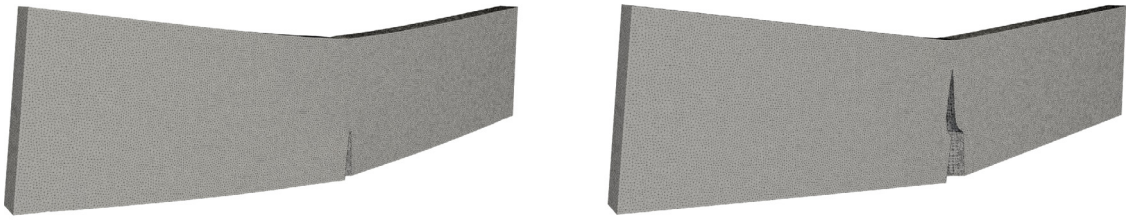


Fig. 9. Cracked beam under three point bending. Number of iterations required using the proposed preconditioner for systems of different size and different numbers of subdomains. For each of the lines the ratio between the total number of unknowns and the number of subdomains (n/n_{sd}) is kept roughly constant, resulting in subdomains of similar size.

to different ratios between the total number of unknowns (n) and the number of subdomains (n_{sd}). Thus, for each line in Fig. 9 the number of dofs per subdomain is kept roughly constant for the different mesh densities. As can be seen, although there are some fluctuations, in most cases, the number of iterations does not change significantly for different system sizes, indicating that the proposed approach is scalable.

4.1.3. Performance of the proposed scheme for propagating cracks

As a final test, the proposed approach is tested for a propagating crack. In Fig. 10, the deformed mesh is illustrated for the initial crack and after 30 steps of crack propagation, while in Fig. 11 the number of iterations and wall time required by the linear solver at each crack propagation step are given for the different alternatives considered using 500 subdomains. In general, similar trends to Fig. 6 can be observed, however, crack propagation tends to affect the preconditioners considered in different ways. In particular, in all cases, the number of iterations and wall time tend to



(a) Initial crack. (b) After 30 crack propagation steps.

Fig. 10. Cracked beam under three point bending. Deformed mesh for the initial and propagated crack after 30 steps of crack propagation for the mesh with $h = 1$ mm.

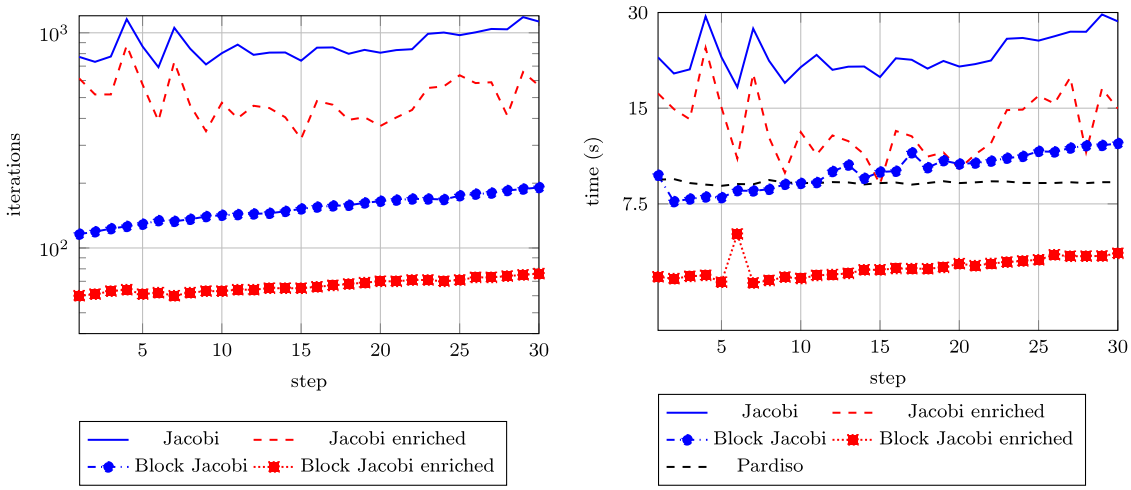


Fig. 11. Cracked beam under three point bending. Number of iterations and wall time at each step of crack propagation for all alternatives considered using 500 subdomains.

increase as the crack propagates, which can be attributed to the increased number of degrees of freedom and its effect on the solution, which, as illustrated in Fig. 10, can be quite substantial. The Jacobi preconditioner, exhibits a much more oscillatory behaviour, indicating that its performance is strongly affected by the way in which subdomains are intersected by the crack. While enriching the deflation space improves the situation in terms of overall performance, it does not lead to a less oscillatory behaviour. On the other hand, the block Jacobi preconditioner is much more robust with respect to crack propagation and when combined to an enriched deflation space it leads to a further improvement in performance.

It should be noticed that, the difference between the proposed approach and Pardiso in Fig. 11 is more pronounced than in Fig. 6, since the timings reported in Fig. 6 also include the time needed for domain partitioning, which in the crack propagation case only needs to be performed once. As an indication of the difference in performance, the total time required for linear solutions for this example using Pardiso is 262 s, while using the proposed approach 144 s, including the time required for partitioning and all other necessary tasks, such as computation of the standard and enriched deflation vectors, projection of the system matrix in the deflation space, initialisation and update of the preconditioner. Furthermore, this difference could have been even more pronounced if a less conservative tolerance was used for the iterative solver, which based on our experiments might have been possible without affecting the accuracy of the results.

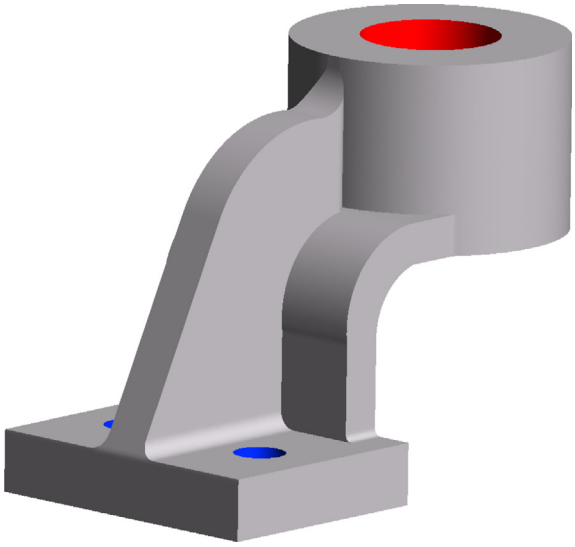


Fig. 12. Cracked bearing bracket geometry and boundary conditions. Blue colour corresponds to fixed displacements and red to a uniform traction in the vertical direction. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Table 2
Cracked bearing bracket. Mesh statistics for the mesh used. The number of enriched dofs refers to the initial configuration, before any crack propagation occurs.

	$h = 1$ mm
Elements	4,598,525
Nodes	793,975
dofs	2,381,925
Enriched dofs	15,552

4.2. Cracked bearing bracket

The next example involves the geometry of a bearing bracket, downloaded from grabcad.com and illustrated in Fig. 12. The same material properties as the previous example are used: $E = 2.1 \times 10^5$ N/mm^2, $\nu = 0.3$. Zero displacements are imposed at the two holes in the bottom (highlighted in blue) and uniform surface tractions are applied in the horizontal direction at the top hole (highlighted in red). The initial location of the crack is shown in Fig. 15(a).

An unstructured mesh with mesh size $h = 1$ mm, whose statistics are given in Table 2, is used. For the enriched part, a geometrical enrichment strategy is used with an enrichment radius $r_e = 3$ mm.

4.2.1. Effectiveness and robustness of the proposed preconditioning scheme for enriched approximations

For this example, only the enriched deflation space, combined with the Jacobi and block Jacobi preconditioners is considered. The standard deflation space is not considered, since in the previous example it was shown to be less effective than the enriched one, while the direct solver would be inefficient due to excessive memory requirements.

In Figs. 13 and 14, the performance of the two preconditioners is illustrated for different numbers of subdomains in terms of iterations and computational time for the bracket without and with a crack respectively. In this example, due to the small size of the crack compared to the component, the response is only slightly affected, resulting in identical numbers of iterations for the healthy and cracked case and almost identical computational times. Moreover, the block Jacobi preconditioner reduces the number of required iterations by almost a full order of magnitude, compared to the Jacobi preconditioner. For higher numbers of subdomains, this also translates to a significant decrease in computational time. Similar to the previous example, optimal performance is obtained for subdomains including 500–1000 dofs, where the direct solver used for the block preconditioner is the most efficient.

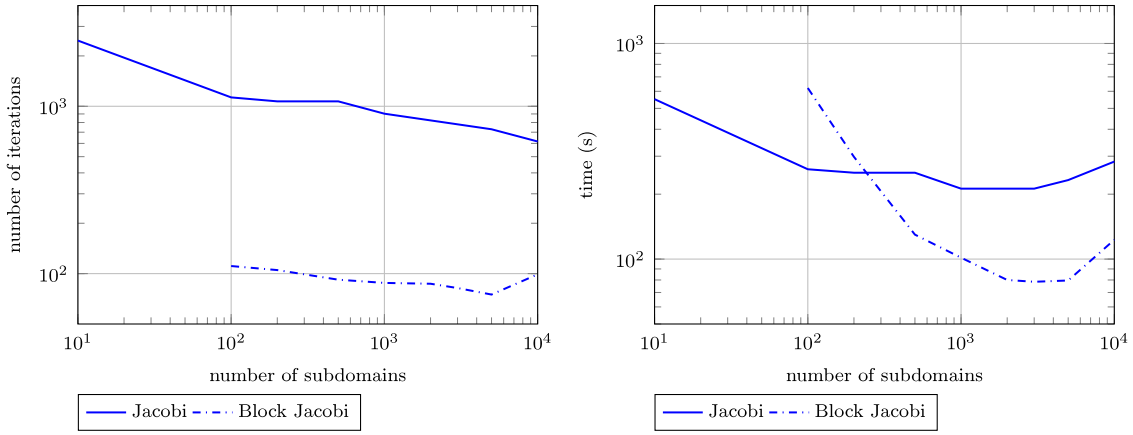


Fig. 13. Cracked bearing bracket. Performance of deflation in conjunction to a Jacobi and a block Jacobi preconditioner for the bracket without a crack.

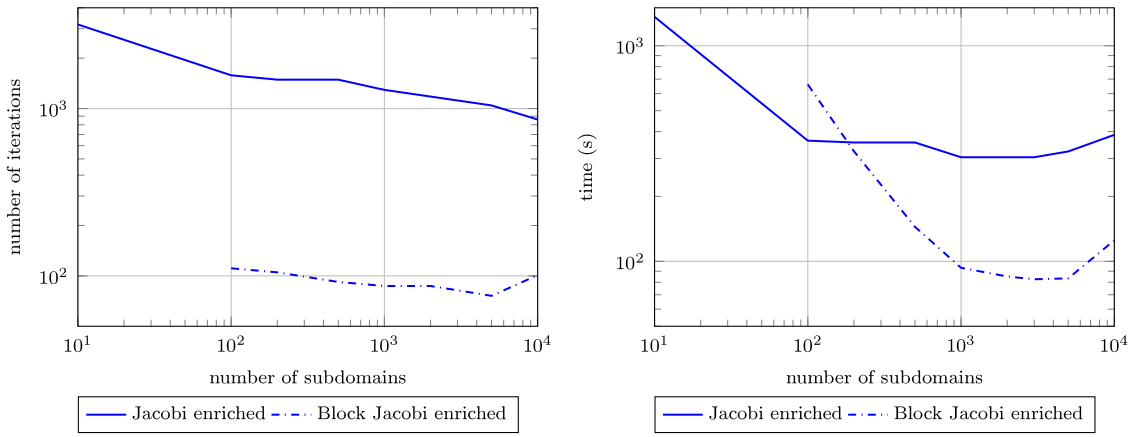


Fig. 14. Cracked bearing bracket. Performance of deflation with an enriched deflation space in conjunction to a Jacobi and a block Jacobi preconditioner for the bracket with a crack.

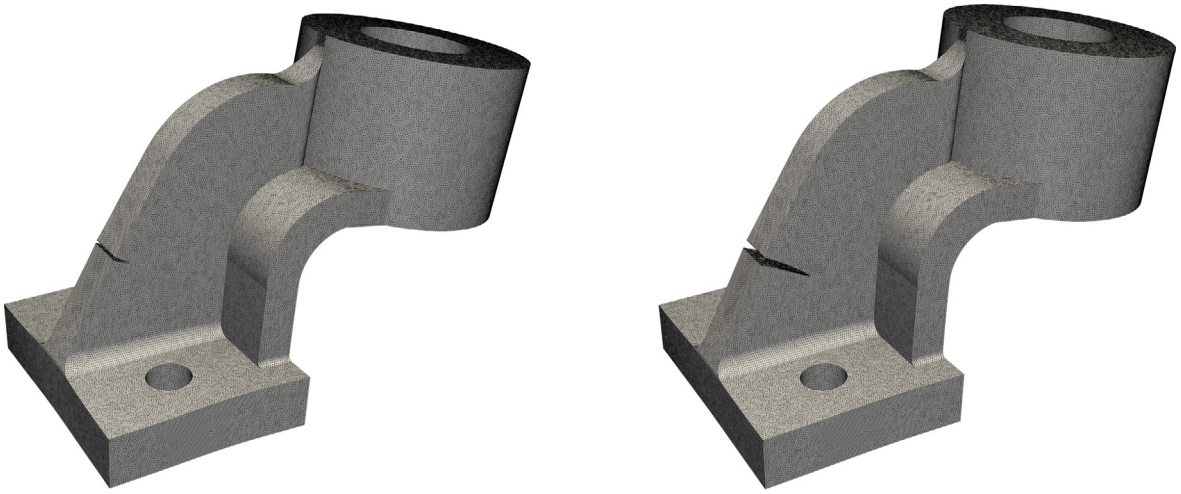
4.2.2. Performance of the proposed scheme for propagating cracks

Similar to the previous example, the final test for this example involves a propagating crack. In Fig. 15, the deformed mesh is illustrated for the initial crack and after 15 crack propagation steps. In Fig. 16 the number of iterations and wall time required at each crack propagation step are given for the two alternatives considered using 5000 subdomains. As shown in the previous paragraph, the proposed block Jacobi preconditioner is much more efficient for this example. Furthermore, in both cases the number of iterations, and, as a result, the computational time required, remains almost unaffected as the crack propagates.

4.3. Cracked compressor blade

In the final example, the geometry of a high-pressure compressor blade, obtained from grabcad.com, is considered, as illustrated in Fig. 17. The material properties used are: $E = 2.1 \times 10^5$ N/mm^2, $\nu = 0.3$, while boundary conditions applied are illustrated in Fig. 17(a), where blue areas correspond to fixed displacements and the red one to uniform tractions in the vertical direction.

The initial crack considered is circular 1 mm crack, introduced at the location illustrated in Fig. 19(a). An unstructured mesh with size in the range $h = 0.2-2$ mm is used, as illustrated in Fig. 17(b), whose statistics are given in Table 3. The smaller mesh size is used in the area around the crack to accurately represent its geometry



(a) Initial crack. (b) After 15 crack propagation steps.

Fig. 15. Cracked bearing bracket. Deformed mesh for the initial and propagated crack after 15 steps of crack propagation.

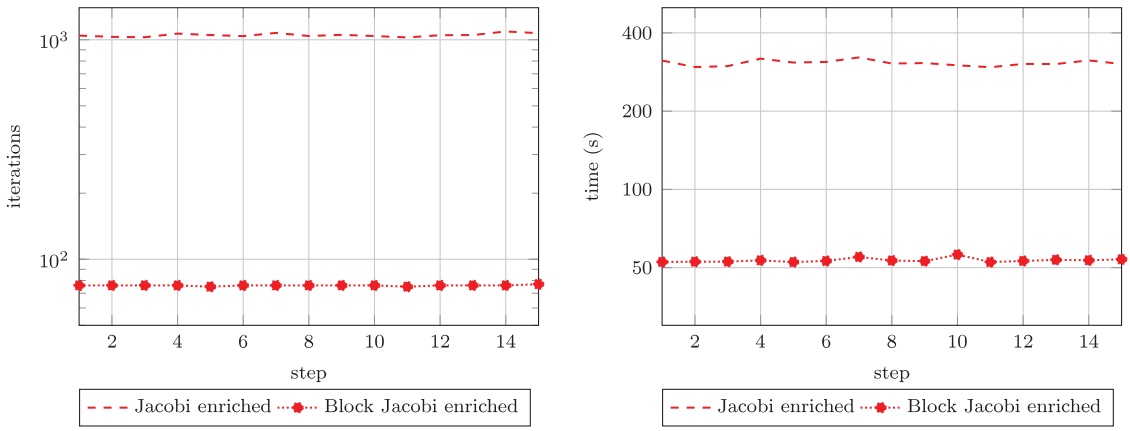
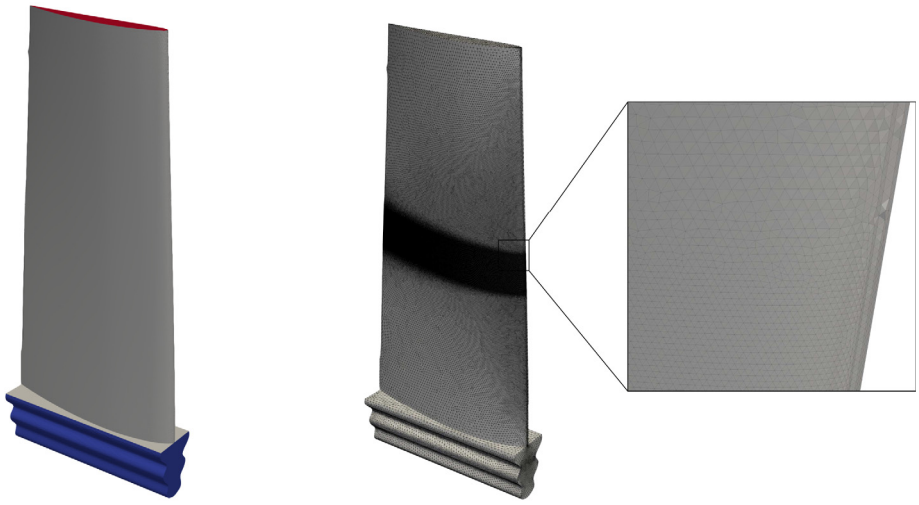


Fig. 16. Cracked bearing bracket. Number of iterations and wall time at each step of crack propagation for the alternatives considered using 5000 subdomains.

Table 3
Cracked compressor blade. Mesh statistics for the mesh used. The number of enriched dofs refers to the initial configuration, before any crack propagation occurs.

	$h = 0.2{-}2$ mm
Elements	10,890,279
Nodes	1,825,964
dofs	5,477,892
Enriched dofs	876

and resulting stress, strain and displacement fields, as well as at other locations of the blade to represent smaller geometrical features. For the enriched part, the enrichment radius is set to $r_e = 0.2$ mm since it was observed that higher enrichment radii led to fluctuations of the crack surface along the boundaries of the domain.



(a) Cracked compressor blade geometry and boundary conditions. Blue colour corresponds to fixed displacements and red to a uniform traction in the vertical direction.

(b) Cracked compressor blade mesh.

Fig. 17. Cracked compressor blade, mesh, geometry and boundary conditions. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4.3.1. Effectiveness and robustness of the proposed preconditioning scheme for enriched approximations

For this example, only results for the block Jacobi preconditioner are reported in Fig. 18, in terms of the number of iterations and wall time required for the blade without and with the crack. The behaviour observed is slightly different from the previous examples, since the number of iterations seems to be decreasing throughout the full range of subdomain numbers tested, while wall time initially decreases and subsequently increases again. This can be attributed to a number of factors. Firstly, the time reported includes the time for partitioning the domain as well as the time for initialising the preconditioner, both of which are quite substantial for this example and can affect the overall timing. In fact, the time required for the solution itself is significantly smaller and keeps decreasing for the 10,000 subdomain case, as will be shown in the following paragraph. Additionally, for the numbers of subdomains required for this example, the size of the linear systems of Algorithms 2 and 3 becomes quite substantial. Since these linear systems have to be solved at each iteration, they can slow down the preconditioner, especially as the number of subdomains increases. Despite these differences, a subdomain size similar to the previous examples yields the best results in terms of efficiency.

In the present example, the numbers of iterations required for the uncracked and cracked case are identical, since, in a similar fashion to the previous example, the crack is very small compared to the domain considered and affects the solution only locally. The differences in wall time observed, can be attributed to the fact that, in our implementation, the preconditioner is initialised without considering the crack and then updated at each crack propagation step by factorising the blocks of the stiffness matrix corresponding to enriched subdomains.

4.3.2. Performance of the proposed scheme for propagating cracks

As a final test the initial 1 mm crack, illustrated in Fig. 19(a), is propagated for 60 steps, until it almost penetrates the blade, as shown in Fig. 19(c). The deformed blade is also illustrated for the initial and final state in Figs. 19(b) and 19(d) respectively, where it can be noticed that the propagated crack substantially alters the response of the blade. For this test, 10,000 subdomains were used requiring between 76 and 77 iterations and 177 s to 178 s per solution throughout the simulation. Similar to the previous example, both the number of iterations and the wall time required were only minimally affected by crack propagation.

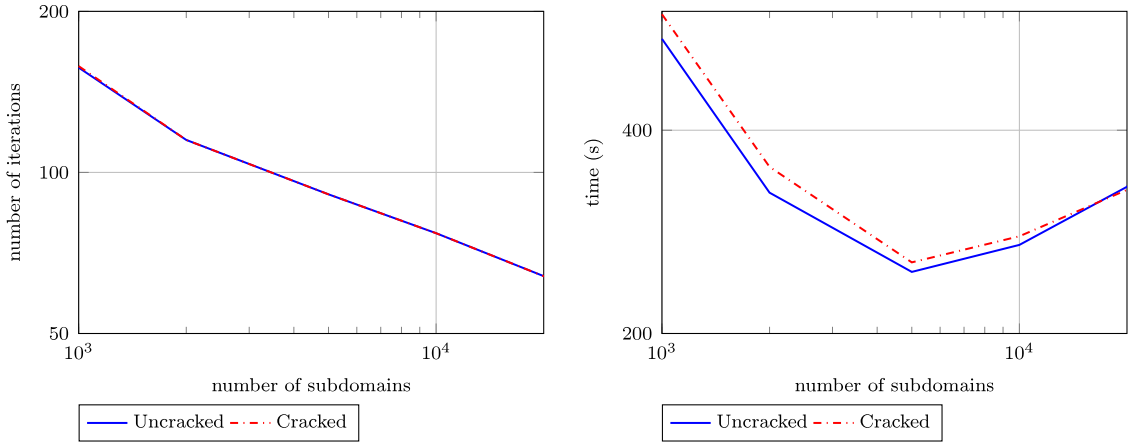
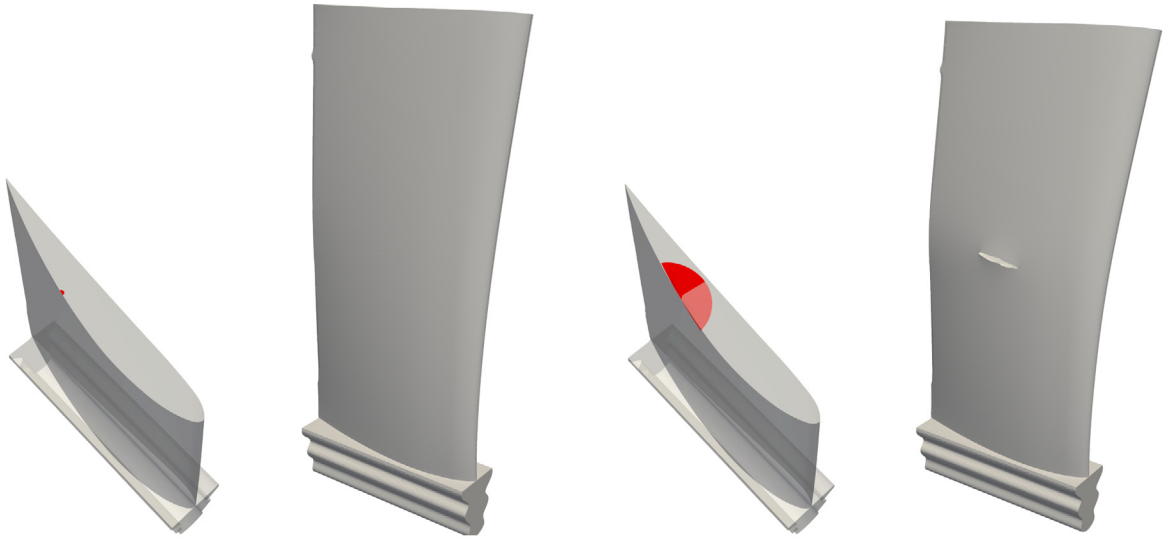


Fig. 18. Cracked compressor blade. Performance of deflation in conjunction to a block Jacobi preconditioner for the cracked and uncracked blade.



(a) Initial crack highlighted in red.

(b) Initial crack deformed mesh.

(c) Final crack highlighted in red.

(d) Final crack deformed mesh.

Fig. 19. Cracked compressor blade. Section with highlighted crack surface and deformed mesh for the initial and propagated crack after 60 steps of crack propagation. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

5. Conclusions and discussion

A preconditioning strategy was proposed to accelerate the iterative solution of linear systems resulting from the discretisation of fracture mechanics problems with extended/generalised finite elements. The approach involves deflation and a block Jacobi preconditioner combined in a multiplicative way. The deflation space typically used for linear elasticity problems is enriched to account for discontinuities and singularities introduced in the solution by enrichment, thus effectively removing low frequency components of the error. The block Jacobi preconditioner is used to remove high frequency components of the error, as well as near-linear dependencies associated with enrichment, by accounting for interactions between different dofs.

A series of numerical studies were conducted to test the performance of the proposed scheme, where the following trends were observed:

- Enriching the deflation space consistently improves the performance of the preconditioner both in terms of iterations required and computational time.
- The proposed block Jacobi preconditioner effectively removes near-linear dependencies introduced by enrichment, leading to substantial improvements in iteration numbers and computational time when compared to a Jacobi preconditioner, which is a common choice for deflated CG methods.
- For the smaller systems tested (≈500,000 dofs), the approach outperforms state of the art direct solvers. Especially for crack propagation, where the preconditioner can be set up once at the beginning of the simulation, this difference in performance can be quite substantial.
- For the larger systems tested, where cracks are small compared to the size of the components considered, the number of iterations and computational time required at each crack propagation step remains almost unaffected as cracks propagate.
- For all the systems tested, subdomain sizes of 500–1000 dofs, where the direct solver used in the block preconditioner is the most efficient, led to the most efficient solutions. However, for significantly larger systems, such a choice might result in excessively large deflation space, negatively affecting performance.

While all of the methods employed in the current work are very well suited for parallel computing, the current implementation only makes limited use of this potential. Therefore, as a direction of future work, a fully optimised implementation of the methods will be developed, utilising distributed memory parallelism, to improve performance. Moreover, since the current approach is essentially a two level preconditioner, future work will focus on extending to a multi-level method, in order to improve scalability and allow the solution of larger problems. Finally, the possibility of re-using solutions obtained in previous crack propagation steps to accelerate the solution of subsequent steps will be explored.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] N. Moës, J. Dolbow, T. Belytschko, A finite element method for crack growth without remeshing, Internat. J. Numer. Methods Engrg. 46 (1) (1999) 131–150.
[2] T. Strouboulis, I. Babuška, K. Copps, The design and analysis of the generalized finite element method, Comput. Methods Appl. Mech. Engrg. 181 (1–3) (2000) 43–69.
[3] J.M. Melenk, I. Babuška, The partition of unity finite element method: basic theory and applications, Comput. Methods Appl. Mech. Engrg. 139 (1–4) (1996) 289–314.
[4] T. Belytschko, R. Gracie, G. Ventura, A review of extended/generalized finite element methods for material modeling, Modelling Simulation Mater. Sci. Eng. 17 (4) (2009) 043001.
[5] T.-P. Fries, T. Belytschko, The extended/generalized finite element method: an overview of the method and its applications, Internat. J. Numer. Methods Engrg. 84 (3) (2010) 253–304.
[6] N. Sukumar, J. Dolbow, N. Moës, Extended finite element method in computational fracture mechanics: a retrospective examination, Int. J. Fract. 196 (1–2) (2015) 189–206.
[7] A. Egger, U. Pillai, K. Agathos, E. Kakouris, E. Chatzi, I.A. Aschroft, S.P. Triantafyllou, Discrete and phase field methods for linear elastic fracture mechanics: a comparative study and state-of-the-art review, Appl. Sci. 9 (12) (2019) 2436.
[8] C. Lang, D. Makhija, A. Doostan, K. Maute, A simple and efficient preconditioning scheme for heaviside enriched XFEM, Comput. Mech. 54 (5) (2014) 1357–1374.
[9] S. Loehnert, A stabilization technique for the regularization of nearly singular extended finite elements, Comput. Mech. 54 (2) (2014) 523–533.
[10] G. Ventura, C. Tesei, Stabilized X-FEM for heaviside and nonlinear enrichments, in: Advances in Discretization Methods, Springer, 2016, pp. 209–228.
[11] C. Duarte, I. Babuška, J. Oden, Generalized finite element methods for three-dimensional structural mechanics problems, Comput. Struct. 77 (2) (2000) 215–232.
[12] N. Chevaugeon, N. Moës, H. Minnebo, Improved crack tip enrichment functions and integration for crack modeling using the extended finite element method, Int. J. Multiscale Comput. Eng. 11 (6) (2013).
[13] P. Laborde, J. Pommier, Y. Renard, M. Salaün, High-order extended finite element method for cracked domains, Internat. J. Numer. Methods Engrg. 64 (3) (2005) 354–381.
[14] K. Agathos, E. Chatzi, S. Bordas, D. Talaslidis, A well-conditioned and optimally convergent XFEM for 3D linear elastic fracture, Internat. J. Numer. Methods Engrg. 105 (9) (2016) 643–677, http://dx.doi.org/10.1002/nme.4982.

[15] K. Agathos, E. Chatzi, S. Bordas, Stable 3D extended finite elements with higher order enrichment for accurate non planar fracture, Comput. Methods Appl. Mech. Engrg. 306 (2016) 19–46.
[16] I. Babuška, U. Banerjee, Stable generalized finite element method (SGFEM), Comput. Methods Appl. Mech. Engrg. 201 (2012) 91–111.
[17] V. Gupta, C.A. Duarte, I. Babuška, U. Banerjee, Stable GFEM (SGFEM): Improved conditioning and accuracy of GFEM/XFEM for three-dimensional fracture mechanics, Comput. Methods Appl. Mech. Engrg. 289 (2015) 355–386.
[18] A. Sanchez-Rivadeneira, C. Duarte, A stable generalized/extended FEM with discontinuous interpolants for fracture mechanics, Comput. Methods Appl. Mech. Engrg. 345 (2019) 876–918.
[19] K. Agathos, S.P. Bordas, E. Chatzi, Improving the conditioning of XFEM/GFEM for fracture mechanics problems through enrichment quasi-orthogonalization, Comput. Methods Appl. Mech. Engrg. 346 (2019) 1051–1073.
[20] K. Agathos, E. Chatzi, S.P. Bordas, A unified enrichment approach addressing blending and conditioning issues in enriched finite elements, Comput. Methods Appl. Mech. Engrg. 349 (2019) 673–700.
[21] V.E. Henson, U. Yang, BoomerAMG: a parallel algebraic multigrid solver and preconditioner, Appl. Numer. Math. 41 (2002) 155–177.
[22] A. Toselli, O. Widlund, Domain Decomposition Methods - Algorithms and Theory, Springer, 2004.
[23] N. Spillane, V. Dolean, P. Hauret, F. Nataf, C. Pechstein, R. Scheichl, Abstract robust coarse spaces for systems of PDEs via generalized eigenproblems in the overlaps, Numer. Math. 126 (2014) 741–770.
[24] A. Reinarz, T. Dodwell, T. Fletcher, L. Seelinger, R. Butler, R. Scheichl, Dune-composites – A new framework for high-performance finite element modelling of laminates, Compos. Struct. (2018) 269–278.
[25] R. Butler, T. Dodwell, A. Reinarz, A. Sandhu, R. Scheichl, L. Seelinger, High-performance dune modules for solving large-scale, strongly anisotropic elliptic problems with applications to aerospace composites, Comput. Phys. Comm. 249 (2020) 106997.
[26] C. Ma, R. Scheichl, T. Dodwell, Novel design and analysis of generalized FE methods based on locally optimal spectral approximations, arXiv arXiv:2103.09545.
[27] E. Béchet, H. Minnebo, N. Moës, B. Burgardt, Improved implementation and robustness study of the X-FEM for stress analysis around cracks, Internat. J. Numer. Methods Engrg. 64 (8) (2005) 1033–1056.
[28] A. Menk, S.P. Bordas, A robust preconditioning technique for the extended finite element method, Internat. J. Numer. Methods Engrg. 85 (13) (2011) 1609–1632.
[29] L. Berger-Vergiat, H. Waisman, B. Hiriyur, R. Tuminaro, D. Keyes, Inexact Schwarz-algebraic multigrid preconditioners for crack problems modeled by extended finite element methods, Internat. J. Numer. Methods Engrg. 90 (3) (2012) 311–328.
[30] H. Waisman, L. Berger-Vergiat, An adaptive domain decomposition preconditioner for crack propagation problems modeled by XFEM, Int. J. Multiscale Comput. Eng. 11 (6) (2013).
[31] L. Svolos, L. Berger-Vergiat, H. Waisman, Updating strategy of a domain decomposition preconditioner for parallel solution of dynamic fracture problems, J. Comput. Phys. 422 (2020) 109746.
[32] S. Bordas, B. Moran, Enriched finite elements and level sets for damage tolerance assessment of complex structures, Eng. Fract. Mech. 73 (9) (2006) 1176–1201.
[33] S. Bordas, P.V. Nguyen, C. Dunant, A. Guidoum, H. Nguyen-Dang, An extended finite element library, Internat. J. Numer. Methods Engrg. 71 (6) (2007) 703–732.
[34] C.A. Duarte, D.-J. Kim, Analysis and applications of a generalized finite element method with global–local enrichment functions, Comput. Methods Appl. Mech. Engrg. 197 (6–8) (2008) 487–504.
[35] B. Hiriyur, R.S. Tuminaro, H. Waisman, E.G. Boman, D.E. Keyes, A quasi-algebraic multigrid approach to fracture problems based on extended finite elements, SIAM J. Sci. Comput. 34 (2) (2012) A603–A626.
[36] A. Gerstenberger, R.S. Tuminaro, An algebraic multigrid approach to solve extended finite element method based fracture problems, Internat. J. Numer. Methods Engrg. 94 (3) (2013) 248–272.
[37] S. Feng, X. Han, A novel multi-grid based reanalysis approach for efficient prediction of fatigue crack propagation, Comput. Methods Appl. Mech. Engrg. 353 (2019) 107–122.
[38] T.B. Fillmore, V. Gupta, C.A. Duarte, Preconditioned conjugate gradient solvers for the generalized finite element method, in: International Workshop on Meshfree Methods for Partial Differential Equations, Springer, 2017, pp. 1–17.
[39] B. Smith, P. Bjorstad, W. Gropp, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, 2004, URL https://books.google.co.uk/books?id=dxwRLu1dBioC.
[40] R. Aubry, F. Mut, S. Dey, R. Löhner, Deflated preconditioned conjugate gradient solvers for linear elasticity, Internat. J. Numer. Methods Engrg. 88 (11) (2011) 1112–1127.
[41] K. Agathos, E. Chatzi, S.P. Bordas, Multiple crack detection in 3D using a stable XFEM and global optimization, Comput. Mech. 62 (4) (2018) 835–852.
[42] M. Duflot, A study of the representation of cracks with level sets, Internat. J. Numer. Methods Engrg. 70 (11) (2007) 1261–1302.
[43] G. Ventura, E. Budyn, T. Belytschko, Vector level sets for description of propagating cracks in finite elements, Internat. J. Numer. Methods Engrg. 58 (10) (2003) 1571–1592.
[44] K. Agathos, G. Ventura, E. Chatzi, S.P. Bordas, Well conditioned extended finite elements and vector level sets for three-dimensional crack propagation, in: Geometrically Unfitted Finite Element Methods and Applications, Springer, 2017, pp. 307–329.
[45] K. Agathos, G. Ventura, E. Chatzi, S.P. Bordas, Stable 3D XFEM/vector level sets for non-planar 3D crack propagation and comparison of enrichment schemes, Internat. J. Numer. Methods Engrg. 113 (2) (2018) 252–276.
[46] A. Sanchez-Rivadeneira, N. Shauer, B. Mazurowski, C. Duarte, A stable generalized/extended p-hierarchical FEM for three-dimensional linear elastic fracture mechanics, Comput. Methods Appl. Mech. Engrg. 364 (2020) 112970.
[47] R.A. Nicolaides, Deflation of conjugate gradients with applications to boundary value problems, SIAM J. Numer. Anal. 24 (2) (1987) 355–365.

[48] Y. Saad, M. Yeung, J. Erhel, F. Guyomarch́, A deflated version of the conjugate gradient algorithm, SIAM J. Sci. Comput. 21 (5) (2000) 1909–1926.
[49] J. Tang, R. Nabben, C. Vuik, Y. Erlangga, Theoretical and numerical comparison of various projection methods derived from deflation, domain decomposition and multigrid methods, in: Reports of the Department of Applied Mathematical Analysis, 2007.
[50] G.D. Cortes, C. Vuik, J.D. Jansen, On POD-based deflation vectors for DPCG applied to porous media problems, J. Comput. Appl. Math. 330 (2018) 193–213.
[51] M.L. Parks, E. De Sturler, G. Mackey, D.D. Johnson, S. Maiti, Recycling Krylov subspaces for sequences of linear systems, SIAM J. Sci. Comput. 28 (5) (2006) 1651–1674.
[52] P. Kerfriden, P. Gosselet, S. Adhikari, S.P.-A. Bordas, Bridging proper orthogonal decomposition methods and augmented Newton–Krylov algorithms: an adaptive model order reduction for highly nonlinear mechanical problems, Comput. Methods Appl. Mech. Engrg. 200 (5–8) (2011) 850–866.
[53] P. Kerfriden, O. Goury, T. Rabczuk, S.P.-A. Bordas, A partitioned model order reduction approach to rationalise computational expenses in nonlinear fracture mechanics, Comput. Methods Appl. Mech. Engrg. 256 (2013) 169–188.
[54] G. Karypis, V. Kumar, METIS: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices, 1997.
[55] G. Guennebaud, B. Jacob, et al., Eigen v3, 2010, http://eigen.tuxfamily.org.
[56] O. Schenk, K. Gärtner, W. Fichtner, Efficient sparse LU factorization with left-right looking strategy on shared memory multiprocessors, BIT Numer. Math. 40 (1) (2000) 158–176.
[57] O. Schenk, K. Gärtner, Two-level dynamic scheduling in PARDISO: Improved scalability on shared memory multiprocessing systems, Parallel Comput. 28 (2) (2002) 187–197.
[58] O. Schenk, K. Gärtner, Solving unsymmetric sparse systems of linear equations with PARDISO, Future Gener. Comput. Syst. 20 (3) (2004) 475–487.
[59] C. Geuzaine, J.-F. Remacle, Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities, Internat. J. Numer. Methods Engrg. 79 (11) (2009) 1309–1331.
[60] J. Ahrens, B. Geveci, C. Law, C. Hansen, C. Johnson, ParaView: An end-user tool for large-data visualization, 2005.
[61] U. Ayachit, The paraview guide: a parallel visualization application, 2015.
[62] S. Bordas, M. Duflot, Derivative recovery and a posteriori error estimate for extended finite elements, Comput. Methods Appl. Mech. Engrg. 196 (35–36) (2007) 3381–3399.
[63] Y. Jin, O. González-Estrada, O. Pierard, S. Bordas, Error-controlled adaptive extended finite element method for 3D linear elastic crack propagation, Comput. Methods Appl. Mech. Engrg. 318 (2017) 319–348.
[64] M. Duflot, S. Bordas, A posteriori error estimation for extended finite elements by an extended global recovery, Internat. J. Numer. Methods Engrg. 76 (8) (2008) 1123–1138.
[65] A. Sadeghirad, D.L. Chopp, X. Ren, E. Fang, J. Lua, A novel hybrid approach for level set characterization and tracking of non-planar 3D cracks in the extended finite element method, Eng. Fract. Mech. 160 (2016) 1–14.
[66] F.-G. Buchholz, A. Chergui, H. Richard, Fracture analyses and experimental results of crack growth under general mixed mode loading conditions, Eng. Fract. Mech. 71 (4–6) (2004) 455–468.