An Abaqus implementation of the extended finite element method

E. Giner [a,*], N. Sukumar [b], J. E. Tarancón [a], F. J. Fuenmayor [a]

[a] *Departamento de Ingeniería Mecánica y de Materiales*

Universidad Politécnica de Valencia, Camino de Vera, 46022 Valencia, Spain.

[b] *Department of Civil and Environmental Engineering*

University of California, One Shields Avenue, Davis, CA 95616, USA.

Abstract

In this paper, we introduce an implementation of the extended finite element method for fracture problems within the finite element software ABAQUS^{TM}. User subroutine (`UEL`) in Abaqus is used to enable the incorporation of extended finite element capabilities. We provide details on the data input format together with the proposed user element subroutine, which constitutes the core of the finite element analysis; however, pre-processing tools that are necessary for an X-FEM implementation, but not directly related to Abaqus, are not provided. In addition to problems in linear elastic fracture mechanics, non-linear frictional contact analyses are also realized. Several numerical examples in fracture mechanics are presented to demonstrate the benefits of the proposed implementation.

Key words: finite element analysis; extended finite element method (X-FEM); stress intensity factor; crack modelling; crack propagation.

* Corresponding author. Tel.: +34-96-3877621; fax: +34-96-3877629.
Email address: `eginerm@mcm.upv.es` (E. Giner).

1 INTRODUCTION

In recent years, the extended finite element method (X-FEM) [1] has emerged as a powerful numerical procedure for the analysis of crack problems. It has been widely acknowledged that the method eases crack growth modelling under the assumptions of linear elastic fracture mechanics (LEFM). Since the introduction of the method about a decade ago, many new extensions and applications have appeared in the scientific literature, with substantially many contributions on X-FEM in recent years. We point the reader to review articles [2,3] and to a recent monograph [4] for general overviews on the X-FEM.

We developed the present implementation of the X-FEM in the finite element code Abaqus [5] for crack propagation simulations in fretting fatigue problems [6]. In these problems cracks emanate from the edge of contact regions between bodies that experience relative displacements of small amplitude (these regions act as strong stress raisers). The Abaqus capabilities for the analysis of contact problems together with the user-defined subroutine options available in the code have proved to be useful for the analysis of these problems. In addition, the user can benefit from the many built-in features of such a code, including pre- and post-processing options. The present implementation can also be adapted by the user to extend its application to a broader class of problems.

The interest shown both by researchers in computational fracture mechanics community and by engineers in industry on our presentations at conferences [7,8] has encouraged us to pursue the present contribution. We attempt

to provide procedures that could be used by fracture mechanics practitioners who are familiar with Abaqus and can thus benefit from the crack growth modeling capabilities of the X-FEM.

This paper focuses on the implementation aspects for two-dimensional LEFM applications containing single or multiple cracks. We place emphasis on the data input format and subroutines that interact with Abaqus as a finite element solver through the user subroutine `UEL` [5]. Some of the important steps in an extended finite element analysis are not considered, such as the crack geometry and mesh interactions that are used to determine the nodes to be enriched, element subdivisions, etc. We would like to remark that these pre-processing tools, which are not directly related to Abaqus, but necessary for a full X-FEM implementation, are not included in this work and need to be provided by the user. This pre-processing step can be tackled in one of many ways, for instance, using geometric predicates or level sets [1,9–11]). Similarly, the computation of stress intensity factors using the domain form of the interaction integral [12,13] is not elaborated; details on the extraction of stress intensity factors in X-FEM can be found in Moës *et al.* [1].

In Sukumar and Prévost [11], a Fortran implementation of the X-FEM is presented, which is supported by benchmark numerical examples [14]. Bordas *et al.* [15] describe an object-oriented programming library for extended finite element analysis. Recently, Wyart *et al.* [16] provide a good description of the possible approaches that can be followed to implement X-FEM in a general-purpose FE software. They propose a substructuring approach to decompose the cracked component into safe and cracked subdomains, which are analyzed

separately by the general-purpose FE software and the extended finite element code, respectively. An alternative approach is pursued in this work: introduction of a generic enriched element based on the user-element capabilities of Abaqus. Although some researchers have introduced similar approaches for Abaqus [1], the authors are only aware of one recent contribution [17] at the time of this writing.

In what follows, a brief review on the fundamentals of the extended finite element method is presented. Then, the kernel of the Abaqus implementation is described in Section 3, including remarks on the extension to non-linear contact problems. Several numerical examples in fracture mechanics are presented in Section 4 to demonstrate the versatility of the implementation, and we close with a few concluding remarks in Section 5.

2 EXTENDED FINITE ELEMENT METHOD

In comparison to the classical finite element method, the X-FEM provides significant benefits in the numerical modelling of crack propagation. In the traditional formulation of the FEM, the existence of a crack is modelled by requiring the crack to follow element edges. In contrast, the crack geometry in the X-FEM need not be aligned with the element edges, which provides flexibility and versatility in modelling. The method is based on the enrichment of the FE model with additional degrees of freedom (DOFs) that are tied to the nodes of the elements intersected by the crack [1]. In this manner, the

[1] See discussion at `http://imechanica.org/node/1125`

discontinuity is included in the numerical model without modifying the discretization, as the mesh is generated without taking into account the presence of the crack. Therefore, only a single mesh is needed for any crack length and orientation. In addition, nodes surrounding the crack tip are enriched with DOFs associated with functions that reproduce the asymptotic LEFM fields. This enables the modelling of the crack discontinuity within the crack-tip element and substantially increases the accuracy in the computation of the stress intensity factors (SIFs).

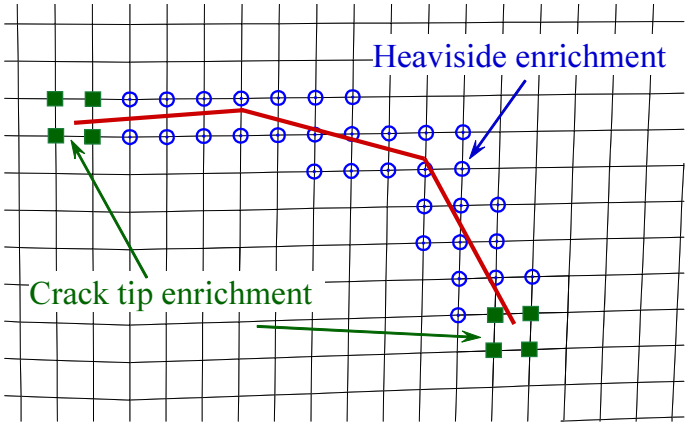


Fig. 1. Enriched nodes in the X-FEM. Circles: nodes with 2 additional DOFs. Squares: nodes with 8 additional DOFs.

2.1 Standard formulation

Fig. 1 shows a portion of the mesh with four-node bilinear elements. The circled nodes are the nodes enriched with two additional DOFs (total of four DOFs per node), whereas the nodes marked with a square are enriched by eight more DOFs (total of ten DOFs per node). Elements that contain at least one enriched node are known as enriched elements. Nodes with two additional DOFs (one for each coordinate direction) have shape functions that multiply

the Heaviside function $H(\mathbf{x})$ (function of unit magnitude whose sign changes across the crack, $H(\mathbf{x}) = \pm 1$). Physically, this function introduces the discontinuity across the crack faces. Nodes with eight additional DOFs are enriched in the two Cartesian directions with four crack tip functions $F_\alpha(\mathbf{x})$ [18]:

$$[F_\alpha(r,\theta),\ \alpha = 1\text{–}4] = \left[\sqrt{r}\sin\frac{\theta}{2},\ \sqrt{r}\cos\frac{\theta}{2},\ \sqrt{r}\sin\frac{\theta}{2}\sin\theta,\ \sqrt{r}\cos\frac{\theta}{2}\sin\theta\right], \tag{1}$$

where r, θ are local polar co-ordinates defined at the crack tip. We note that the span of the above functions can reproduce the asymptotic mode I and mode II displacement fields in LEFM, which gives rise to the near-tip singular behavior in strains and stresses. It is well-documented in the literature [1,14], and also verified through our studies that these functions significantly improve the accuracy of K_{I} and K_{II} extraction.

The displacement approximation for crack modelling in the extended finite element method takes the form [1]:

$$\mathbf{u}_{\mathrm{xfem}}(\mathbf{x}) = \sum_{i\in\mathcal{I}} N_i(\mathbf{x})\mathbf{u}_i + \sum_{i\in\mathcal{J}} N_i(\mathbf{x})H(\mathbf{x})\mathbf{a}_i + \sum_{i\in\mathcal{K}}\left[N_i(\mathbf{x})\sum_{\alpha=1}^{4} F_\alpha(\mathbf{x})\mathbf{b}_{i\alpha}\right], \tag{2}$$

where $\mathcal{I}$ is the set of all nodes in the mesh, $N_i(\mathbf{x})$ is the nodal shape function and $\mathbf{u}_i$ is the standard DOF of node i ($\mathbf{u}_i$ represents the physical nodal displacement for non-enriched nodes only). The subsets $\mathcal{J}$ and $\mathcal{K}$ contain the nodes enriched with Heaviside function $H(\mathbf{x})$ or crack-tip functions $F_\alpha(\mathbf{x})$, respectively, and $\mathbf{a}_i$, $\mathbf{b}_{i\alpha}$ are the corresponding DOFs. If there is no enrichment, then the above equation reduces to the classical finite element approximation $\mathbf{u}_{\mathrm{fe}}(\mathbf{x}) = \sum_i N_i(\mathbf{x})\mathbf{u}_i$. Hence, X-FEM retains many of the advantages of the

finite element method.

It is important to note that the additional DOFs $\mathbf{a}_i$, $\mathbf{b}_{i\alpha}$ in Eq. (2) are only added to the nodes that are enriched. Moreover, in this implementation no nodes are enriched with DOFs $\mathbf{a}_i$, $\mathbf{b}_{i\alpha}$ simultaneously (they are mutually exclusive) and hence the Heaviside functions that introduce the crack discontinuity are not used at the nodes of the crack-tip element. The crack discontinuity within the crack-tip element is modelled via the enrichment function $\sqrt{r}\sin\frac{\theta}{2}$ in Eq. (1), which is discontinuous at $\theta = \pm\pi$.

As in the standard FEM, it is necessary to perform numerical integration over the element domain to compute the element stiffness matrix. However, the elements that contain the crack include a displacement discontinuity due to the X-FEM formulation. These elements must be subdivided into subdomains in which the crack is one of the subdomain boundaries to carry out the numerical integrations, as shown in Fig. 2. It is important to emphasize that the mesh topology and connectivity are retained during the whole process of crack propagation, which is the main advantage of the X-FEM. Unless otherwise stated, in the numerical examples of Section 4 we have used 7 integration points in triangular subdomains and 5×5 integration points in both quadrilateral subdomains and within elements that are not subdivided but contain at least one enriched node. For crack-tip elements, a quasi-polar integration can also be performed [19,20].

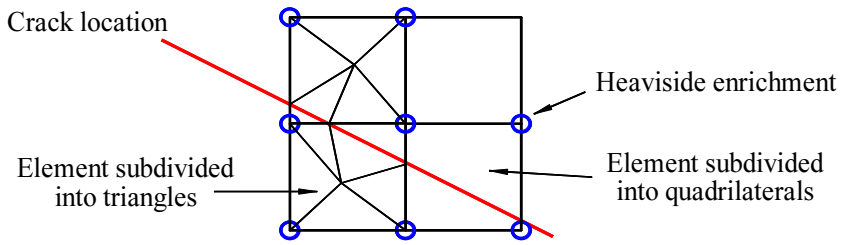


Fig. 2. Subdivision of elements intersected by a crack for integration purposes.

2.2 Shifted-basis formulation

From Eq. (2), it is clear that the physical displacement at an enriched node i, $\mathbf{u}_{\text{xfem}}(\mathbf{x}_i)$ is given by the standard DOFs $\mathbf{u}_i$ plus the enriched contribution $H(\mathbf{x}_i)\mathbf{a}_i$ or $F_\alpha(\mathbf{x}_i)\mathbf{b}_{i\alpha}$. This implies that the standard DOFs $\mathbf{u}_i$ (the ones used by Abaqus for representing the physical displacement in its internal contact or plotting procedures) do not correspond to the true displacement computed with X-FEM. In order to make the DOFs $\mathbf{u}_i$ of an enriched node i be the physical solution of the nodal displacement, the X-FEM has been implemented according to the following modification of Eq. (2), the so-called shifted-basis enrichment [10,21]. In Eq. (3), $\mathbf{x}_i$ denotes the nodal coordinates of an enriched node i. In this way, $\mathbf{a}_i$, $\mathbf{b}_{i\alpha}$ do not contribute to the value of the physical displacement at the enriched node i:

$$\begin{aligned}\mathbf{u}_{\text{xfem}}(\mathbf{x}) = &\sum_{i\in\mathcal{I}} N_i(\mathbf{x})\mathbf{u}_i + \sum_{i\in\mathcal{J}} N_i(\mathbf{x})\left[H(\mathbf{x}) - H(\mathbf{x}_i)\right]\mathbf{a}_i \\ &+ \sum_{i\in\mathcal{K}}\left[N_i(\mathbf{x})\sum_{\alpha=1}^{4}\left[F_\alpha(\mathbf{x}) - F_\alpha(\mathbf{x}_i)\right]\mathbf{b}_{i\alpha}\right].\end{aligned} \tag{3}$$

Since $H(\mathbf{x})$ and $F_1(\mathbf{x})$ are discontinuous functions across the crack, we choose $H(\mathbf{x}) = 1$ if $\mathbf{x}$ is on or above the crack, and $H(\mathbf{x}) = -1$ otherwise. Similar choices are made in defining $F_1(\mathbf{x})$. These choices ensure that the value of the

enrichment functions at any node is single-valued for any crack geometry. The enriched contribution vanishes at an enriched node, but not at an integration point. This procedure has been very useful for plotting deformed shapes in Abaqus. In addition, it has proven to be a good way to combine user elements with the Abaqus contact capabilities, as explained in Sections 3.5 and 3.6.

3 ABAQUS IMPLEMENTATION FOR 2D LEFM

In this section we describe the main features related to the Abaqus [5] implementation of the X-FEM through the user subroutine `UEL`. Implementing the X-FEM in the commercial code Abaqus does imposes certain restrictions, but it also provides access to many of the available features of such a code. As stated earlier, we will focus on the input files and subroutines directly related to Abaqus. However, we will also give a general overview of the pre- and post-processing steps used in our implementation.

3.1 Pre-processing: Crack-mesh interaction

A crucial step in any implementation of the X-FEM is the definition of the nodes to be enriched as a result of the crack and mesh geometries. This task can be relatively easy for 2D problems and can be tackled in different ways, whereas it becomes more involved in 3D. In order to determine the nodes to be enriched and the sign of the Heaviside function $H(\mathbf{x}_i)$ for 2D problems, it is sufficient to evaluate the nodal distances to the nearest crack segment and to the crack tips. In Reference [1] this is done by checking the sign of scalar

products of the distance vector and the normal and tangential vectors to the crack. A similar approach is based on the use of geometric predicates [11], where the sign of a determinant is computed to ascertain whether a point lies to the left, right, or on a line segment. The much more general level set method [22] has also been applied to LEFM problems [9,23]. It couples very well with the extended finite element formulation and enables efficient crack growth modelling. In crack modelling using level sets, the crack geometry is represented by two level set functions, which consist of signed distance functions used to specify the location of the crack. In 3D, the use of the level set method [24] or related techniques like the fast marching method for capturing crack propagation [25,26] becomes necessary.

In the present implementation, we have followed the simplest approach as described in Reference [1,27]. From a typical Abaqus input file, `.inp`, the nodal coordinates and mesh topology are saved as ASCII files `m0XY.prn`, `m0Top.prn`. A routine reads the nodal coordinates, mesh topology and geometry of cracks and computes the nodal distances to the nearest crack segment and to the crack tips. We note that the crack geometry is described in terms of line segments (restricted to straight-line segments in this work). As output of the pre-processing stage the files listed in Table 1 are generated. These files are incorporated within the Abaqus input file (described in Section 3.2) or read by the user element subroutine (described in Section 3.3):

Files listed in Table 1 contain the essential information that must be prepared to enter the Abaqus analysis stage; a more detailed description is provided at the web link given in footnote 2. The way in which intersected elements are

Table 1
Pre-processing input files for the X-FEM Abaqus analysis.

`GGnodeX`	Nodes belonging to enriched elements with corresponding signed distances.
`GGelemX`	Enriched elements with flags indicating the type of subdivision.
`GGXYC`	Coordinates of vertices that describe each crack.
`GGinfoX`	Number of cracks, maximum number of vertices for all cracks, number of enriched elements and number of their nodes.

`SETNodeX2dof`
`SETNodeX4dof`
`SETNodeX10dof`
	Set containing non-enriched node numbers belonging to enriched elements.
	The same with Heaviside enriched nodes.
	The same with crack-tip enriched nodes.

`TopNoX`	Element topology list of non-enriched elements.
`TopX`	Element topology list of enriched elements.
`TopXTypeX`	Analogous to `TopX` with enrichment type for each node.
`TopXoverlay`	Analogous to `TopX` with an increased element number to generate duplicate elements for the *overlay elements*.

subdivided is not critical (provided the subelements are convex [11]), because no inherent restrictions are placed on the shape of partitioned elements. However, a tolerance that avoids subdivision when very small regions are obtained is recommended [1,11]. Fig. 2 gives an idea of the type of subdivision carried out in this work. Another option is to use the partitioning algorithm described in Reference [28].

3.2 Structure of the input file

Once the elements and nodes to be enriched are defined, the Abaqus execution procedures are called to link the user subroutine `UEL_XFEM` that incorporates the core of the X-FEM formulation and solves the problem. In Appendix A we give an example of an input file (`.inp`) that can be used as a template[2].

[2] Files can be downloaded from `http://aim.upv.es/doc/XFEMAbq.zip`

We have made extensive use of the convenient `*nset` and `*elset` commands to group the nodes and elements in sets. We describe now the main features introduced in the input file.

Through the `*user element` command, in #1 [3] we define a 4-node user element of 12 DOFs per node (labelled as U12) to be used for all enriched elements. Abaqus admits distinct numbers of DOFs per node in the same user element, but we have chosen to set all nodes to 12 DOFs per node and then constrain the non-used DOFs at a later stage. Following Abaqus convention rules [5], DOFs numbered 1,2 are the standard 2D displacements in the x_1, x_2 directions. In Abaqus, DOFs numbered 3,4 are intended for the x_3-displacement and for the rotation about the x_1-axis. However, they will be used here for the extra DOFs associated with the Heaviside enrichment. Degrees of freedom 5–7 and 11–15 will be associated with the crack-tip enrichment. In Abaqus, DOFS 5,6 are originally meant for rotations about x_2 and x_3 axes, DOF 7 for warping amplitudes of beam sections and DOFs 11–15 for the first and successive temperatures in shell and plate elements. We do not use DOFs 8–10 (DOF 10 is not used by Abaqus). Note that the use of DOFs originally intended for other applications introduces limitations in the type of analysis and capabilities that can be performed. However, the user might define alternative DOF assignments depending on the desired type of analysis.

Other information that is provided with the `*user element` command are the number of user-defined properties: 2 real-valued properties and 5 integer-valued properties. These values are introduced in #6 as described below. The

[3] The notation # is used to indicate the section in the `.inp` file given in Appendix A.

maximum number of solution-dependent state variables per element is also given. This is set to a large number and will be used for output of magnitudes at integration points of enriched elements (stresses, Jacobians, etc.). The output is defined in #9.

In #2 the nodal coordinates and the topology of standard elements is introduced, using the files described in Section 3.1. The standard elements are grouped into the element set `ELEMTOPNOX` and also all the associated nodes into the corresponding node set.

The topology of enriched elements is introduced in #3. Here, the element set and node set are both called `ELEMTOPXU12`. They group all enriched elements and all nodes belonging to enriched elements, respectively. In #4 other convenient sets are introduced, especially those that will be used in #7 to restrict non-used DOFs in an enriched element.

In #5 the input of the overlay elements is carried out. This is not an essential step and can be omitted if desired. Note that these elements are assigned a different material (`MaterOverlay`). The material property assignment for the standard, overlay and enriched elements is done in #6. The introduction of properties for the enriched elements is done through the command `*Uel property`. The first two parameters are real-valued properties, corresponding to the Young's modulus E and Poisson's ratio ν. The same command is used to introduce five integer-valued properties: a flag indicating either plane stress or plane strain analysis, the number of integration points in enriched elements (for non-subdivided, for triangular-subdivided and for quadrilateral-subdivided elements) and the dimension of the physical domain of the prob-

lem (2D in this work). In #7 we introduce boundary conditions to constrain non-used DOFs for nodes that belong to enriched elements. This is done in accordance to the enrichment key 0, 1 or 2 (see Section 3.1).

A very important issue is the load step definition done in #8. Solving static problems would imply the usual analysis procedure `*Static`. However, we have chosen to use a coupled thermo-mechanical analysis procedure `*Coupled temperature-displacement`. In this way, Abaqus will solve for the DOFs numbered 1–7 and 11–15 simultaneously, since DOFs 11–15 are originally conceived for nodal temperatures, as indicated earlier.

Finally, output settings are specified in #9. User elements have limited capabilities in Abaqus and are not output to the `.odb` file (output data base) for plotting purposes. Therefore, only information associated with the standard (non-enriched) elements will be written to the `.odb` file. On the other hand, the solution values for the enriched DOFs can be printed to the `.dat` file: DOFs 1–7 through the label `U` and DOFs 11–15 through the label `NT` (nodal 'temperatures'). Of course, output to the `.dat` file is not necessary. However, it is very useful to output information to the binary results file `.fil` for further post-processing (e.g., for computing stress intensity factors through domain integrals). The integration point values of stresses, Jacobian, spatial derivatives of the shape functions, etc., are written as solution-dependent state variables through the label `SDV`. This is done for all the enriched elements grouped in the set `ELEMTOPXU12`. Similar information is also output to the `.fil` file for standard elements, grouped in the set `ELEMTOPNOX`.

For running an analysis including the user-subroutine, the execution procedure is of the form [5]:

`abaqus job=<`*input file name*`> user=UEL_XFEM`

This call will compile the user-subroutine `UEL_XFEM`, which constitute the core of the implementation. This subroutine is included in the Appendix B and is now briefly described. The subroutine heading and variable declarations follow the Abaqus conventions and, in addition, we have introduced new variables that are listed in Appendix B. Firstly, the real and integer properties set in #6 of the `.inp` file are read. Essentially, these correspond to the material properties and integration orders, as explained in Section 3.2. Next, information related to the pre-processing stage commented in Section 3.1 is read. This includes the number of cracks, crack-path vertices, enriched nodes and elements, type of enrichment and crack-element intersection points.

After initializing some vectors and matrices and if the element key `U12` and other conditions are fulfilled, the subroutine `int2D_X` is called. This routine defines the location of integration points according to the appropriate subdivision, and computes the total number of integration points `gint` for the current element. In the routine `TypeXelement` the keys for the enrichment type associated with the element nodes are read. Then, the routine `K_U12` computes the element stiffness matrix. Once the overall system of equations is solved, Abaqus calls again the user subroutine `UEL` and the force vector and the residual force vector are calculated at the end of the current time

increment. Finally, the stresses and other magnitudes are computed at Gauss points and stored for output to the results file `.fil` in the routine `SVARS_U12`.

The subroutine that computes the element stiffness matrix, `K_U12`, is included in Appendix C. After reading the element nodal coordinates and the stress-strain constitutive matrix (isotropic linear elastic), a loop over the total number of integration points of the enriched element is entered. First, the standard shape functions and their derivatives are computed. Then, if the enriched element contains Heaviside enriched nodes, the routine `heaviside` is called, which returns the value of the $H(\mathbf{x})$ function at the integration point plus the values of $H(\mathbf{x}_i)$, i.e. the values of H at nodes since we are using the shifted-based given by Eq. (3). Analogously, for crack-tip enriched nodes, the spatial derivatives of $(N_i F_\alpha)$ are calculated in the routine `fCrackTip`, together with the nodal values $F_\alpha(\mathbf{x}_i)$ for the shifted-basis enrichment. The strain-displacement element matrix $\mathbf{B}$ is then constructed and the element stiffness matrix $\mathbf{k}^e$ (order 48×48) is computed and returned as the Abaqus variable `AMATRX`.

Before exiting the subroutine `K_U12`, the strain-displacement element matrix $\mathbf{B}$ and the Jacobian at integration points are stored to be passed to the subroutine `SVARS_U12`. After solving the overall system of equations, Abaqus calls again the user subroutine `UEL` in order to update the solution-dependent state variables stored in `SVARS`. In the subroutine `SVARS_U12` (see Appendix D) strains, stresses, strain energy density and strain-like magnitudes (such as $\partial u_2/\partial x_1$, $\partial u_1/\partial x_2$) are computed at the integration points. These are stored in the array `SVARS`, together with the Jacobian, spatial derivatives of the shape

functions N_i and global coordinates of integration points. This information is needed for further post-processing of domain integrals to extract the stress intensity factors. It is written by Abaqus in the results binary file `.fil` through the label `SDV` as set in the input file (see Section 3.2).

3.4 Post-processing and SIF computation

Abaqus internal procedures for computing SIFs through domain integrals are not applicable to the extended finite element solution, since the information generated by user elements can not be processed by Abaqus. Therefore, we have post-processed the solution of both standard and enriched elements outside Abaqus in an external routine. After obtaining the extended finite element solution, the results file `.fil` contains all the relevant information. This file must be read according to Abaqus conventions for output [5]. A subroutine named `ABQMAIN` must be programmed for the appropriate reading. This subroutine is compiled and linked through the Abaqus execution procedure `abaqus make job=<`*subroutine file name*`>` and then run with `abaqus <`*subroutine file name*`>`. The output file reading is rather specific [5], so we have included this post-processing subroutine together with further details at the web link given in footnote 2. The subroutine file name is `ijarea`.

As is customary in 2D implementations of the X-FEM [1], the interaction integral [29,30], which is recast in domain form, is used to compute K_{I} and K_{II}, since energetic methods based on domain integrals yield accurate SIFs. Following Reference [1], the q-function used in the domain integral is an annular function defined by a radius r_q measured from the crack tip: $q = 1$ for nodes

within a circle of radius r_q and $q = 0$ for the rest of the nodes.

For the crack orientation prediction based on the values of K_{I} and K_{II}, the MTS criterion [31] (maximum tangential stress or hoop stress $\sigma_{\theta\theta}$) is used:

$$\theta_c = \arccos\left(\frac{3K_{\mathrm{II}}^2 + \sqrt{K_{\mathrm{I}}^4 + 8K_{\mathrm{I}}^2 K_{\mathrm{II}}^2}}{K_{\mathrm{I}}^2 + 9K_{\mathrm{II}}^2}\right), \tag{4}$$

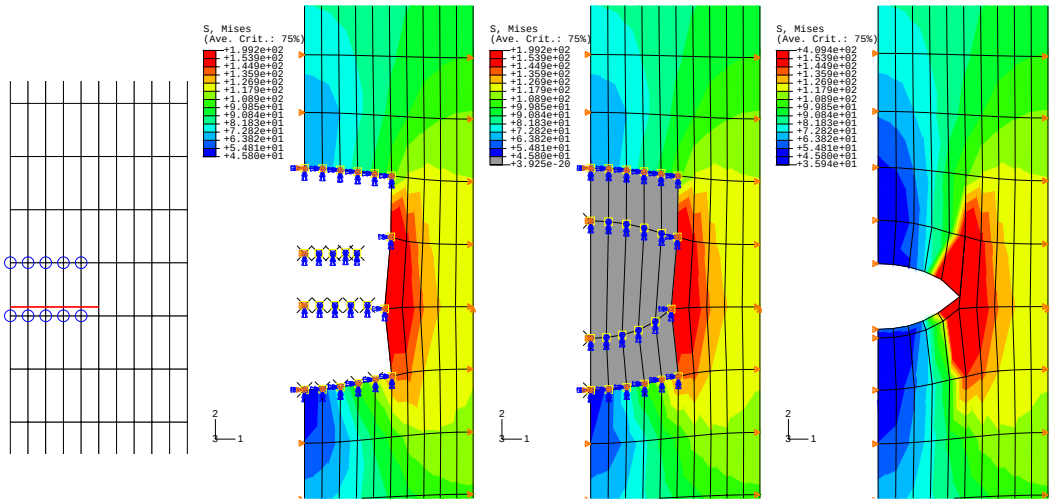
where θ_c is the angle that will follow the crack for each of the crack increments. θ_c is measured with respect to a local polar coordinate system with its origin at the crack tip and aligned with the direction of the existing crack. The sign convention is such that $\theta_c < 0$ when $K_{\mathrm{II}} > 0$ and vice-versa. Other criteria lead to very similar orientation angles for 2D problems (see a recent review in Reference [32]). Once the crack growth orientation is determined, a propagation increment Δa is added to the existing crack geometry and the analysis procedure is repeated.

3.5 Plotting

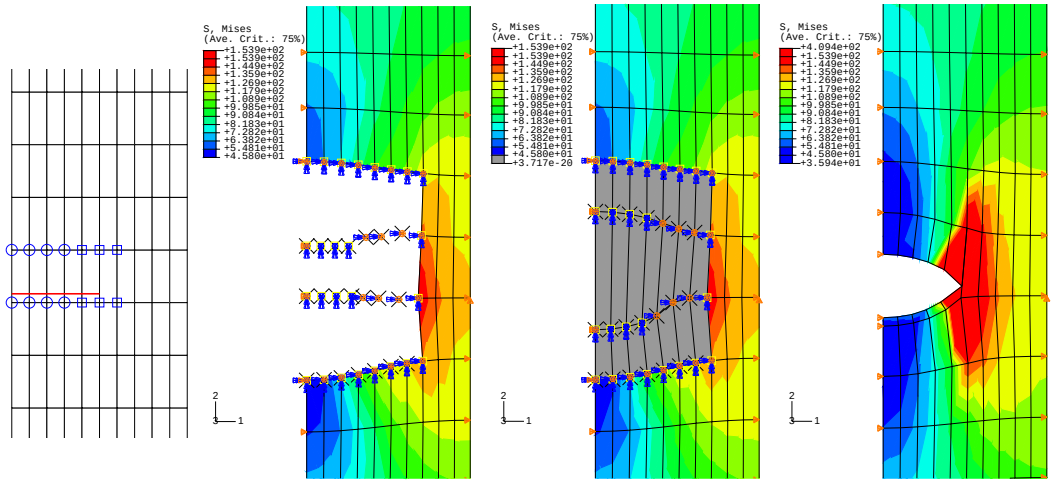
Currently, Abaqus does not have capabilities for user-element plotting because the code does not post-process the information generated by user elements. To plot the deformed shape after an extended finite element analysis, we have used standard 4-node linear elements with very small (negligible) stiffness connected to the nodes of every enriched element and retaining the same connectivity. Since a shifted-basis formulation is used (see Section 2.2) the standard DOFs of the nodes of an enriched element contain the corresponding physical displacements. As the overlay elements share the same DOFs, the

deformed shape can be visualized. Of course, the interpolation within the overlay elements is a standard bilinear interpolation and can not capture local discontinuities nor nonlinear variation of displacements due to the Heaviside and crack-tip enrichment functions. For the same reason and since overlay elements have negligible stiffness, stress or strain plots within overlay elements do not represent the correct variations.

Fig. 3 shows a portion of a cracked finite strip loaded under uniform normal stress. The lateral sides are constrained in the x_1-direction and therefore this model represents a sequence of infinite collinear cracks in a plate loaded in tension. The crack location and the enriched nodes are shown in the sketch on the left for two types of enrichment: only Heaviside enrichment and a full X-FEM enrichment including crack-tip functions. Three contour plots of the von Mises stress field are represented for each type of enrichment. From left to right, these show the extended finite element solution without a shifted-basis formulation, the extended finite element solution with a shifted-basis formulation plus overlay elements and a standard FE solution for comparison purposes. The DOF constraints are plotted on the enriched nodes according to the type of enrichment (each type of enrichment implies the constraint of the non-used DOFs as explained in Section 3.2). The shifted basis allows the representation of the true location of the nodes, with overlay elements that help to visualize the discretization. The enriched nodes location can be compared with the standard FE solution, shown on the rightmost plot of Fig. 3. Note that for the FE solution, a constraint equation for the node located at the crack tip was included to make the displacement field compatible with the



(a) Only Heaviside enrichment



(b) Heaviside and crack-tip enrichment

Fig. 3. Plotting enriched elements in Abaqus. From left to right: crack location and enriched nodes; von Mises contour plot without a shifted-basis formulation; the same contour plot with a shifted-basis formulation and overlay elements; comparison with a standard FE solution.

neighbouring side. For the enrichment with only Heaviside functions, Fig. 3(a), the extended finite element and FE solutions provide exactly the same DOF solution [4]. As expected, it can be observed in Fig. 3(b) that the extended finite element stress distribution is not the same as the FE solution, since the

[4] Strictly speaking, the von Mises stress distribution is slightly different near the nodes that belong to enriched elements due to the Abaqus averaging procedure, which cannot take into account the true user element results.

former includes the effect of the crack-tip enrichment functions.

3.6 Contact problem

Abaqus capabilities are limited insofar as user elements can not form part of a contact surface. The use of overlay elements can be of interest in applications in which the enriched elements (user-elements) must form part of contact surfaces. This situation arises when other bodies contact near a surface-breaking crack, as in fretting fatigue.

We have overcome this shortcoming using the overlay elements with a negligible relative stiffness as described earlier. Overlay elements are used to establish a master surface for the NTS ("node-to-segment") Abaqus contact algorithms. Since these elements share the same nodes as the enriched elements, displacements associated with an overlay element are governed by the enriched element. Therefore, it is crucial that the nodal physical displacements correspond to the standard DOFs, i.e., the first two DOFs of an enriched node. The shifted basis enables this feature. Obviously, a small displacement assumption must be considered, because the displacement interpolation along the sides of an overlay element is linear.

We have used this approach for the numerical example that appear in Section 4.2.4. It has also been used in Reference [33], where a variant of this Abaqus implementation with a different enrichment basis is developed. This enables the enrichment of other type of singularities within the framework of the partition of unity finite element method, such as one that arises at the

end of a squared ended contact zone under a sliding condition.

4 NUMERICAL EXAMPLES

4.1 Westergaard's crack problem

In order to assess the accuracy of the proposed implementation, a problem with an exact reference solution has been solved for a sequence of uniformly refined meshes. The problem analyzed is an infinite plate with a crack of finite length $2a$, biaxially loaded with remote uniform tractions (see Fig. 4). The exact solutions for the SIFs for this problem are: $K_{\mathrm{I,ex}} = \sigma\sqrt{\pi a}$ and $K_{\mathrm{II,ex}} = \tau\sqrt{\pi a}$.

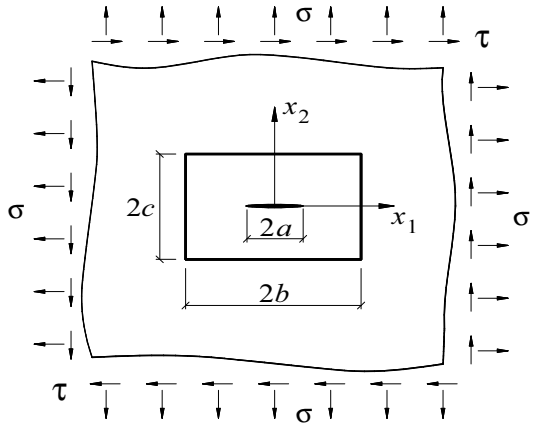


Fig. 4. Westergaard's crack problem.

The Westergaard's solution to the exact stress fields at any point of the plate can be expressed in terms of stress functions (see Reference [34]). In Reference [35], we presented explicit expressions for the stress fields in terms of the spatial coordinates, which enables the computation of equivalent nodal forces for a finite portion of the domain. We note in passing that the problem presented here does not simply correspond to the pure singular asymptotical

stress field, but includes all the terms of the series expansion. For the biaxial loading with remote uniform traction σ, the stress field at a point (x_1, x_2) associated with mode I loading is:

$$\sigma_{11}^{\mathrm{I}} = \frac{\sigma}{\sqrt{|t|}}\left[\left(x_1\cos\frac{\phi}{2} - x_2\sin\frac{\phi}{2}\right) + x_2\frac{a^2}{|t|^2}\left(m\sin\frac{\phi}{2} - n\cos\frac{\phi}{2}\right)\right], \tag{5a}$$

$$\sigma_{22}^{\mathrm{I}} = \frac{\sigma}{\sqrt{|t|}}\left[\left(x_1\cos\frac{\phi}{2} - x_2\sin\frac{\phi}{2}\right) - x_2\frac{a^2}{|t|^2}\left(m\sin\frac{\phi}{2} - n\cos\frac{\phi}{2}\right)\right], \tag{5b}$$

$$\sigma_{12}^{\mathrm{I}} = x_2\frac{a^2\sigma}{|t|^2\sqrt{|t|}}\left(m\cos\frac{\phi}{2} + n\sin\frac{\phi}{2}\right), \tag{5c}$$

and for loading with remote uniform traction τ (antisymmetric mode or mode II) the stress fields at points (x_1, x_2) belonging to the half plane $x_1 \geq 0$ are given by

$$\sigma_{11}^{\mathrm{II}} = \frac{\tau}{\sqrt{|t|}}\left[2\left(x_2\cos\frac{\phi}{2} + x_1\sin\frac{\phi}{2}\right) - x_2\frac{a^2}{|t|^2}\left(m\cos\frac{\phi}{2} + n\sin\frac{\phi}{2}\right)\right], \tag{6a}$$

$$\sigma_{22}^{\mathrm{II}} = x_2\frac{a^2\tau}{|t|^2\sqrt{|t|}}\left(m\cos\frac{\phi}{2} + n\sin\frac{\phi}{2}\right), \tag{6b}$$

$$\sigma_{12}^{\mathrm{II}} = \frac{\tau}{\sqrt{|t|}}\left[\left(x_1\cos\frac{\phi}{2} - x_2\sin\frac{\phi}{2}\right) + x_2\frac{a^2}{|t|^2}\left(m\sin\frac{\phi}{2} - n\cos\frac{\phi}{2}\right)\right], \tag{6c}$$

where $m, n, |t|$ and ϕ, which are real-valued functions of x_1, x_2, are defined as

$$m = \mathrm{Re}\, t = x_1^2 - x_2^2 - a^2, \tag{7}$$

$$n = \mathrm{Im}\, t = 2x_1x_2, \tag{8}$$

$$|t| = |m + \mathrm{i}n| = \sqrt{m^2 + n^2}, \tag{9}$$

$$\phi = \arg\bar{t} = \arg(m - \mathrm{i}n) \qquad \text{with } \phi \in [-\pi, \pi]. \tag{10}$$

The crack length is $a = 1$ and the dimensions of the finite portion of the domain

are $b = 2a$, $c = a$. Five uniform meshes have been considered, with element sizes $h = a/4$, $a/8$, $a/16$, $a/32$ and $a/64$. The nodal equivalent forces applied on the boundary of the model are computed for the remote loads σ, τ that yield $K_{\mathrm{I,ex}} = K_{\mathrm{II,ex}} = 1$. These nodal equivalent forces are sketched in Fig. 5 for the third mesh of the refinement sequence. The x_1- and x_2-displacements are constrained at the crack tip and an anti-symmetry constraint equation is imposed between points of the x_2-axis to avoid rigid body rotation [35]. The Young's modulus is $E = 10^7$ (units of pressure), the Poisson's ratio is $\nu = 0.333$ and plane stress condition is assumed.

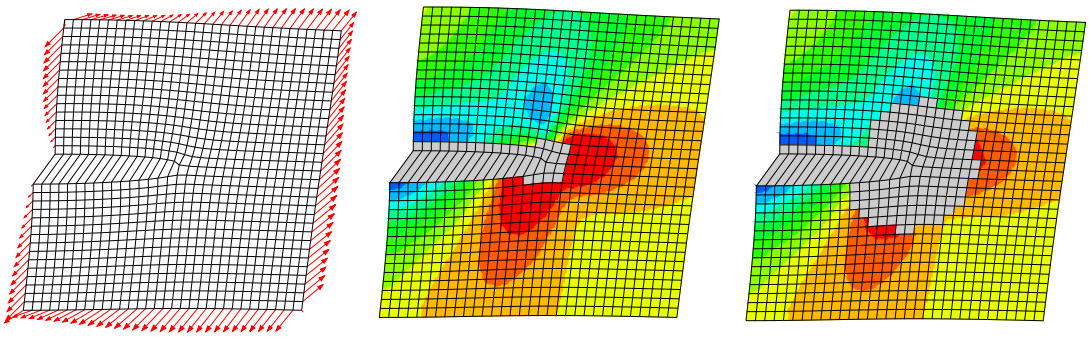


Fig. 5. Westergaard's crack problem. Third mesh of a sequence of uniformly refined meshes. von Mises contour plot using an X-FEM topological enrichment (center) and a geometric enrichment (right).

For the extended finite element solution, the crack location has been chosen to end at a node to simplify the application of the displacement boundary conditions. To verify the accuracy of the SIFs with the proposed implementation, two enrichment schemes for the crack-tip functions have been tested: the standard topological enrichment (Fig. 5, center) and the geometric enrichment (Fig. 5, right). The geometric enrichment follows the strategy presented in References [19] and [20], i.e., the crack-tip enriched nodes are those located within a fixed area surrounding the crack tip. The chosen fixed region is a circle of radius $0.38a$. For numerical integration in enriched elements, we have used

5×5 Gauss quadrature for quadrilateral subdomains and 73-point quadrature for triangular subdomains (adjacent to the crack-tip).

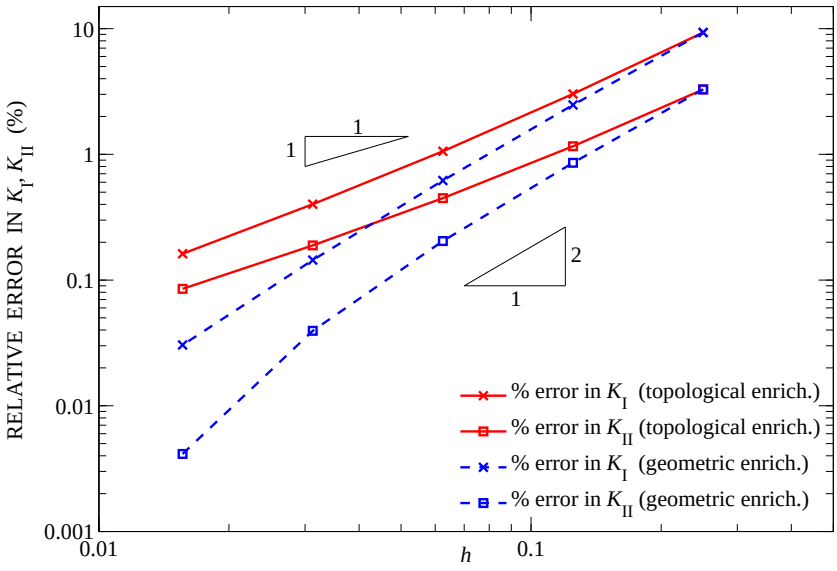


Fig. 6. Westergaard's crack problem. Relative error in K_{I} and K_{II} (in %).

The relative error obtained for both K_{I} and K_{II} (in percent) is plotted in Fig. 6. As reported in the literature, it can be seen that the error in the SIFs is in general very low, due to the enrichment with crack-tip functions. As in previous studies [19,20], the effect of the singularity on the convergence rate is only removed if geometric enrichment is introduced. It is well-known that the error in energy norm of a standard FE solution for a singular problem with uniform mesh refinement is bounded by

$$\|\mathbf{e}\|_E \leq C h^{\min(p,\lambda)}, \tag{11}$$

where $\mathbf{e} = \mathbf{u} - \mathbf{u}^h$ is the error in displacements introduced by the finite element approximation, C is a constant that depends on the problem, h is the characteristic element size, p is the order of the elements used in the discretization

($p = 1$ in this work) and λ is the order of the singularity ($\lambda = 0.5$ in LEFM). The square of the error in the energy norm is related to the error in strain energy and therefore to the error in the strain energy release rate G and the error in the SIFs [36]:

$$e_{(K)} \leq C^2 h^{2\min(p,\lambda)}, \tag{12}$$

where $e_{(K)} = K_{\text{ex}} - K^h$ is the error in the SIF. Therefore, if the effect of the singularity is not removed, the expected convergence rate is 1 ($p = 1$, $\lambda = 0.5$). If the effect of the singularity is removed with geometric enrichment then Eq. (12) simply reduces to $e_{(K)} \leq C^2 h^{2p}$ and the convergence rate is increased to 2. The results in Fig. 6 are in good agreement with these *a priori* estimates.

4.2 Crack propagation under mixed mode conditions

The following examples reveal the merits of the proposed implementation in Abaqus to simulate mixed-mode crack growth under quasi-static conditions. The crack orientation angle is governed by the values of K_{I} and K_{II} and is computed through Eq. (4). Note that current capabilities for crack growth in Abaqus (Version 6.7) are limited to propagation between two distinct initially bonded contact surfaces, which must be defined *a priori* by the user. Therefore, the incorporation of X-FEM substantially complements and enhances existing Abaqus options.

4.2.1 Eccentric crack in cantilever beam

This problem is considered in References [18,14] and is illustrated in Fig. 7. It is known that the crack propagation of an initial crack a_0 located slightly off the midplane follows a path that departs away from the initial plane. The dimensions of the problem are $a_0 = 2$, $w = a_0$ and $l = 3a_0$. Plane strain condition is assumed with material properties as in Section 4.1 and the concentrated load is $P = 1$ (units of force). We have solved for two initial crack locations whose offset from the midplane is $\pm 0.035w$. Eleven crack growth increments have been considered, with a fixed value of $\Delta a = 0.05a_0$. Fig. 7 shows a detailed view of the enriched nodes for the last increment of the propagation. The Abaqus von Mises plots show the expected crack growth pattern for the two initial cracks considered, which is in qualitative agreement with previously reported results [14].

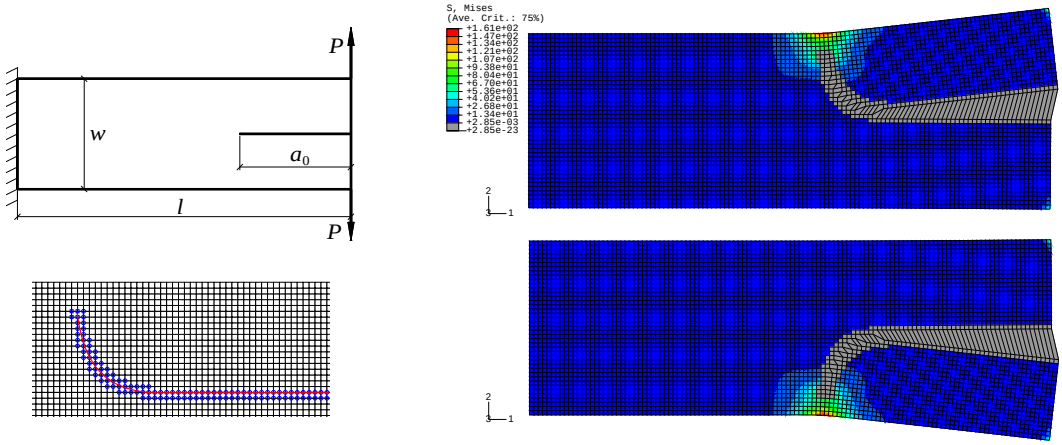


Fig. 7. Slightly eccentric crack in a cantilever beam. A detailed view of the enriched nodes for the last crack growth increment is shown.

4.2.2 Crack in a plate with a hole

The problem shown in Fig. 8 is an adaptation of an example presented in Reference [10]. The initial crack length is $a_0 = 10$ mm (all dimensions in the sketch of Fig. 8 are given in mm), the material is aluminum 7075-T6, with $E = 71.7$ GPa, $\nu = 0.33$ and a plane strain state is considered. The load applied for the extended finite element analysis is $P = 20$ kN, and linear elastic material behaviour is assumed. The analysis of quasi-static crack propagation has been carried out with twelve crack increments of $\Delta a = 3$ mm each. Fig. 8 shows the crack propagation path obtained after twelve crack increments and a plot of the von Mises stress field.

For this problem, we carried out experimental tests with specimens 16 mm thick in order to compare the crack path to the one predicted numerically. Fatigue tests were performed to produce a smooth crack growth with a stress ratio of $R = 0.1$. The applied alternating load was reduced as the crack grew to avoid premature rupture. This does not affect the crack propagation path, since the latter is based on the MTS criterion Eq. (4) that, in turn, depends on the ratio $K_{\mathrm{II}}/K_{\mathrm{I}}$ only. The applied load yields proportional values of K_{I} and K_{II}, and therefore the predicted orientation angle is independent of the applied load. It can be seen that the crack trajectory obtained experimentally is in good agreement with the numerically predicted path.

4.2.3 Growth of multiple cracks

Fig. 9 shows the application of the present implementation to a specially devised problem with multiple cracks. The crack growth modelling of both

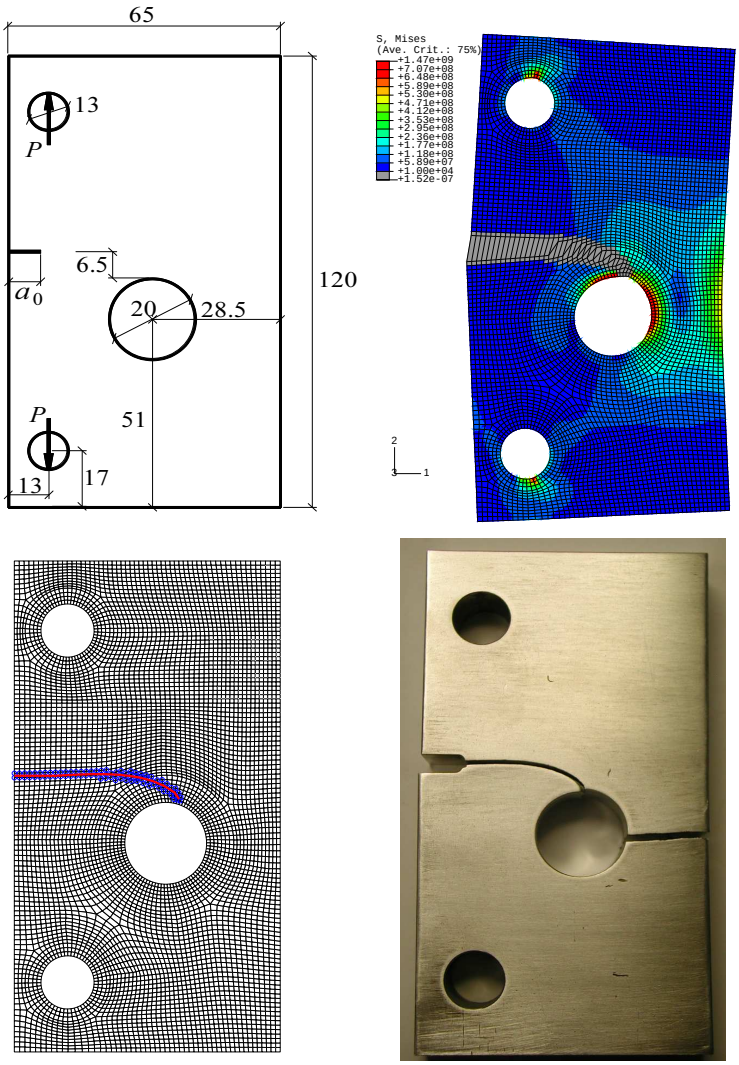


Fig. 8. Crack in a plate with a hole. The crack growth path predicted using X-FEM agrees well with experimental observations.

internal cracks and surface cracks emanating from stress raisers is considered. An initial crack length of $a_0 = 4$ is assumed for all surface cracks (cracks denoted from C to I). The initial crack length for the internal cracks (cracks A and B) is $2a_0$ and their location is given by the coordinates $(-20.5, -16.5)$ for the lower tip of crack A and $(35, 5)$ for the upper tip of crack B. The initial orientations of cracks A to I measured from the x_1-axis are 40^o, -165^o, 10^o, -45^o, 180^o, 35^o, -145^o, 165^o and 25^o respectively. Plane stress condition is assumed with material properties as in Section 4.1. The applied tractions are

$\sigma_1 = 500$ and $\sigma_2 = 300$ (units of pressure).

Five quasi-static crack growth increments have been considered, with $\Delta a = 3.5$. Of course, both tips of internal cracks A, B are allowed to grow at each increment. The final configuration is shown in Fig. 9 (bottom right). Although no quantitative results for this configuration are available for the purpose of comparison, a qualitative assessment can be made. For instance, for the crack A, it can be seen that the crack orientation starting from both tips is normal to the maximum principal stress direction (in this zone, close to the x_2-direction due to the near influence of boundary conditions). This is in accordance with expectations, since the MTS criterion is applied to predict the crack orientation.

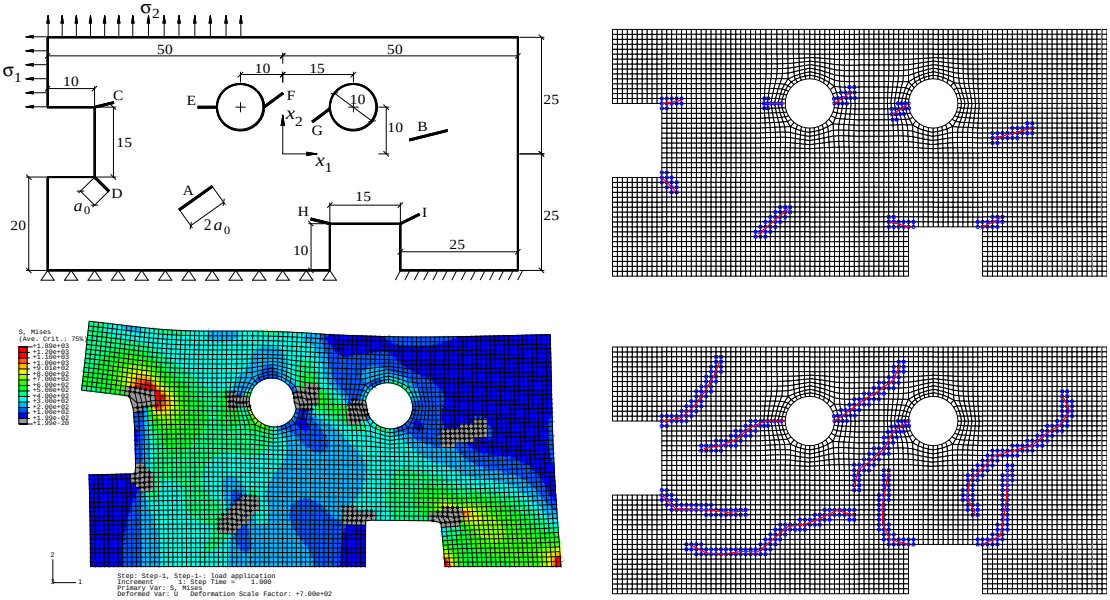


Fig. 9. Crack growth modelling of multiple cracks (internal and surface cracks). Plots on the right hand side show the initial crack locations and the corresponding crack paths after five quasi-static growth increments.

4.2.4 Application to fretting fatigue

One of the applications of the present implementation of the X-FEM is to fretting-fatigue problems [6]. This phenomenon is characterized by the presence of two or more contacting bodies, subjected to relative displacements of small amplitude. The stresses exhibit steep gradients in the vicinity of the contact zone and the combination of such a high stress gradient with the cyclic nature of the loading leads to the nucleation of small cracks and their eventual propagation. The modelling of the crack propagation stage with X-FEM and the Abaqus implementation greatly simplifies these analyses, in which there are interactions between the stress contact field and the crack. Note that previous work in the literature introduce simplifications in the model to be solved [6]. Some of the studies assume a fixed analytical distribution of the contact stresses, which can not take into account the effect of the crack presence on the contact distribution. Others consider analytical models assume that the boundary borders are remote. There are also standard FE models that simplify the crack growth to a normal straight crack to avoid the complications associated with remeshing. All these limitations are overcome with the proposed X-FEM implementation in Abaqus.

Fig. 10 shows a fretting example of a cylindrical (Hertzian) contact on a flat specimen under the action of a normal distributed load p. The specimen is subjected to a fatigue load σ_{B} (bulk stress). An initial crack a_0 is located at the end of the contact zone of width $2a_{\mathrm{H}}$. The radius of the contacting cylinder is $r_{\mathrm{pad}} = 25$ mm, the specimen half-thickness is $w = 20$ mm and the normal distributed load is equivalent to a vertical constant force of 40 kN. This load

produces a contact region of semi-width $a_{\mathrm{H}} = 177.9\,\mu$m, having assumed the same material as in Section 4.2.2. The maximum value of the bulk stress is $\sigma_{\mathrm{B,max}} = 90$ MPa and the crack is slanted –105º with respect to the x_1-axis with a length of $a_0 = 200\,\mu$m.

As explained above, the contact between the indenter and the enriched elements that enter into contact is carried out by means of the overlay elements. The von Mises contour plot reveals the strong crack-contact interaction that exists at the first stages of the crack growth. It is verified that the interaction effect can modify the SIF values. As the SIF range is raised to a power (greater than 3 for aluminum 7075-T6) in the typical crack growth models (Paris law and similar), an accurate SIF estimation is desirable in order to minimize the error in the estimated life. For further details, we refer to Reference [6] where a specific study on the fretting application is presented. The contact procedures and non-linear solver capabilities of Abaqus are thus combined with the advantages of X-FEM for crack modelling.

5 CONCLUDING REMARKS

In this paper, we presented a procedure for the implementation of the X-FEM within the commercial FE code Abaqus for two-dimensional fracture problems. The implementation was based on the user element subroutine `UEL` and enables the modelling of different crack locations and orientations using a single mesh that is easily generated. In addition, use of the crack-tip enrichment significantly improved the accuracy of the computed SIFs. We focused

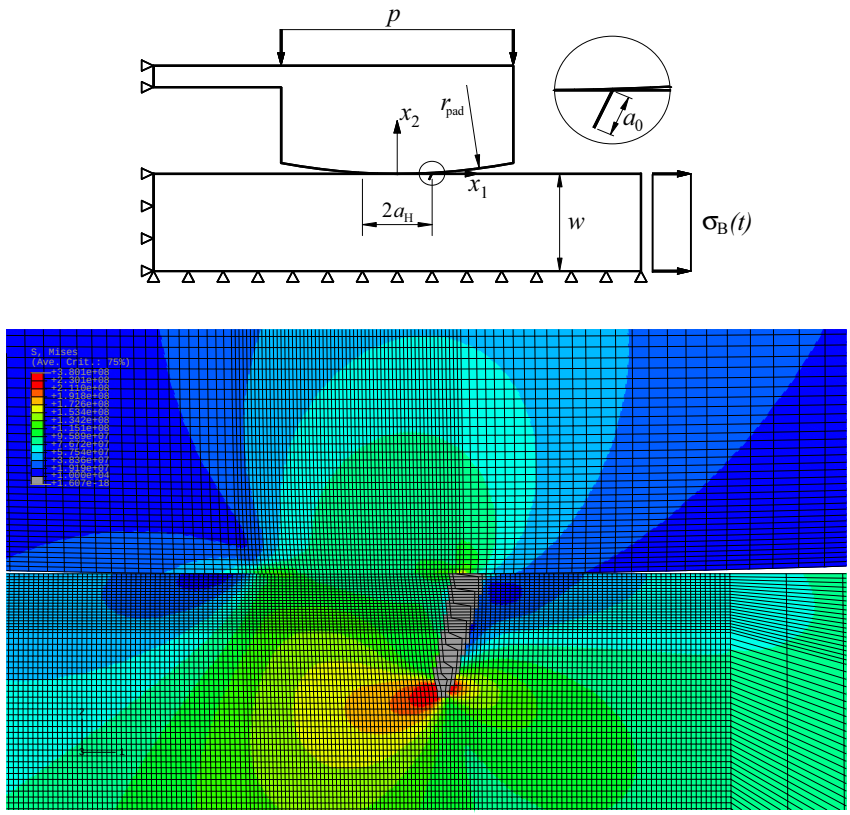


Fig. 10. Application to fretting fatigue. Enlarged view of the von Mises contour plot of a cylindrical indenter contacting a cracked specimen.

on the main procedures that interact with Abaqus: the structure of the input file, the user subroutine for the enriched elements, the element stiffness matrix computation and the outputs for further post-processing. These routines are in the open-source, and therefore fracture mechanics analysts can use and adapt these procedures. Various numerical examples in fracture mechanics were solved to demonstrate the accuracy and reveal the merits of the implementation. Through these examples, different issues were addressed, such as the convergence rates using topological versus geometric enrichment, crack propagation path under mixed-mode conditions, ability to analyze multi-cracked components, and application to cracks emanating from contact stress raisers under fretting-fatigue conditions.

Acknowledgements

The authors wish to thank the Ministerio de Ciencia y Tecnología for the support received in the framework of the project DPI2007-66995-C03-02. The financial support received from the Vicerrectorado de Innovación y Desarrollo (Polytechnical University of Valencia) and from the Conselleria d'Empresa, Universitat i Ciència (Generalitat Valenciana) is also gratefully acknowledged. This support enabled the research visit of E. Giner to UC Davis, where parts of this research were accomplished.

A Template of the input file `.inp`

```
*Heading
TEMPLATE FOR X-FEM WITH ABAQUS
**
** =====================================================
** #1. USER ELEMENT DEFINITION (CALLED U12, 12 DOF/NODE)
** =====================================================
*User element, nodes=4, type=U12, properties=2, iproperties=5, coordinates=2, variables=9000
1,2,3,4,5,6,7,11,12,13,14,15
**
** =====================================================
** #2. NODES & STANDARD ELEMENTS
** =====================================================
** -- ALL NODES
*Node, input=.\files\m0XY.prn
** -- ONLY THE NON-ENRICHED ELEMENTS (STANDARD ELEMENTS)
*Element, type=CPS4, input=.\files\TopNoX, elset=ELEMTOPNOX
*Nset, nset=ELEMTOPNOX, elset=ELEMTOPNOX
*Solid Section, elset=ELEMTOPNOX, material=Material-1
1.
**
** =====================================================
** #3. ENRICHED ELEMENTS (USER ELEMENTS)
** =====================================================
*Element, type=U12, input=.\files\TopX, elset=ELEMTOPXU12
*Nset, nset=ELEMTOPXU12, elset=ELEMTOPXU12
**
** =====================================================
** #4. NSETS & ELSETS INCLUSION
** =====================================================
** -- NODE SETS BELONGING TO ENRICHED ELEMENTS
*Nset, nset=NodeX2dof
*include,input=.\files\SETNodeX2dof
*Nset, nset=NodeX4dof
*include,input=.\files\SETNodeX4dof
*Nset, nset=NodeX10dof
*include,input=.\files\SETNodeX10dof
**
** -- TO INCLUDE OTHER SETS (SPECIFIC TO THE PROBLEM)
*include,input=.\files\m0sets.prn
**
** =====================================================
** #5. OVERLAY ELEMENTS (if desired)
** =====================================================
*Element, type=CPS4, input=.\files\TopXoverlay, elset=ElemTopXoverlay
*Solid Section, elset=ElemTopXoverlay, material=MaterOverlay
1.
**
```

```
** =====================================================
** #6. MATERIALS DEFINITION
** =====================================================
*Material, name=Material-1
*Elastic
1e7, 0.333
*Material, name=MaterOverlay
*Elastic
1e-14, 0.333
** (negligible stiffness)
**
** -- USER DEFINED PROPERTIES FOR USER ELEMENTS. Keys:
**   1st Parameter: E (Young's modulus)
**   2nd Parameter: nu (Poisson's ratio)
**   3rd Parameter: plane stress = 1; plane strain = 2
**   4th Parameter: orderC(1) = Quadrature order for quadrilaterals (in each direction)
**              NOTE: only for enriched elements no subdivided (quadrilaterals)
**   5th Parameter: orderC(2) = Quadrature order for triangles (total points)
**              NOTE: only for enriched elements subdivided into triangles
**   6th Parameter: orderC(3) = Quadrature order for quadrilaterals (in each direction)
**              NOTE: only for enriched elements subdivided into 2 quadrilaterals (elemX = type 4)
**   7th Parameter: Dimension of the physical domain of the problem: 2=2D
*Uel property, elset=ELEMTOPXU12
1.0e7, 0.333,1,5,7,5,2
**
** =====================================================
** #7. BOUNDARY CONDITIONS
** =====================================================
**  -- BC (USED IN X-FEM TO CONSTRAIN THE NON-USED EXTRA DOF)
*Boundary, OP=MOD
NodeX2dof, 3, 15, 0.0
NodeX4dof, 5, 15, 0.0
NodeX10dof, 3, 4, 0.0
**
** -- OTHER BC (SPECIFIC TO THE PROBLEM)
** ... etc ...
**
** =====================================================
** #8. LOAD STEPS
** =====================================================
*Step
Step-1-: load application
** -- WE DON'T USE *Static. WE USE THE FOLLOWING TO INVOLVE DOFS 1-7 & 11-15
*coupled temperature-displacement, steady state
1., 1., 1e-05, 1.
**
** -- LOADS (SPECIFIC TO THE PROBLEM)
** ... etc ...
**
** =====================================================
** #9. OUTPUT FILES
** =====================================================
** -- OUTPUT FIELD TO .odb
*Output, field, op=NEW, frequency=1
*Node Output
U
*Element Output
S, E
** -- OUTPUT PRINT TO .dat (not necessary)
** -- 'U' TO LIST DOFS 1-7 & 'NT' TO LIST TEMP. ASSOCIATED DOFS 11-...
*Node print, nset=ELEMTOPXU12,frequency=1
U
*Node print, nset=ELEMTOPXU12,frequency=1
NT
** -- OUTPUT WRITE TO  .fil
*Node file, nset=ELEMTOPNOX
COORD,U
*El file, elset=ELEMTOPNOX, POSITION=INTEGRATION POINT
S,E,ENER,IVOL
** -- TO WRITE USER DEFINED OUTPUT VARIABLES FOR POST-PROCESSING
** -- (INFORMATION AT GAUSS POINTS OF ENRICHED ELEMENTS)
*El file, elset=ELEMTOPXU12
SDV
**
*End Step
```

B User subroutine UEL_XFEM

```
      SUBROUTINE UEL(RHS,AMATRX,SVARS,ENERGY,NDOFEL,NRHS,NSVARS,
     * PROPS,NPROPS,COORDS,MCRD,NNODE,U,DU,V,A,JTYPE,TIME,DTIME,
     * KSTEP,KINC,JELEM,PARAMS,NDLOAD,JDLTYP,ADLMAG,PREDEF,NPREDF,
     * LFLAGS,MLVARX,DDLMAG,MDLOAD,PNEWDT,JPROPS,NJPROP,PERIOD)
C
```

```
      INCLUDE 'ABA_PARAM.INC'
c
c     User subroutine for computation of element stiffness matrix and
c     the element force vector

c     ABAQUS defined variables:
      DIMENSION RHS(MLVARX,*), AMATRX(NDOFEL,NDOFEL), PROPS(*),
     * SVARS(NSVARS), ENERGY(8), COORDS(MCRD,NNODE), U(NDOFEL),
     * DU(MLVARX,*), V(NDOFEL), A(NDOFEL), TIME(2), PARAMS(*),
     * JDLTYP(MDLOAD,*), ADLMAG(MDLOAD,*), DDLMAG(MDLOAD,*),
     * PREDEF(2,NPREDF,NNODE), LFLAGS(*), JPROPS(*)
c
c     Important variables (list not exhaustive)
c     JELEM  Current element number
c     AMATRX Element stiffness matrix (element contribution to the stiffness
c            matrix of the overall system of equations)
c     RHS    Element residual force vector (element contribution to the right-hand-side
c                 vector of the overall system of equations)
c     F      Element force vector (AMATRX times the updated solution for nodal dofs)
c     E      Young's modulus
c     Nu     Poisson's ratio
c     PSS    1 - Plane stress
c            2 - Plane strain
c     orderQ Vector that stores the following quadrature orders:
c     orderQ(1) = Quadrature order for quadrilaterals (in each direction)
c            NOTE: only for non-subdivided enriched elements (quadrilaterals)
c     orderQ(2) = Quadrature order for triangles (total points)
c            NOTE: only for enriched elements subdivided into triangles
c     orderQ(3) = Quadrature order for quadrilaterals (in each direction)
c            NOTE: only for enriched elements subdivided into 2 quadrilaterals (elemX = type 4)
c     dimens Dimension of the physical domain: 2=2D
c            Actually, it should suffice with the ABAQUS variable MCRD, but ABAQUS automatically
c            sets MCRD=3, even though for a 2D problem, since we are using the third and further
c            available dof for the enriched nodes.
c     NNODE  Number of nodes per element
c     NelmX  Number of enriched elements
c     NnodX  Number of nodes that belong to enriched elements
c     TypeX  Matrix that stores the key number to the type of node in an enriched element:
c              0 - Non-enriched (2 dof)
c              1 - Heaviside enrichment (4 dof)
c              2 - Crack tip enrichment (10 dof)
c     TypeXe Vector that stores the key in TypeX for the nodes of the current element
c     ix     Vector that stores the node numbers of the current element (connectivity)
c     Xe(8)  X nodal coordinates of the current element
c                 (it is duplicated to ease the counting from the 4th to the 1st node)
c     Ye(8)  Y nodal coordinates of the current element
c     NCracks Number of cracks
c     NCP    Number of crack path points (vertices)
c     maxNCP Maximum number of crack path points (vertices)
c     XYC    Matrix that stores the coordinates of crack path points
c     XYC0   Crack tip coordinates associated with the crack tip enriched element
c     XYCPrev Crack tip coordinates associated with the previous crack path point
c     gint   Total number of integration points (either with or without subdivision)
c     flag   Subdivision indicator (1 for subdivision)
c     mpg    Maximum expected number of integration points for an enriched element
c     sg     Matrix that stores the coordinates and weights of the integration points
c     xypg   Matrix that stores the coordinates of the integration points
c     Dist   Matrix that stores distances to crack from nodes of enriched elements
c            (this information is previously preprocessed, for example in Matlab)
c     ElemGG Matrix that stores information about the elements to be enriched, type of
c            crack intersection and points of interesection
c            (this information is previously preprocessed, for example in Matlab)
c     BatG   Matrix that stores the [B] matrix for each enriched element at Gauss points
c     DBatG  Matrix that stores the [D][B] matrix for each enriched element at Gauss points
c     JatG   Vector that stores the Jacobian = det([J]) for each enriched element at Gauss points
c
      Declaration of variables for XFEM user element
      CHARACTER*256 OUTDIR  ! to read the working directory
      INTEGER LENOUTDIR  ! working directory string length
      INTEGER i,j,k,PSS,orderQ(3),gint,flag,dimens
      INTEGER NCracks,maxNCP,NelmX,NnodX,TypeXe(NNODE),ix(NNODE)
      INTEGER,PARAMETER :: mpg=1650 ! up to more than 40x40 Gauss integration points per element
      INTEGER,ALLOCATABLE:: TypeX(:,:),NCP(:)
      REAL*8 E, Nu
      REAL*8 F(NDOFEL)
      REAL*8 sg(3,mpg),xypg(2,mpg),Xe(8),Ye(8),XYC0(2),XYCPrev(2)
      REAL*8, ALLOCATABLE:: XYC(:,:,:),Dist(:,:),ElemGG(:,:)
      REAL*8, ALLOCATABLE:: BatG(:,:),DBatG(:,:),JatG(:)
c     &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
c     &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
c
c     Read real and integer properties set at the ABAQUS input file
      E = PROPS(1)
      Nu = PROPS(2)
      PSS = JPROPS(1)
      orderQ(1) = JPROPS(2)
```

```
      orderQ(2) = JPROPS(3)
      orderQ(3) = JPROPS(4)
      dimens = JPROPS(5)
c     Read the working directory
      CALL GETOUTDIR(OUTDIR,LENOUTDIR)
c     ************************************************************************
c     **** Read information previously preprocessed, for example in Matlab ****
c     ************************************************************************
c     Read number of cracks, max number of crack path points,
c          number of enriched elements and enriched nodes.
      OPEN(68,FILE=OUTDIR(1:LENOUTDIR)//'\files\GGInfoX')
      READ(68,*) NCracks,maxNCP,NelmX,NnodX
      CLOSE(68)

c     Allocate dimensions
      ALLOCATE (TypeX(NnodX,2), NCP(NCracks))
      ALLOCATE (XYC(NCracks,maxNCP,2), Dist(NnodX,3), ElemGG(NelmX,10))

c     Read coordinates of path points for each crack
      OPEN(68,FILE=OUTDIR(1:LENOUTDIR)//'\files\GGXYC')
      DO i=1,NCracks
        READ(68,*) NCP(i)
        DO j=1,NCP(i)
          READ(68,*) (XYC(i,j,k),k=1,2)
        END DO
      END DO
      CLOSE(68)

c     Read list of enriched nodes, type of enrichment and distances
      OPEN(68,FILE=OUTDIR(1:LENOUTDIR)//'\files\GGnodeX')
      DO i=1,NnodX
        READ(68,*) (TypeX(i,j),j=1,2),(Dist(i,j),j=2,3)
        Dist(i,1)=TypeX(i,1)
      END DO
      CLOSE(68)

c     Read list of enriched elements, type of enrichment and intersection points
      OPEN(68,FILE=OUTDIR(1:LENOUTDIR)//'\files\GGelemX')
      DO i=1,NelmX
        READ(68,*) (ElemGG(i,j),j=1,10)
      END DO
      CLOSE(68)

c     Call initializing routines for matrix and vectors
      CALL initializeM(RHS,NDOFEL,NRHS)
      CALL initializeM(AMATRX,NDOFEL,NDOFEL)
      CALL initializeV(ENERGY,8)
      CALL initializeV(SVARS,NSVARS)

c     Verification of element type (type=12 for enriched element)
      IF (JTYPE.EQ.12) THEN
c      ************************************
c      *    4 NODE ENRICHED ELEMENT WITH    *
c      *    UP TO 12 DOF/NODE FOR X-FEM     *
c      ************************************

       IF (LFLAGS(1).EQ.71) THEN
c      Coupled thermal-stress, steady state analysis
        IF (LFLAGS(3).EQ.1) THEN
c       Normal implicit time incrementation procedure.
c       User subroutine UEL must define the residual vector in RHS
c       and the stiffness matrix in AMATRX

c         Routine that defines the location of integration points according to
c         the appropriate subdivision. This enables to know the total number of
c         integration points for the current element, stored in gint, and whether
c         the element is subdivided for integration (flag=1) or not.
          CALL int2d_X(JELEM,NelmX,ElemGG,MCRD,NNODE,COORDS,orderQ,
     *          NCracks,maxNCP,NCP,XYC,gint,sg,Xe,Ye,flag,mpg,xypg,
     *          XYC0,XYCPrev)

c         Allocate dimensions once the total number of integration points gint is known
          ALLOCATE(BatG(3*gint,NDOFEL),DBatG(3*gint,NDOFEL),JatG(gint))
          CALL initializeM(BatG,3*gint,NDOFEL)
          CALL initializeM(DBatG,3*gint,NDOFEL)
          CALL initializeV(JatG,gint)

c         Search of the enrichment type for the nodes of the current element.
c         The keys to the enrichment types are stored in the element vector TypeXe
          CALL TypeXelement(OUTDIR,LENOUTDIR,JELEM,NNODE,NelmX,
     *               ix,TypeXe)

c         Element stiffness matrix computation, stored in AMATRX
          CALL K_U12(E,Nu,AMATRX,NDOFEL,NNODE,dimens,MCRD,
     *          COORDS,PSS,NnodX,ix,TypeXe,Dist,XYC0,XYCPrev,
     *          gint,sg,Xe,Ye,flag,BatG,DBatG,JatG)

c         Routine that multiplies AMATRX times U to obtain the force vector F
```

```
c          at the end of the current increment
           CALL MULT_V(AMATRX,NDOFEL,NDOFEL,U,F,NDOFEL)

c          Compute the residual force vector
           DO I=1,NDOFEL
             RHS(I,1) = RHS(I,1) - F(I)
           END DO

c          Compute stresses at Gauss points for post-processing purposes
c          Store them as SVARS for output to the results file (.fil)
           CALL  SVARS_U12(JTYPE,JELEM,SVARS,NSVARS,U,NDOFEL,BatG,
     *                   DBatG,JatG,gint,mpg,xypg)
         END IF
        END IF
       END IF
       RETURN
       END
```

C Element stiffness matrix. Subroutine: K_U12

```
      SUBROUTINE K_U12(E,Nu,AMATRX,NDOFEL,NNODE,dimens,MCRD,
     *           COORDS,PSS,NnodX,ix,TypeXe,Dist,XYC0,XYCPrev,
     *           gint,sg,Xe,Ye,flag,BatG,DBatG,JatG)

      IMPLICIT NONE
      INTEGER NDOFEL,NNODE,dimens,MCRD,PSS,NnodX,gint,flag,pos
      INTEGER l,i,j,kk,TypeXe(NNODE),ix(NNODE)
      REAL*8 E,Nu,Dist(NnodX,3),sg(3,*)
      REAL*8 AMATRX(NDOFEL,NDOFEL),XYC0(2),XYCPrev(2)
      REAL*8 Xe(2*NNODE),Ye(2*NNODE),COORDS(MCRD,NNODE),xl(dimens,NNODE)
      REAL*8 xsj(gint),shp(3,4)
      REAL*8 dNF(NNODE,2,4),Fnode(NNODE,4),H,Hnode(NNODE)
      REAL*8 B(3,NDOFEL), DB(3,NDOFEL), BT(NDOFEL,3), D(3,3)
      REAL*8 BatG(3*gint,NDOFEL),DBatG(3*gint,NDOFEL),JatG(gint)
      LOGICAL NodeType1,NodeType2

c     NOTES:
c      Routine shapef2D is called to compute standard shape functions,
c      derivatives and jacobian at integration points. This routine outputs:
c         shp(3,*)  - Shape functions and derivatives at point
c                     shp(1,i) = dN_i/dx = dN_i/dx1
c                     shp(2,i) = dN_i/dy = dN_i/dx2
c                     shp(3,i) = N_i
c         xsj       - Jacobian determinant at point
c      Local coordinates of integration points are passed in sg(1,*), sg(2,*)
c      Integration weights are passed in sg(3,*)
c     &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
c     &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&

c     Initialize AMATRX and logical variables
      CALL initializeM(AMATRX,NDOFEL,NDOFEL)
      NodeType1=.false.
      NodeType2=.false.

c     Reduce info passed thru COORDS (3D) to xl (2D)
      DO i=1,dimens
        DO j=1,NNODE
          xl(i,j)=COORDS(i,j)
        END DO
      END DO

c     Define constitutive stress-strain elastic matrix
      CALL CALC_D(PSS,D,E,Nu)

c     Specify the type of nodal enrichment
      DO i=1,NNODE
        IF (TypeXe(i).eq.1) THEN
          NodeType1=.true.
        ELSEIF (TypeXe(i).eq.2) THEN
          NodeType2=.true.
        END IF
      END DO

c     Numerical integration loop over gint integration points
      DO l = 1,gint
c       Compute shape functions, derivatives and jacobian at integration point
        CALL shapef2D(sg(1,l),xl,shp,xsj(l),dimens,NNODE,ix,.false.)
        IF (flag.eq.1) THEN !Element is subdivided for integration
            xsj(l) = sg(3,l) !The integration weight includes the jacobian
        ELSE !Element is not subdivided. Standard integration
```

```
          xsj(l) = xsj(l)*sg(3,l)
        ENDIF

c       Value of the Heaviside function at integration point
c         (This call is also used to store the values of H
c          at nodes of the element for modified enrichment)
        IF (NodeType1) THEN
          CALL heaviside(NnodX,Dist,NNODE,ix,shp,H,Hnode)
        ENDIF

c       Derivatives of shape functions Ni times enrichment functions Fj at integration point
c         (This call is also used to compute the derivatives of shape functions Ni times
c          enrichment functions Fj at nodes of the element for modified enrichment)
        IF (NodeType2) THEN
          CALL fCrackTip(XYC0,XYCPrev,shp,Xe,Ye,dNF,Fnode)
        ENDIF

c       STIFFNESS MATRIX COMPUTATION:
c       Assembly of element matrix B (denoted as B) at integration point

        CALL initializeM(B,3,NDOFEL)
        Pos=1
c       Loop over nodes
        DO i= 1,NNODE
c         Contribution to B of derivatives of standard shape functions
          B(1,Pos)  = shp(1,i)
          B(2,Pos+1)= shp(2,i)
          B(3,Pos)  = shp(2,i)
          B(3,Pos+1)= shp(1,i)
c         Contribution to B of derivatives of shape functions times Heaviside function
          IF (TypeXe(i).eq.1) THEN
            B(1,2+Pos)  = shp(1,i)*(H-Hnode(i))
            B(2,3+Pos)  = shp(2,i)*(H-Hnode(i))
            B(3,2+Pos)  = shp(2,i)*(H-Hnode(i))
            B(3,3+Pos)  = shp(1,i)*(H-Hnode(i))
c         Contribution to B of derivatives of shape functions times crack tip functions
          ELSEIF(TypeXe(i).eq.2) THEN
            DO kk= 1,4
              B(1,2*kk+2+Pos)= dNF(i,1,kk)-shp(1,i)*Fnode(i,kk)
              B(2,2*kk+3+Pos)= dNF(i,2,kk)-shp(2,i)*Fnode(i,kk)
              B(3,2*kk+2+Pos)= dNF(i,2,kk)-shp(2,i)*Fnode(i,kk)
              B(3,2*kk+3+Pos)= dNF(i,1,kk)-shp(1,i)*Fnode(i,kk)
            END DO
          END IF
          Pos=Pos+12 !Each node has 12 dof
        END DO ! i = End loop over element nodes

        DB=matmul(D,B) ! Matrix D*B
        BT=transpose(B)  ! B transpose
c       Integration of BT*D*B
        AMATRX= AMATRX + matmul(BT,DB)*xsj(l)
c       Store information at each integration point for further post-processing
        DO  i=1,3
          DO j=1,NDOFEL
            BatG(3*(l-1)+i,j)=B(i,j)
            DBatG(3*(l-1)+i,j)=DB(i,j)
          END DO
        END DO
        JatG(l)=xsj(l)
      END DO ! l = End loop for each integration point
      RETURN
      END
```

D Output at integration points. Subroutine SVARS_U12

```
      SUBROUTINE SVARS_U12(JTYPE,JELEM,SVARS,NSVARS,U,Dof,BatG,DBatG,
     *                     JatG,gint,mpg,xypg)

c     Calculates and/or stores the following magnitudes at the element integration points,
c     storing them in SVARS: strains, stresses, strain energy density, dv/dx, du/dy, jacobian,
c     dNi/dx, dNi/dy, global coordinates of integration points.

      IMPLICIT NONE
      INTEGER i,j,k,NSVARS, Dof, gint, JTYPE,JELEM,mpg
      REAL*8  SVARS(NSVARS), U(Dof),BatG(3*gint,Dof),DBatG(3*gint,Dof)
      REAL*8  JatG(gint),B(3,Dof),DB(3,Dof),Bdvdx(3,Dof),Bdudy(3,Dof)
      REAL*8  EPS(3),SIG(3),W,dvdx(3),dudy(3),JAC,xypg(2,mpg)
c     &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
```

```
c     &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
c     First value stored in SVARS is the total number of integration points
c     of the enriched element
      SVARS(1)=gint
      DO i=1,gint
        JAC=JatG(i)
        DO k=1,3
          DO j=1,Dof
            B(k,j)=BatG(3*(i-1)+k,j)
            Bdvdx(k,j)=B(k,j) ! For computation of dv/dx
            Bdudy(k,j)=B(k,j) ! For computation of du/dy
            DB(k,j)=DBatG(3*(i-1)+k,j)
          END DO
        END DO
        CALL MULT_V(B,3,Dof,U,EPS,3) ! Compute strains EPS
        CALL MULT_V(DB,3,Dof,U,SIG,3) ! Compute stresses SIG
        W=0.5d0*(EPS(1)*SIG(1)+EPS(2)*SIG(2)+EPS(3)*SIG(3))

c       Computation of dv/dx & du/dy
c       Set to zero positions in the 3rd row of B associated with dN/dy
        DO j=1,Dof,2
          Bdvdx(3,j)=0.0d0
        END DO
        CALL MULT_V(Bdvdx,3,Dof,U,dvdx,3) !compute dv/dx, stored in dvdx(3)
c       Set to zero positions in the 3rd row of B associated with dN/dx
        DO j=2,Dof,2
          Bdudy(3,j)=0.0d0
        END DO
        CALL MULT_V(Bdudy,3,Dof,U,dudy,3) !compute du/dy, stored in dudy(3)

c       Store in SVARS the following information at integration points
        SVARS(1+20*(i-1)+1)=EPS(1)
        SVARS(1+20*(i-1)+2)=EPS(2)
        SVARS(1+20*(i-1)+3)=EPS(3)
        SVARS(1+20*(i-1)+4)=SIG(1)
        SVARS(1+20*(i-1)+5)=SIG(2)
        SVARS(1+20*(i-1)+6)=SIG(3)
        SVARS(1+20*(i-1)+7)=W
        SVARS(1+20*(i-1)+8)=dvdx(3)
        SVARS(1+20*(i-1)+9)=dudy(3)
        SVARS(1+20*(i-1)+10)=JAC   ! Jacobian includes integration weight
c       Store in SVARS the shape functions derivatives dNi/dx, dNi/dy for external computation
c       of dq/dx, dq/dy (used in domain interaction integrals).
c       (we take them from the positions associated with the standard dofs)
        SVARS(1+20*(i-1)+11)=B(1,1)
        SVARS(1+20*(i-1)+12)=B(1,13)
        SVARS(1+20*(i-1)+13)=B(1,25)
        SVARS(1+20*(i-1)+14)=B(1,37)
        SVARS(1+20*(i-1)+15)=B(2,2)
        SVARS(1+20*(i-1)+16)=B(2,14)
        SVARS(1+20*(i-1)+17)=B(2,26)
        SVARS(1+20*(i-1)+18)=B(2,38)
        Store in SVARS the global coordinates of integration points
        SVARS(1+20*(i-1)+19)=xypg(1,i)
        SVARS(1+20*(i-1)+20)=xypg(2,i)
      END DO !i loop over all integration points of the element
      RETURN
      END
```

References

[1] Moës N, Dolbow J, Belytschko T. A finite element method for crack growth without remeshing. *Int J Numer Methods Engng* 1999;46(1):131–150.

[2] Karihaloo BL, Xiao QZ. Modelling of stationary and growing cracks in FE framework without remeshing: a state-of-the-art review. *Comp & Struct*

2003;81:119–129.

[3] Abdelaziz Y, Hamouine A. A survey of the extended finite element. *Comp & Struct* 2008;86:1141–1151.

[4] Mohammadi S. Extended finite element method. Oxford: Blackwell Publishing, 2008.

[5] Hibbitt, Karlsson & Sorensen, Inc. ABAQUS/Standard User's Manual, v. 6.5, Pawtucket, Rhode Island, 2004.

[6] Giner E, Sukumar N, Denia FD, Fuenmayor FJ. Extended finite element method for fretting fatigue crack propagation. *Int J Solids Struct* 2008;45:5675-5687.

[7] Giner E, Vercher A, González OA, Tarancón JE, Fuenmayor FJ. Crack growth in fretting-fatigue problems using the extended finite element method. In: Mota-Soares CA *et al.*, editors. III European Conference on Computational Mechanics. Springer, Lisbon, 2006. p. 253.

[8] Giner E, Sukumar N, Fuenmayor FJ, Tarancón JE. Singularity enrichment for complete contact sliding problems using X-FEM. In: Chen JS, Liu WK, editors. VII World Congress on Computational Mechanics. Los Angeles, 2006.

[9] Stolarska M, Chopp DL, Moës N, Belytschko T. Modelling crack growth by level sets in the extended finite element method. *Int J Numer Methods Engng* 2001;51:943–960.

[10] Ventura G, Budyn E, Belytschko T. Vector level sets for description of propagating cracks in finite elements. *Int J Numer Methods Engng* 2003;58:1571–1592.

[11] Sukumar N, Prévost J-H. Modeling quasi-static crack growth with the extended finite element method. Part I: Computer implementation. *Int J Solids Struct* 2003;40(26):7513–7537.

[12] Li FZ, Shih CF, Needleman A. A comparison of methods for calculating energy release rates. *Engng Fracture Mech* 1985;21(2):405–421.

[13] Moran B, Shih CF. Crack tip and associated domain integrals from momentum and energy balance. *Engng Fracture Mech* 1987;27(6):615-642.

[14] Huang R, Sukumar N, Prévost J-H. Modeling quasi-static crack growth with the extended finite element method. Part II: Numerical applications. *Int J Solids Struct* 2003;40(26):7539–7552.

[15] Bordas S, Nguyen PV, Dunant C, Guidoum A, Dang HN. An extended finite element library. *Int J Numer Methods Engng* 2007;71:703–732.

[16] Wyart E, Duflot M, Coulon D, Martiny P, Pardoen T, Remacle J-F, Lani F. Substructuring FE-XFE approaches applied to three-dimensional crack propagation. *J Comput Appl Math* 2008;215:626–638.

[17] Shi J, Lua J, Waisman H, Phillip L, Belytschko T, Sukumar N, Liang Y. X-FEM toolkit for automated crack onset and growth prediction. In: 49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference. Schaumburg, IL, April 2008.

[18] Belytschko T, Black T. Elastic crack growth in finite elements with minimal remeshing. *Int J Numer Methods Engng* 1999;45(5):601–620.

[19] Béchet E, Minnebo H, Moës N, Burgardt B. Improved implementation and robustness study of the X-FEM for stress analysis around cracks. *Int J Numer Methods Engng* 2005;64(8):1033–1056.

[20] Laborde P, Pommier J, Renard Y, Salaün M. High-order extended finite element method for cracked domains. *Int J Numer Methods Engng* 2005;64:354–381.

[21] Zi G, Belytschko T. New crack-tip elements for XFEM and applications to cohesive cracks. *Int J Numer Methods Engng* 2003;57:2221–2240.

[22] Osher S, Sethian JA. Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations. *J Comput Phys* 1988;79(1):12-49.

[23] Duflot M. A study of the representation of cracks with level sets. *Int J Numer Methods Engng* 2007;70(11):1261–1302.

[24] Gravouil A, Moës N, Belytschko T. Non-planar 3D crack growth by the extended finite element and level sets - Part II: Level set update. *Int J Numer Methods Engng* 2002;53(11):2569–2586.

[25] Sukumar N, Chopp DL, Moran B. Extended finite element method and fast marching method for three-dimensional fatigue crack propagation. *Engng Fracture Mech* 2003;70(1):29–48.

[26] Sukumar N, Chopp DL, Béchet E, Moës N. Three-dimensional non-planar crack growth by a coupled extended finite element and fast marching method. *Int J Numer Methods Engng* 2008;76(5):727–748.

[27] Daux C, Moës N, Dolbow J, Sukumar N. Arbitrary branched and intersecting cracks with the extended finite element method. *Int J Numer Methods Engng* 2000;48(12):1741–1760.

[28] Sukumar N, Moës N, Moran B, Belytschko T. Extended finite element method for three-dimensional crack modelling. *Int J Numer Methods Engng* 2000;48(11):1549–1570.

[29] Chen FHK, Shield RT. Conservation laws in elasticity of the J-integral type. *J Appl Math Phys (ZAMP)* 1977;28:1–22.

[30] Yau JF, Wang SS, Corten HT. A mixed-mode crack analysis of isotropic solids using conservation laws of elasticity. *J Appl Mech* 1980;47:335–341.

[31] Erdogan F, Sih GC. On the crack extension in plates under plane loading and transverse shear. *J Basic Engng* 1963;85:519-527.

[32] Richard HA, Fulland M, Sander M. Theoretical crack path prediction. *Fatigue Fract Engng Mater Struct* 2005;28:3-12.

[33] Giner E, Sukumar N, Fuenmayor FJ, Vercher A. Singularity enrichment for complete sliding contact using the partition of unity finite element method. *Int J Numer Methods Engng*, published online, DOI:10.1002/nme.2359.

[34] Gdoutos EE. Fracture Mechanics: an Introduction. Solid Mechanics and its Applications. Dordrecht, Holland, Kluwer Academic Publishers, 1993.

[35] Giner E, Fuenmayor FJ, Baeza L, Tarancón JE. Error estimation for the finite element evaluation of G_{I} and G_{II} in mixed-mode linear elastic fracture mechanics. *Finite Elem in Analysis & Design* 2005;41:1079–1104.

[36] Giner E, Fuenmayor FJ, Tarancón JE. An improvement of the EDI method in linear elastic fracture mechanics by means of an *a posteriori* error estimator in G. *Int J Numer Methods Engng* 2004;59:533–558.