Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Numerical simulation of multiple 3D fracture propagation using arbitrary meshes

Adriana Paluszny [*], Robert W. Zimmerman

Department of Earth Science and Engineering, Imperial College, London, United Kingdom

ARTICLE INFO

Article history:
Received 26 May 2010
Received in revised form 27 October 2010
Accepted 21 November 2010
Available online 25 November 2010

Keywords:
Crack propagation
Three-dimensional
Brittle
Arbitrary mesh
FEM
J Integral

ABSTRACT

This paper describes a mesh-independent finite element based method for propagating fractures in three dimensions. The iterative algorithm automatically grows fractures in a 3D brittle medium represented by an isotropic linear elastic matrix. Growth is controlled by an input failure and propagation criterion. The geometry and mesh are stored separately, and mesh refinement is topologically guided. Propagation results in the modification of crack geometry, as opposed to changes in the mesh, as the arbitrary tetrahedral mesh adapts to the evolving geometry. Stress intensity factors are computed using the volumetric J Integral on a virtual piecewise cylinder. Modal stress intensity factors are computed using the decomposition method. Mesh and cylinder size effects are studied, as is computational efficiency. A through-going crack embedded in a thick slab, and a horizontal and inclined penny-shape crack, are used to validate the accuracy of the method. The predicted stress intensity factors are in good agreement with analytical solutions. For six integration points per tip segment, integration local to single tips, and a cylinder radius that adapts to the local geometric conditions, results agree with analytical solutions with less than 5% deviation from experimental results.

1. Introduction

Interest in simulating crack growth and pattern formation extends across a variety of scientific and engineering fields, including structural analysis, material design, nuclear waste disposal risk assessment, oil and gas reservoir engineering, and subsurface ore mining. Numerical simulations are instrumental in predicting the formation and behavior of these fracture systems, due to the geometric and physical complexity inherent in fracture phenomena.

Numerous recent studies have attempted to model fracture propagation in 3D. Two main approaches can be identified in the literature for fracture analysis: discrete and smeared, also known as geometric/non-geometric, or grid/subgrid methods [1]. Non-geometric methods represent cracks at a sub-grid level; thus, the cracks exist within the mesh but are not explicitly described by it (e.g. smeared crack, Jirasek [2]; mesh-free crack, Rabczuk and Belytschko [5], Bordas et al. [6]; cracking particles, Rabczuk and Belytschko [7]). Smeared approaches represent cracks as an iso/anisotropic damage concentration band within a mesh element from which fracture geometry can be inferred, instead of being explicitly defined. Mesh-free methods, including cracking particles, keep track of fractures implicitly by tagging nodes during crack advance.

In contrast, geometric based algorithms represent cracks discretely (e.g. boundary element based, Carter et al. [8], Franc-3D [9]; finite element based, Lin and Smith [10], Schöllmann et al. [11]; extended finite element method, Bordas et al. [3], Wyart et al. [4]). The finite element approach defines fracture geometry at the grid level, while the extended finite element represents cracks using discrete discontinuities in an intra-element basis, and relies on limited level set descriptions of crack paths. Geometric methods provide a discrete volumetric description of the cracks, allowing for flow or fragmentation simulations to be conducted on the ensuing geometries. Nick et al. [12] and Paluszny and Matthai [13] are examples of applications of discrete crack propagation to the study of flow properties of fractured rock masses. Some of the geometric approaches depend on a stress intensity factor library, which limits computations to a specific set of boundary conditions and initial crack geometries (e.g. [14]). Others impose limits on the size of the elements allowed to be defined along the crack tip [10,15]. But, most importantly, the great majority of geometric methods do not maintain a representation of the fractures separate from the mesh, and rely on fragile mesh editing techniques, such as in situ insertion of new crack nodes, edges and faces, to capture mesh growth (e.g. [11,16–18]). The discrete fracture growth method presented here captures fracture shapes accurately by differentiating fracture and matrix domains explicitly using solid modeling techniques. Using this approach, a geometric description of the crack can be extracted at any stage of the simulation. Thus, growth operations are performed directly on the geometry, as opposed to attempting to insert a crack configuration into an arbitrary mesh [17]. Here, fractures are manipulated independently of their discretization in space.

* Corresponding author. Tel.: +44 20 7594 7435; fax: +44 20 7594 7444.
E-mail address: apaluszn@imperial.ac.uk (A. Paluszny).

0045-7825/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2010.11.013

Numerical simulation of 3D discrete fracture propagation using the finite element method often requires special symmetric, brick-type mesh configurations around the crack tip for stress intensity factor computations using the J Integral [19,20]. Most methods, such as the J Integral [19,20], the virtual crack extension technique [21–23], the nodal force method [24], the body force method [25], and the displacement correlation technique [26] rely on a specific mesh configuration around the tip. The alternative boundary element method does not require meshing, but does not allow for heterogeneous material property definition. Other finite element based methods, such as the one presented by Okada et al. [27], do not require a brick mesh, but constrain node locations around the tip. Brick mesh generation around fracture tips becomes prohibitive for complex, evolving three-dimensional geometries. However, cutting edge methodologies developed for finite element based 3D crack propagation do not seem to address this issue, and continue to rely on the collocation of brick elements around crack tips (e.g. [11,17,28]), or avoid the computation of stress intensity factors around the tip altogether (e.g. computation of local dissipation around the crack front by Gürses and Miehe [18]).

Research in robust and automatic tetrahedral meshes over the past decades has made available tools that can generate and optimize volumetric tetrahedral and hybrid meshes for complex geometries with great efficiency (e.g. [29]). To take full advantage of this technology, and facilitate automatic 3D crack propagation, it becomes imperative to further develop methodologies for stress intensity factor computation in arbitrary meshes in order to take full advantage of high quality, mature hybrid mesh generation technologies (cf. [30–33]). In this work, we demonstrate a mesh-independent computation of stress intensity factors that allows for automatic crack growth simulation. We implement the equivalent domain integral method for the computation of J [34] originally formulated for brick-structured meshes, to compute stress intensity factors on arbitrary meshes. The method is mesh-independent, and combines 2D integration in a polar coordinate system local to each crack segment, with a Langrangian computation of the J Integral [35]. Integration is performed on a virtual piecewise cylinder [35]. Modal stress intensity factors, required for the determination of crack growth direction, are computed using the decomposition method [36,37]. Alternatively, integration can also be performed over a larger domain comprising all tetrahedral elements in the neighborhood of the tip [38]. However, these approaches require the sampling of a large region around the tip to attain accuracy. When simulating multiple crack propagation, these large integration domains may not be available, due to the proximity of neighboring cracks.

The contribution of this paper is twofold: we propose the decoupling of the geometric description of crack propagation from the meshing to enhance control over the tip front resolution and improve shape housekeeping. Additionally, the decoupling of geometry from meshing is advantageous, as it allows the framework to gain modularity, since meshing, geometry, and deformation techniques may evolve independently, and can thus be profited from as advances become viable. In particular, in the advent of isogeometric analysis [39], it becomes of particular interest to formulate mesh-independent approaches for crack propagation that rely on a discrete, smooth, parametric representation of the matrix and fractures embedded within it. The second contribution of the paper is the introduction of the reduced virtual integration technique, which exploits the virtual cylinder integration domain concept combined with an analytical virtual extension function definition, with a focus on small integration domains and localized computations. Both are essential components of a multi-fracture propagation algorithm, based on the FEM and on linear elastic fracture mechanics, to investigate fracture pattern formation based on a local failure criterion. The proposed methodology applies to quasi-static growth, and is the extension of an originally two-dimensional method [40]. The result is a numerical method that deals with the simultaneous growth of multiple arbitrarily-shaped fractures, while remaining flexible, accurate, and efficient. As suggested by Ingraffea and Wawrzynek [41] it allows for interchangeable modules for failure and propagation criteria that can be assembled together to form problem-specific formulations. Growth is determined by local energy concentration at the tip, and is proportional to the stress intensity factor around each tip node. A geometry kernel keeps track of fracture intersections and potential closure. The output is a three-dimensional evolving fracture pattern including discrete aperture distribution, full shape description, and stress state. Benchmarking verifies that the numerical methods and propagation algorithm perform adequately for various sub-problems of fracture set formation. Details and validation of the 2D version of this method can be found in Paluszny [42] and Paluszny and Matthai [40]. The underlying assumptions of the model are: (1) the matrix is linearly elastic, isotropic, and homogeneous; (2) fractures grow from an initial set of planar ellipses; (3) propagation is quasi-static and strain rate independent; and (4) there is no cohesion/traction between fracture walls.

The paper is organized as follows. Section 2 presents details of fracture geometry. Section 3 discusses the governing equations of deformation and propagation. Section 4 discusses relevant implementation details. Section 5 lays out the validation of the implemented methods. Finally, Section 6 summarizes the contributions of our work.

2. Fracture geometric representation

An important step in the extension of the existing 2D fracture propagation method is to formulate a 3D geometric representation of the cracks [8]. Discrete cracks are represented using solid modeling techniques that take into account the evolutionary nature of the geometry [43]: solids are defined by a Boundary Representation (BREP), surfaces are parametrically represented by 3D polygonal meshes or Non-Uniform Rational B-Splines (NURBS) [44] surfaces, and tips are defined using polylines or NURBS curves. An expanding set of ring-like tip fronts captures the geometric extension of the crack due to growth (see Fig. 1). These might be coplanar, as in (a), or non-coplanar, as in (b).

The domains of the simulation, matrix and fractures, have independent geometric BREPs. Fractures are stored as negative non-convex polyhedra, while the shape of the host body is stored as a positive non-convex polyhedron. Faceted polyhedra allow for systematic Boolean operations to be applied whenever a fracture shape interacts with a boundary or a neighboring crack wall.

Crack geometry evolves as a function of deformation and the mesh is regenerated to adapt to this change, as opposed to growth which follows the element boundaries of an underlying mesh structure. In the present simulations, bodies are represented by a combination of smooth and faceted surfaces and curves, and the mesh is only an instrument to obtain information about the deformation and to predict its effect on the geometry of the host matrix and its fractures.

Our strategy relies on the availability of automatic volume meshing technologies. At present, we rely on a robust, fast, command-driven octree volumetric mesher capable of generating high quality meshes from water-tight BREP geometries. Thus, the research effort is shifted from operating and editing a mesh, to geometric modeling of the ensuing crack shape.

Several procedures are key to the success of the decoupling of geometry from mesh: information flow from geometry to mesh, fracture surface and tip extraction, tip smoothing and discontinuity

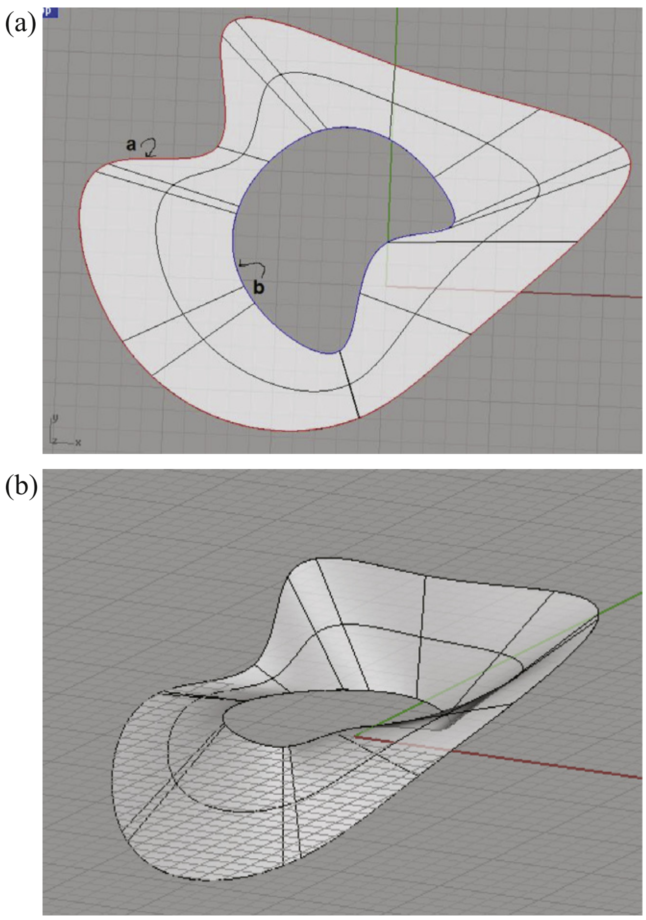

Fig. 1. 3D crack geometry extension. Ring-like expanding tip front of a 3D crack. The two curves are approximated by a NURBS curve. A ruled NURBS surface describes the region between them represents the new surface crack area resulting from a propagation step. In (a) these are coplanar and in (b) they are not.

tracking, identification of local tip coordinate system, geometric handling of propagation, and tip snapping.

2.1. Geometry to mesh

The geometry is a collection of entities, g, which is a set of points, curves, and surfaces. These are classified into families which are maintained after meshing, allowing rapid identification of crack geometry without the need to search or operate on the mesh. Each geometric entity either belongs to the representation of a particular fracture, or belongs to the matrix. The geometry of each fracture, f_i, is stored as a set of surfaces and curves, $g_{surface}$ and g_{curve}, respectively. The matrix, m, is also initially represented by a set of surfaces. It follows that

$$\forall g | (\exists i | g \in f_i) \vee (g \in m) \tag{1}$$

$$\forall g | g_{surface} \overset{meshing}{\rightarrow} \{t_i\}_{i \rightarrow triangles} \tag{2}$$

$$\forall g | g_{curve} \overset{meshing}{\rightarrow} \{b_i\}_{i \rightarrow bars} \tag{3}$$

After meshing, each surface becomes a set of triangles, and each tip curve a set of segments. There are no restrictions on the number of tips or surfaces that can belong to any of these shapes. However, their combination must form a water-tight BREP of the body to ensure meshing success. Water-tightness of the BREP is ensured by defining intersection curves between free surfaces using linear splines. Thus, a tip front can be defined by a combination of NURBS with varying degrees. This avoids leaking due to mismatch of higher order intersection splines.

2.2. Fracture surface and tip extraction

Tip curves describe growth, and the fracture body becomes the ensuing polyhedron that grows driven by the addition of ruled surfaces to its boundaries (see Fig. 2). Thus, once the mesh is generated it becomes necessary to identify which elements belong to each fracture, T_f, and to the matrix boundary, T_m. In order to keep track of this grouping, surfaces are tagged before meshing by assigning them to a layer or family which is retained during meshing; a feature available in most volumetric meshers. After extraction, each set of triangles is classified into a family: fracture, T_f, or matrix, T_m, and each tip curve a set of segments organized into tip sets, B_f and B_m. It follows that

$$\forall i | (\exists f_i | T_i \in f_i) \vee (T_i \in m) \tag{4}$$

$$\forall i | (\exists f_i | B_i \in f_i) \vee (B_i \in m) \tag{5}$$

where T_i is a triangle representing part of the tip surface, and B_i is a tip segment. Therefore, tips can belong to a fracture or to the matrix, as they might correspond to a fracture that has intersected the boundary and now shares its geometry.

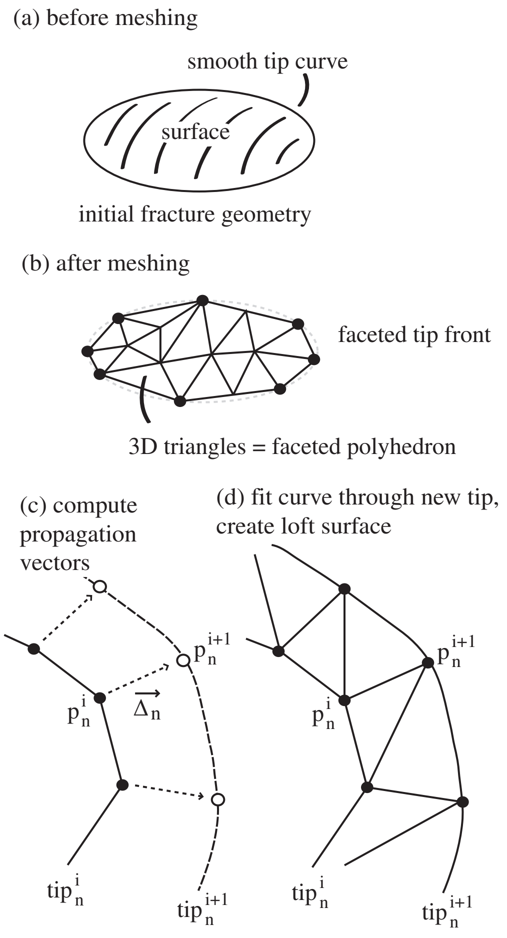
(a) before meshing
smooth tip curve
surface
initial fracture geometry
(b) after meshing
faceted tip front
3D triangles = faceted polyhedron
(c) compute propagation vectors
(d) fit curve through new tip, create loft surface
p_n^{i+1}
p_n^i
$\overrightarrow{\Delta_n}$
tip_n^i
tip_n^{i+1}
p_n^{i+1}
p_n^i
tip_n^i
tip_n^{i+1}

Fig. 2. Crack growth. (a) Fracture is a flat, planar surface cropped by a smooth curve (in this case elliptic), (b) after meshing, the fracture surface becomes a polyhedron, fracture tip becomes a polyline, (c) after the computation of physical parameters, an external curve defines the new tip position, and (d) a smooth curve is fitted through the new tip. A lofted surface approximates the new fracture surface. The fracture is, for illustration purposes, of arbitrary shape.

Tips can be similarly stored in families. Alternatively, these can be extracted from the fracture surface set, by extracting the borders of the ensuing surface sets. In this case, each pair of neighboring triangles is tested for overlap of their shapes – on a single triangle plane –, and if the measured angle at the edge denotes sharpness, the edge of the polyhedron is extracted and becomes a tip candidate. Furthermore, other tip extraction methods, such as the ridge extraction method [45], can also be implemented.

Systematic tracking of tip geometry is an instrument to optimize the crack growth algorithm, as it allows for an identification of tip nodes, and their volumetric element neighbors, which does not scale with tip front size.

2.3. Tip smoothing and discontinuity tracking

Tips are tracked individually during growth, thus, propagation does not depend on the initial resolution of the tip. Instead, it is derived from its equivalent smooth geometric description at each growth step. This is achieved by using a combined discrete fracture representation (as a polyline) with a continuous description (NURBS curve). When tips are discretized, they are stored as sets, which have no specific order. In order to create a smooth curve representation of the tip, it is necessary to identify the order of the segments that compose the original polyline, lost during meshing. However, during growth, regions of the crack tip might become deactivated due to intersection with an external boundary. These segments, $B_i \notin tip$ are also extracted from the mesh. Thus, the fracture curve boundary may become a set of discontinuous segment sets, which are selectively smoothed during growth. The underlying mesh connectivity is exploited to organize the segment set into a set of discontinuous polylines, Q_{fk}, tip and non-tip polylines k corresponding to a fracture f. Thus, $\forall i \,\exists\, k | B_i \in Q_{fk} \wedge (Q_{fk} \in tip \vee Q_{fk} \notin tip)$. The resulting, $Q_{fk} \in tip$ can then be independently approximated using smooth curves. Thus,

$$Q_f \rightarrow \langle Q_{f1}, Q_{f2}, \ldots Q_{fk} \rangle \tag{6}$$

where Q_f denotes the closed curve boundary of the fracture surface, composed by smooth, active tips and faceted, dormant tip segments. Subsequently,

$$\forall Q_{fk} \in tip \Big| \left(Q_{fk} \rightarrow Q_{fk}^{spline} \right) \tag{7}$$

thus, each tip polyline can be approximated using a NURBS curve, B_{fk}^{spline}, used to extract a smooth front after meshing. Once Q_f is identified the corresponding NURBS surface, fitted through the fracture walls, is trimmed. In order to preserve fracture shape integrity, this must be done separately for each fracture wall. A set of points can be approximated and trimmed by a NURBS curves or surface using standard B-Spline approximation techniques [46]. This resolution independent approach avoids the introduction of artificial kinks into the crack representation, and allows for resolution to be defined during meshing, as shown in Fig. 2.

2.4. Identification of the local tip coordinate system

Once B_{fk}^{spline} is generated, its Frenet frame (cf. [46]) can be exploited to generate an automatic fast description of the local coordinate system along the crack front (see Fig. 3). For a NURBS curve c_i, the local coordinate system can be extracted locally as a function of its local derivative

$$x_3 = \frac{-c_i'(p_n)}{|c_i'(p_n)|} \tag{8}$$

$$x_2 = \frac{-c_i'(p_n) \times c_i''(p_n)}{|c_i'(p_n) \times c_i''(p_n)|} \tag{9}$$

$$x_1 = x_2 \times x_3 \tag{10}$$

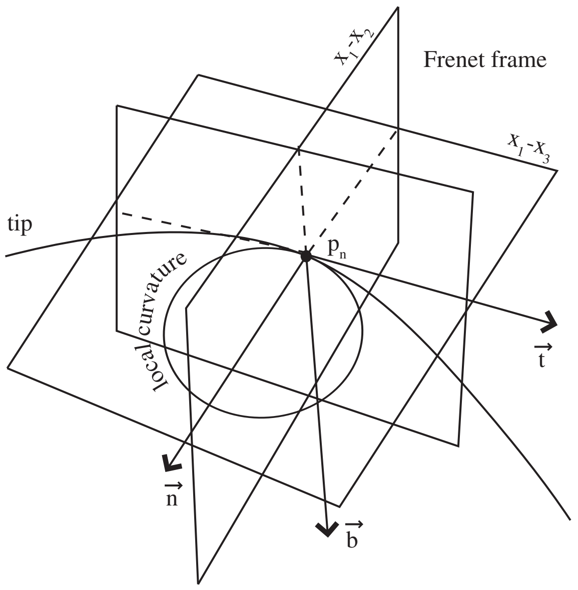

Fig. 3. Frenet frame. Describes the local coordinate system of a parametric curve as a function of its local derivatives. The embedded circle denotes the local curvature radius of the curve. Here $-\vec{n} = x_1$, $-\vec{b} = x_2$, and $-\vec{t} = x_3$.

where $c_i'(p_n)$ and $c_i''(p_n)$ are the first and second derivative of the NURBS curve at p_n. The local coordinate system is essential to the integration of stress intensity factors ahead of the tips, as described in Section 3.

2.5. Geometric handling of propagation

At each propagation step, a new crack tip curve is computed. This is achieved by approximating a set of new crack tip points and fitting a NURBS curve along them. Thus, growth is reduced to computing propagation vectors, $\vec{\Delta}_n$, for all locations along the tip. The new crack tip surface represents the region between the previous and new crack tip curves, captured by a lofted surface (cf. [47]) between the old and new crack tips. This ruled surface, which corresponds to the new crack surface area, is defined as an internal, split wall of the geometry. Thus, the fracture body is extended by merging its body to a surface describing the extension of the crack. This surface is a collapsed polyhedron that acts as if it would cut the material ahead of the fracture. However, there is no actual mesh cutting within this process. The new geometry initially lacks volume, but can open as a response to deformation (see Fig. 1).

There are three main processes related to geometrically quantifying growth: propagation, which deals with the extension of the shape; intersection, which handles geometric interaction with other shapes; and closure, which handles possible mesh tangling due to localized and extreme reduction of aperture.

2.5.1. Propagation

The propagation criterion estimates the extension of the crack and its direction. Given $\vec{\Delta}_n$, we compute the new location of each tip node, interpolate a curve through them, and extend the crack front by lofting a surface between the NURBS curves defining the old and new crack fronts. The propagation computation is performed by tip segment, thus, for each tip node p_n, $\vec{\Delta}_n$ is the average propagation vector between the two segments that the tip node belongs to, and

$$p_n^{j+1} = p_n + \vec{\Delta}_n \tag{11}$$

where p_n^{j+1} is the new tip node location. Fracture shape, including aperture distribution, crack surface and tip description, are emergent properties of the simulation. Fig. 4 shows an example of a single crack propagating until it reaches the boundary. Fig. 5b and c are examples of multi-fracture growth: in both cases, fractures grow over several iterations marked with geometrical contour lines.

2.5.2. Intersection

Once a tip reaches a free surface it arrests, inducing the fracture shape to merge with the foreign object. There are two distinguishable cases: the tip arrests on a block boundary or on a fracture wall. In the first case, the fracture shape becomes part of the block description. In the second, fracture shapes merge into one object.

Intersections between growing fractures are handled using 3D Boolean operations. Specifically, fracture shapes are merged at intersection. We track intersection with other surfaces by checking if a crack advance vector intersects with any other surface within the geometry. When a fracture shape extends beyond a neighboring crack wall, the algorithm corrects the new lofted surface by adjusting the location of the tip advance. It follows that the intersection point between the advance vector and the external triangle, becomes the new final location of the tip node

$$p_n^{j+1} = I(\langle p_n, p_n^{j+1}\rangle, T_i) \tag{12}$$

where $\langle p_n, p_n^{j+1}\rangle$ is the advance vector, T_i is the external intersected triangle, and I is an operation denoting ray to triangle intersection. Fracture shape is subsequently merged with the proximal crack as it intersects it, see Fig. 6, by tagging the new lofted surface as a split/internal wall. Thus, the mesher identifies contact between bodies and merges them during pre-processing. Fig. 5a shows an example of a fracture arresting at intersection with a neighboring crack. It follows that for surfaces, g_s, and curves, g_c,

$$\forall f' | \exists g | ((g_s \in f' \wedge g_s \in m) \wedge (g_{c=tip} \in f' \wedge g_{c=tip} \in m)) \tag{13}$$

where f' is a fracture that has intersected the matrix, and their geometry has merged. In the case when two fractures intersect, then

$$\exists g | (g_s \in f_i \wedge g_s \in f_{i+1}) \wedge (g_{c=tip} \in f_i \wedge g_{c=tip} \in f_{i+1}) \tag{14}$$

The advance correction presented here models how an opening mode crack, at intersection with an open surface, ceases to grow and connects both paths enclosing a single region. Further post-growth geometric housekeeping includes fitting a NURBS curve through new node locations and stitching the new crack front to create the new lofted surface extension (see Section 2.3).

2.5.3. Closure

After each growth step, all fracture surfaces and curves are updated using the new deformed FEM mesh. Geometry nodes are moved to capture movement due to deformation. This tracks wall movements and monitors fracture aperture. At some locations, aperture reduces to the point of local fracture closure. Fracture closure also occurs when the fracture BREP is self-intersecting. This happens when the walls over-displace toward each other due to compressive forces. In this case, the tangled mesh is healed by automatically reconstructing the closed fracture walls from an interpolated average skeleton. The fracture skeleton is tracked

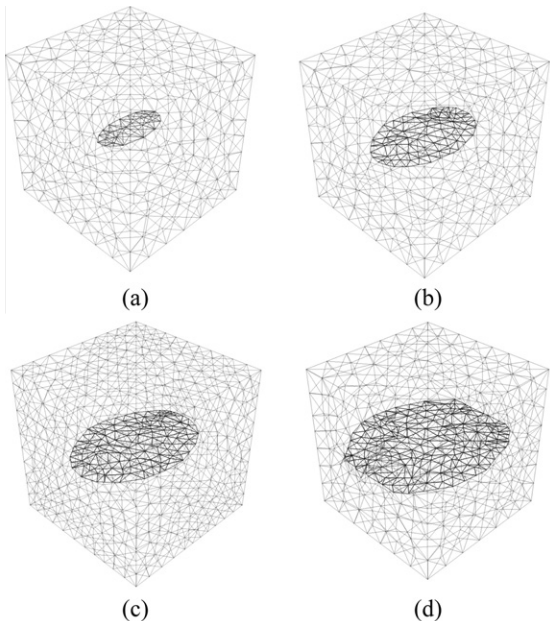

Fig. 4. Fracture propagation. Detail of the fracture mesh at four stages of propagation. In the last step the fracture reaches the boundary.

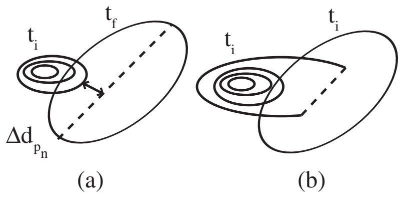

Fig. 6. Fracture-fracture intersection. (a) Cylinder size should have a maximum radius of $\Delta d_{p_n}/2$, so that the integration domain does not include the effects of the neighboring singularity and (b) once fractures intersect, their shapes are meshed and are subsequently handled as a single fracture with multiple tips.

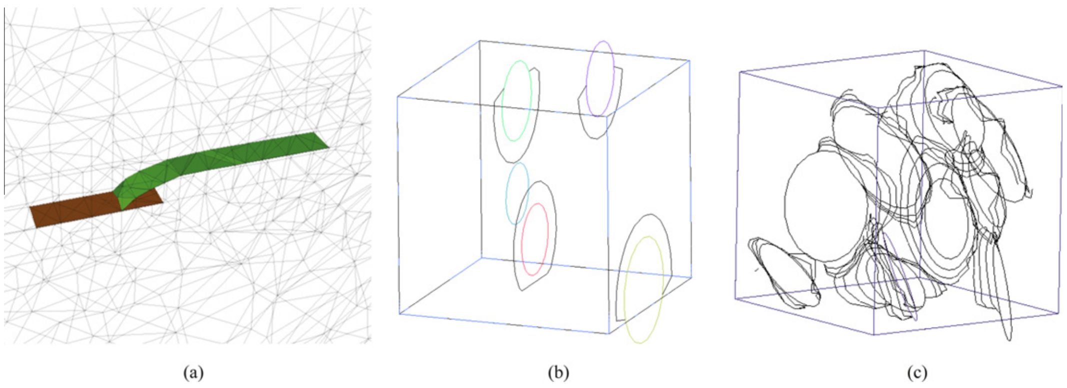

Fig. 5. Multi-fracture growth examples. (a) Two fractures intersect, (b) growth contours of four fractures mark new fracture tip position, fifth fracture does not grow, and (c) growth contours of ten fractures, each iteration is marked by a single contour, complex shapes are due to local stress field complexity.

and updated during growth, and represents an alternative framework to reconstruct partially closed cracks. Thus, when a fracture surface self intersects; using this algorithm, the fracture is partially closed at the location of the self-intersection. The fracture is not permitted to heal in this model, thus, those partially closed areas are untangled, and the simulation continues. Untangling can also be overcome by defining stiff elements inside the fracture that prevent mesh tangling. However, this approach requires an internal mesh. Another alternative, supporting crack healing, involves the splitting of the fracture BREP into several independent sub-regions allowing for some of them to remain open while others close.

2.6. Tip snapping

Once a tip becomes proximal to a foreign surface the volumetric mesh which discretizes the space between them might not be refined enough, conducing to large discretization errors (as described in Section 4.4). In this case, a tip can be induced to snap onto the external surface during propagation. It follows that the intersection between the extension of the advance vector to the proximal surface defines the new tip location

$$p_n^{j+1} = I(\langle p_n^{j+1}, p_n^{ext}\rangle, T_p) \tag{15}$$

where $\left\langle p_n^{j+1}, p_n^{ext} \right\rangle$ is the extended advance vector and T_p is the external proximal triangle. The length of the extended advance vector can be fixed or can by dynamically adjusted to the local discretization of the grid. Thus, for a low quality region, where a proximal free surface can be located, and in the event of tip propagation, the tip is extended to snap onto the external surface.

2.7. Geometric guided meshing setup

Once the geometry is defined we generate a mesh using an octree-based volumetric tetra mesher. Geometric information is used to set parameters for mesh generation in the following ways: (1) new tip surfaces are defined as internal/split walls, to ensure local discontinuity of the volumetric mesh, (2) geometric entities of a single fracture share a family tag, so that ensuing elements can be recognized after meshing, (3) curvature based refinement is controlled by the number of segments that would fit along a circle of the local radius of curvature, set to a minimum of ten to capture local curvature variations, (4) proximity based refinement is controlled by the minimum number of elements created at a gap between two geometric entities. Meshing parameters are not defined in an absolute manner, but are relative to the local conditions of the geometry and memory available for computations.

3. Deformation and propagation: governing equations

Deformation is modeled by the equations for homogeneous, isotropic, linear elastic media (cf. [48]). Thus,

$$\sigma = D(\varepsilon - \varepsilon_0) + \sigma_0 \tag{16}$$

where σ and ε are the Cauchy stress and infinitesimal strain vectors respectively (cf. [49]), σ_0 and ε_0 refer to the initial stress and strain, and D is a linear elastic stiffness matrix. For a static system, $\partial_o\sigma + F = 0$, where ∂_o is the differential matrix operator and F represents body forces exerted on the object such as gravity, dilatation, and acceleration. The elastic deformation equation is solved using the finite element method. Displacements are solved at element nodes, whereas material properties are defined, and stresses and strains are derived, at the element's Gaussian integration points. The 3D domain is discretized using a mesh composed by lines, triangles, and tetrahedra, i.e. isoparametric quadratic bars, triangles and tetrahedra. In particular, elements around the crack tip are quarter point elements, which capture the singularity ensuing at the fracture tip (see Fig. 7). These have been shown to improve accuracy in stress intensity factor calculations by up to 6% [50].

The inside of the fractures is not meshed. Thus, fractures exist as negative volumes surrounded by matrix elements. The fracture tip is meshed using bars, and its surface using triangles. By grouping elements into families, one can reconstruct, after meshing, which elements belong to which cracks, with time complexity O(1). Once the crack tip and crack surface is extracted, it is stored in a set with no particular order. As the simulation advances, and crack geometry changes, the mesh automatically adapts to cope with geometric change. This is achieved with full, as opposed to partial, remeshing.

Most numerical integration techniques for J, specific to finite element applications, require a symmetric discretization ahead of the tip, i.e., using brick elements such as hexahedra and prisms. This restriction is not commonly available in 3D meshing engines [51]. In the present case, the mesh is arbitrary and optimizes element interpolation quality. Moreover, generated discretizations can be hybrid (mix of tetrahedra, hexahedra, prisms and pyramids), thus, the mesh structure around the tip is not fixed but adapted to local geometry.

3.1. Stress intensity factor

The stress intensity factor values, K, are approximated at the center of each fracture tip segment. The stress intensity factor quantifies the energy concentrated around the crack tip. For linear elastic, brittle materials, $J = G$, where G is the strain energy release rate, and the J Integral, introduced by Cherepanov [19] and Rice [20], captures the amplitude of the singular field close to a sharp crack tip. The J Integral, formulated as a volume integral in local space (see Fig. 7a), is the amount of energy released by a unit crack extension in the direction of the crack plane, defined as

$$J_v = \frac{1}{A_c}\int_V \left(\sigma_{ij}\frac{\partial u_j}{\partial x_k} - W\delta_{ik}\right)\frac{\partial q_k}{\partial x_i}dv \tag{17}$$

where W denotes the strain energy density, A_c is the increased crack surface area, δ is the Kronecker delta, and q is an arbitrary weighting vector function representing the virtual crack extension. It follows that $J = G = K^2E_{eff}$, and [34]

$$E_{eff} = E\left[\frac{1}{1+\nu^2} + \left(\frac{\nu}{\nu+1}\right)\frac{\varepsilon_z}{\varepsilon_x+\varepsilon_y}\right] \tag{18}$$

where E is the elastic modulus, ν is Poisson's ratio, and ε_x, ε_y, and ε_z are the local principal strains.

In the reduced virtual integration technique J is integrated numerically over the volume of a virtual unit cylinder using Gaussian quadrature points defined in a local polar coordinate system. These are assumed to lie on a circle perpendicular to the crack front, on the $x_1 - x_2$ plane (see Fig. 8). The integration is performed

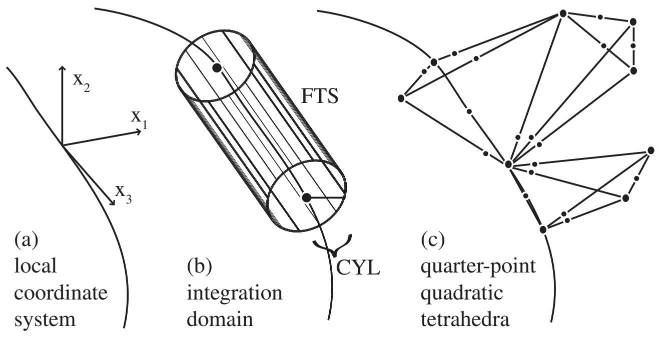

Fig. 7. Tip locality. (a) Local coordinate system, (b) Integration domain defined by the virtual cylinder, and (c) Singular quarter-point tetrahedra at the tip.

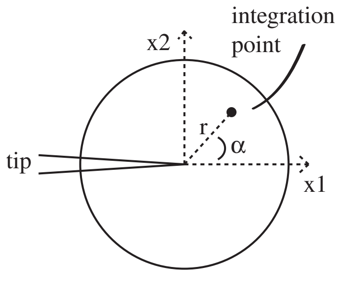

Fig. 8. Integration points of the circle, ip^{2D}, are defined in a local, polar coordinate system.

first over the area of a virtual circle perpendicular to the crack front and centered on the crack tip (see Fig. 7). As a second step, the computed J is integrated over the length of the crack tip. Details of the extension of this method to include body forces, crack surface tractions, temperatures, and pore pressure have been described by Cervenka [52] and Cervenka and Saouma [35].

Using this scheme, each integration point is defined by a radius, r, and an angle relative to the crack tip front on the $x_1 - x_2$ plane, θ. During the computation, the polar coordinate of the integration point $p_{rst}^{(ip)}$ is mapped to global coordinates $p_{xyz}^{(ip)}$, the element that contains $p_{xyz}^{(ip)}$ is found, the stresses are interpolated, the displacement gradients and strains are computed, and finally the contribution to the J Integral is computed. The integration point variables are mapped to the nodes using a distance-weighted averaging approach; alternatively, the super convergence patch recovery technique can be applied to improve accuracy [53].

The first step is to compute the location of the integration point of the virtual cylinder:

$$p_{rst}^{(ip)} = (r\cos\theta, r\sin\theta, 0) \tag{19}$$

$$p_{xyz}^{(ip)} = p_{rst}^{(ip)} R_{LtG} + t_c \tag{20}$$

where $p_{rst}^{(ip)}$ is the position of the integration point relative to the virtual circle, $p_{xyz}^{(ip)}$ is the position of the integration point in the global coordinates system, t_c is the tip center, and R_{LtG} is the rotation matrix from the local circle coordinate system to the global coordinate system, defined as

$$R_{LtG} = \begin{pmatrix} x_1[x] & x_1[y] & x_1[z] \\ x_2[x] & x_2[y] & x_2[z] \\ x_3[x] & x_3[y] & x_3[z] \end{pmatrix}^{-T} \tag{21}$$

where x_1, x_2, and x_3 are the normalized principal vectors of the crack's local coordinate system.

By interpolating the previously-extrapolated stress node values to $p_{xyz}^{(ip)}, \partial u(p_{xyz}^{(ip)})$ at each of the integration points can be approximated. The quantities $\sigma(p_{xyz}^{(ip)}) = D\varepsilon$ and $\varepsilon = 1/2(\partial\ u + \partial u^T)$ are computed locally, instead of by interpolation, to reduce the impact of the discretization error.

Integration is performed in the local crack coordinate system, and then converted back to the global coordinate system using the Jacobian mapping:

$$J'_{circle} = \sum_{ip=1}^{N_{ip2D}} \left(\sigma_{ij} \frac{\partial u_j}{\partial x_1} \frac{\partial q_1}{\partial x_j} - W \frac{\partial q_j}{\partial x_j} \right) |J_{det}| w \tag{22}$$

where all variables are evaluated at the integration point, $N_{ip^{2D}}$ represents the total 2D integration points, $W = 0.5\sigma_{ik}\varepsilon_{ik}$, x_1 is the direction of the propagation front, and all variables are assumed to be in the local crack coordinate system. J_{det} and w are the Jacobian determinant and weight of the transformation, where $J_{det} = (\partial x/\partial r) = r$, and r represents the crack local coordinate system. In the case of one integration point, $\alpha = 0$, $r = 2/3$ and $w^{(ip)} = (3/2)\pi$. Integration points and weights – from 1 to 40 integration points – for the unit circle, expressed in polar coordinates, can be found in [52].

The virtual extension function q at the location $\sigma\left(p_{xyz}^{(ip)}\right)$, is smooth function that has a value of l_k on the tip, and zero at the borders of the integration domain, where l_k is the magnitude of the virtual crack advance. The vector q can also be interpreted as a vector that represents the virtual crack extension direction. For simplicity, most authors assume that its direction is x_1. For $l_k = 1$, we define q as a linearly varying function along the axis of a unit cylinder, $t[-1:1]$, and along the radius, $r[-1:1]$. Accurate computation of the q gradient has been shown to have a significant effect on the quality of the solution [38]. The gradient of q is normally computed using finite element-based interpolation (e.g. [35]). Rajaram [54] showed that this has a significant detrimental impact on the quality of the solution, and can be bypassed by computing the gradient using an analytical expression. The next step is to integrate the approximation of J'_{circle} along the crack tip,

$$J'_v = \int_{tip} J'_{circle} ds \approx \sum_{ip=1}^{N_{ip1D}} J'_{circle}(ip^{1D}) \left| J_{det}^{(ip^{1D})} \right| w^{(ip^{1D})} \tag{23}$$

where $J'_v = A_c * J_v$, $N_{ip^{1D}}$ represents the total 1D integration points – three in the present case, and $J_{det}^{(ip^{1D})}$ and $w^{(ip^{1D})}$ are the Jacobian determinant and weight of the transformation, respectively.

This 1D integration is performed using three Gaussian integration points, and yields the virtual cylinder's total integral; see Fig. 9. The J integral is accumulated at each tip segment locality, as opposed to other approaches which involve more than one tip segment (e.g. [35]). However, as is later discussed, oscillations may arise when choosing small integration domains. We show that these can be reduced by extending the domain over several elements. The main advantage of integrating over a single tip segment is that it is completely localized. Thus, tips can be handled independently, without requiring the extraction of tip order from the mesh. This is a significant simplification, as tips might become arbitrarily discontinuous during growth, due to intersection with other fractures and fragmentation of the matrix body, and their

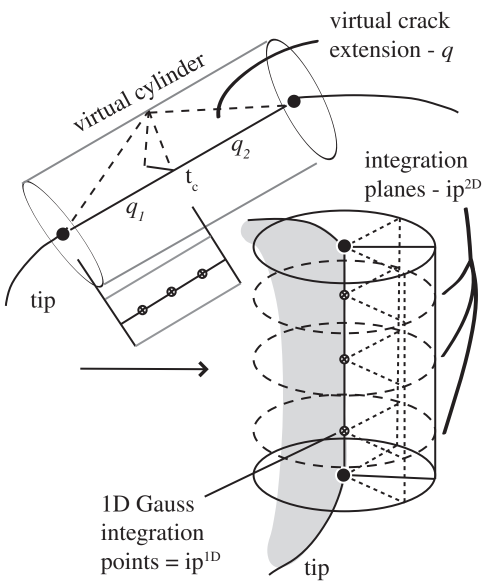
virtual crack extension - q
virtual cylinder
q2
tc
q1
integration planes - ip2D
tip
1D Gauss integration points = ip1D
tip

Fig. 9. Cylinder and integration point detail. q is defined in a piecewise manner over a single segment. J is accumulated at each of the planes within the virtual cylinder that corresponds to the integration point of the linear bar, ip^{1D}. Within each plane, J is integrated using 2D integration points in polar coordinates, $ip^{2D} = p_{rst}^{(ip)}$.

tracking might become strenuous. Furthermore, our approach aims to obtain good approximations of the stress intensity factor, with as few integration points as possible. For the computations presented in this paper, integration is performed over three circles for q_1 and q_2, using one integration point per circle. In this case, a total of three integration points is accumulated per tip segment, reducing computational time significantly.

The cylinder radius, r_{cyl}, defined as a dimensionless value – function of the local tip segment size –, is constrained by the location of other fractures and free surfaces, as the integration domain is required not to include any external singularities. Thus, the cylinder size must be smaller than half the distance to any neighboring surface, Δd_{p_n} (see Fig. 6). Rajaram et al. [38] showed that the quality of the approximation is a function of the size of the integration domain with respect to the dimension of the crack. They restrict their computations to domains that are up to half the size of the crack length. Cervenka and Saouma [35] present results where the domain integral radii varies from 2 to 10 times the size of the tip element and employ 15–40 integration points per segment. However, for multiple crack propagation, large integration domains become problematic as crack density increases. Additionally, larger integration domains also imply significant penalties in efficiency.

We propose the radius of the cylinder to be a dynamic parameter that can be adjusted to the conditions of the tip locality. Using numerical results we define an empirical function to guide the definition of r_{cyl} as a function of radius of curvature. As a first step we define bounds for r_{cyl}. Rigby and Aliabadi [37] propose

$$r_{cyl} \leqslant 0.5\rho \tag{24}$$

where ρ is the local radius of curvature. Rajaram et al. [38] further propose two bounds

$$\bar{t}_s/\rho^* \leqslant 0.25 \tag{25}$$

$$max_e/\bar{t}_s \leqslant 4 \tag{26}$$

where $\bar{t}_s$ is the average tip length along the front, max_e is the maximum length of the sampled element in the direction of x_1, and ρ^* is the relevant macroscopic crack dimension, which approximates the total length of the crack curve. Thus, Eq. (25) measures the tip refinement, while Eq. (26) measures the change in the element size normal to the local crack front. The above are formulated as generic bounds that are applied globally, thus, refinement will be controlled by the smallest, sharpest features, and will consequently penalize efficiency. We propose a fully local approach, which examines the vicinity of the tip before and after meshing to determine the optimal cylinder size. Before meshing, we assign for each fracture a minimum tip segment size, t_s of

$$2\pi\rho/n_{ref} \geqslant t_s \tag{27}$$

where n_{ref} determines the local refinement of the curve, ideally, $n_{ref} \ll \rho/t_s$. Large contours may be influenced by parts of the crack other than the point of interest [55]. In a locality, ρast can be approximated by $\pi\rho$, thus, Eq. (27) is a local expression of Eq. (25) that can be used as an *a priori* quality criterion. Once the length is fixed we can constrain r_{cyl} with Eq. (24). Finally,

$$r_{cyl} \approx \min(0.5\rho, t_s, \Delta d_{p_n}/2) \tag{28}$$

This definition of r_{cyl} adds two further constraints to the original bound proposed in Eq. (24): (1) r_{cyl} should not include external singularities – based on geometry, and (2) r_{cyl} should be proportional to the local tip segment length to better estimate modal K and avoid effects of neighbor geometry on the integral Rigby and Aliabadi [37]. Numerical results presented in Section 5 were specifically performed for

$$r_{cyl} = min(0.25\rho, 0.75t_s, \Delta d_{p_n}/2) \tag{29}$$

showing that we obtain accurate approximations well within the proposed range.

The accurate computation of J depends on several factors: (1) accuracy of local stress, strain, and displacement gradient; these are usually computed at the integration points, and then are mapped to the nodes. Subsequently, the finite element shape functions can be used to approximate these at an arbitrary location within the element. Other factors include (2) smooth definition of gradient of q within the integration domain, (3) correct mappings between local and global space, and (4) size of the integration domain.

Restrictions to crack configuration, specifically the minimum distance between crack tip and other free surfaces [38], are dropped by varying the virtual cylinder size. In contrast to previous authors, the geometry of our crack tips is stored separately, in the form of a NURBS curve. Thus, the local coordinate system required for the computations can be derived at low cost from the parametric curve, and does not require storage. This is important, because as fractures continue to grow, the tip curve increases significantly in size and required computations per tip segment become staggering if they are not optimized.

Original versions of stress intensity factor computations on arbitrary meshes perform integrations over a group of neighbor tips instead of at a single tip location, compute q derivatives numerically instead of analytically, interpolate values of σ and ε instead of locally computing them, and only present accurate results for large integration domains and large amount of circle integration points. We present a set of rules to select the size of the integration domain as a function of the domain.

3.2. Failure and propagation criteria

Once the stress intensity factor is computed along the crack tip, it can be used to estimate the magnitude and direction of crack growth. Modeling crack growth relies on three locally determined criteria: a failure criterion, and criteria for propagation magnitude and direction. Failure criteria determine whether or not a sample will fail (e.g. sub-critical crack growth, Broek [56]). Propagation criteria predict the extent of growth (e.g. Paris-type propagation, Charles [57]) by relating the energy accumulated around the locality of a tip with the amount of new created fracture surface area. Propagation direction criteria, often defined as a variant of the previous, predict the direction of crack growth (e.g. maximum circumferential stress, Erdogan and Sih [58]). These three components provide the fracture growth simulator with the information it requires to extend cracks.

In this study, a specific choice of these criteria is not advocated, as these have been shown to exhibit material and in situ conditions dependency (e.g. [59]). Frequently, these are empirical relationships derived from experimental studies. Instead, they are considered here to be part of the input of the model. The result of evaluating these criteria is a propagation vector $\vec{\Delta}_n$ of direction θ_n^j in the $x_1 - x_2$ plane, and of magnitude Δa_n^j. Thus, after each iteration, for each node along the tip front, a new location, $p_n^{j+1} = p_n^j + \vec{\Delta}_n$, can be predicted.

4. Implementation and efficiency

The 3D fracture propagation methods described here were developed as an extension to the mechanics module of the Complex System Modeling Platform (CSMP++), an object-oriented finite-element based API specialized in the simulation of complex geological processes [63]. The code for the FEM deformation and for the fracture propagation is developed using the C++ programming language. Meshing is outsourced, and performed by an octree volumetric mesher. The system of equations resulting from the finite element method accumulation is solved using the Fraunhofer SAMG Solver [64]. The current prototype is full operational on a 32bit PC as well as on a 64bit/ 64 GB RAM machine. All efficiency measurements are performed on an Intel Xeon X5460, with two quad cores running at 3.16 GHz.

The implementation of 3D crack propagation relies on the following building blocks: a 3D geometric kernel that handles geometric fracture housekeeping; a fracture mechanics kernel to evaluate stress intensity factors, failure and propagation criteria; a finite element kernel to conduct deformation; and a non-interactive volumetric mesher to generate discrete geometric descriptions for the latter. Implemented in an object-oriented language, the four different modules are independent and interchangeable.

4.1. Mesh size

Several concepts are used to characterize measurements of the mesh: fracture tip size, $\bar{t}_s$, is the average size of a tip segment and is an indicator of the refinement of the mesh at the tip; the global mesh refinement or global maximum element size, max_g, is an indicator of overall mesh resolution; and the maximum length of an element in front of a tip in the direction of the local x_1 (see Fig. 7a), max_e, can be used as an indicator of the gradient of the mesh in the integration domain. Ideally max_g is unconstrained, t_s is locally defined, and max_e can be used to estimate the ideal size of the integration cylinder, as discussed previously. t_s is constrained by geometry in the upper bound, and by efficiency in the lower bound. It is desired to generate meshes that are sufficiently resolved to honor topology and capture shapes, using the smallest possible number of elements. Thus, mesh resolution variation becomes a desirable feature in the mesher. It is the case for most volumetric mesh generators, where families can be defined, that a specific mesh size can be assigned per family and even per geometric entity. Thus, the size of the elements of an entire fracture tip or a specific segment can be constrained. Thus, elements at the tip front can be smaller in those places where external surfaces are proximal, and larger elsewhere.

4.2. Efficiency

Table 1 summarizes the meshing and running times of the presented technique, as compared to other approaches. Building the objects, including the generation of the three-dimensional mesh, converting the elements to quadratic, and loading all internal data-structures consumes approximately 17% of the total iteration time. This includes writing the mesh to disk and re-reading it for object construction, as this has not been optimized in our code. Meshing time could be potentially reduced by 40–50% for a closer mesher-simulator integration. In contrast, local remeshing techniques require identification and matching of remeshed regions, which scale with increased complexity. Local remeshing is preferred when a large number of variables have to be mapped after remeshing, as it reduces mapping time and diminishes diffusivity – both inherent problems to plasticity dominated simulations. Thus, the benefit of local remeshing is not due to the reduced meshing domain, but to the reduced post-processing steps on mesh renewal. Local remeshing has a very low footprint on low density datasets reducing meshing time to less than 1% of the simulation time, but can increase to up to approximately 40% of total simulation time and node quantity and complexity increase (timings reported include I/O operations: Wicke et al. [69]). In contrast, we show that a total remeshing approach scales linearly with complexity while ensuring high-quality elements at all steps. Thus, the total remeshing approach does not appear significantly less efficient than its local meshing counterpart.

In contrast, computing the displacements takes an average 60% of the iteration time, and deriving the displacement gradients, strains, and stresses can take up 20% of the time if computed for the entire mesh. Computing stress intensity factors using one integration point can take from 0.1 up to 1% of the iteration for cylinder sizes of one to 10 times the local tip length. For 40 integration points, for small cylinders computation might take up to 20 times more, and for larger cylinders (10 times) computation might take up to 200 times more. Thus, for large cylinder sizes, and large number of integration points, the time to compute stress intensity factors becomes comparable to the time for the FEM computation. Therefore, it is of interest to reduce the cylinder size and amount of integration points as much as possible to avoid efficiency penalties. Furthermore, for datasets with an average 1.74% of tip nodes, intersections take an average 4.15% of the computational time. Fig. 10 shows the relationship between tip node density increase and intersection computational time. For datasets with approximately 20 k nodes, values remain below 5%, despite the steady increase in node density. This is mainly due to the high-level optimization of intersection computation, which restricts checks to the new fracture surface strips only.

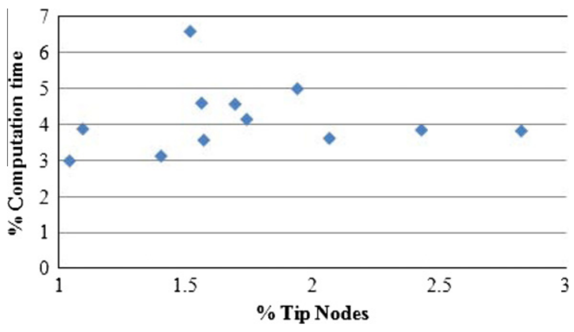

Fig. 10. Computational time for intersection checks and geometric handling as a function of node density.

Table 1
Efficiency summary. All time measurements are performed on an Intel Xeon X5460 with two quad cores running at 3.16 GHz.

Mesh	Nodes	Meshing	Running	Extract shape	Extend nodes
Irregular tetra	10k	14 s (19.7%)	1 min 11 s	<0.1 s	<0.1 s
Irregular tetra	23k	28 s (17.5%)	2 min 40 s	<0.1 s	<0.1 s
Irregular tetra	50k	60 s (17.6%)	5 min 40 s	<0.1 s	2 s
Irregular tetra	175k	180 s (15%)	20 min	0.2 s	5 s
Irregular tetra	360k	460 s (17.8%)	43 min	0.5 s	13 s
[49]					
Irregular tetra	10k	20 s	10 min	–	–
Regular tetra	28k	600 s	30 min	–	–
Bricks	15k	1200 s	120 min	–	–

4.3. Accuracy

It is important to compare the accuracy of the presented method to other numerical solutions available in the literature (see Table 2). For solutions using brick meshes, accuracy in the computation of modal stress intensity factors is reported to be a strong function of mesh refinement: for a center through-going crack an average error of 2.6%, for a horizontal penny shaped crack errors range from 0.5 up to 10.2%, for the embedded elliptic crack errors are in the range from 2 to 5.6%, and for the inclined penny-shaped between 3 and 20% (all reported by FRANC3D, 2003). Rigby and Aliabadi [37], also using brick meshes, report errors as low as 1% for both fine and coarse meshes for the penny shaped crack, and for the inclined penny crack up to 2.1% for K_I, 2.2% for K_{II}, and up to 9.5% for K_{III}. For finite element based solutions also using brick meshes, Banks-Sills [50] reports an average error of 1.3% for both virtual crack extension and J Integral methods, and slightly higher 2.5% error for the displacement extrapolation method, both measured for penny shaped horizontal cracks under tension. Additionally, the authors report differences of up to 4.3% between axis-symmetric and full 3D solutions. For arbitrary mesh computations, Cervenka and Saouma [35] report errors of 4.8–20% for an edge crack in tension, and around 10–15% for the inclined penny shaped crack. Finally, Rajaram [54] showed that for arbitrary meshes, an accuracy of 1–5% can be attained for irregular tetrahedral meshes.

Computation of stress intensity factors along a crack front often leads to oscillating results. Banks-Sills [50] reports oscillations of computed stresses of up to 10% for 20 node brick meshes, which can be greatly reduced but not eliminated to around 1% by increasing the interpolation degree of the used elements and using more accurate 27 node hexahedral elements. Rajaram et al. [38] report that these oscillations are maintained when using tetrahedral meshes, and show that results oscillate systematically ~8% around the solution for regular tetrahedral meshes. The authors also show that oscillations become irregular when using arbitrary tetrahedral meshes [54]. These are attributed to (1) varying node connectivity around crack tip nodes, (2) discontinuous gradients perpendicular to the crack front, (3) perturbations on the q gradient field. Rajaram et al. [38] address this problem by extending the integration domain over four elements, integrating at every other node location, and computing q analytically. They show that this strategy significantly reduces oscillations.

4.4. Measuring discretization error

Kishimoto et al. [66] define three types of errors when computing J: the error in the computation of the nodal displacements, the error in the derivation of the stress and strain fields using the finite element shape functions, and the error incurred during the integration to calculate J. The magnitude of the error can be decreased by increasing mesh refinement, increasing the interpolation degree of elements, and by adapting the shape and size of individual elements to the local solution shape. The discretization error at the element can be expressed as [53]

$$e_E^{abs} = max|\bar{\sigma}^h - \bar{\sigma}^*|_{(ip)} \tag{30}$$

where e_E is the absolute error at the element, $\bar{\sigma}^*$ is the deviatoric stress interpolated from the nodal values onto the integration points, and $\bar{\sigma}^h$ is the deviatoric stress at the integration point. The mesh error can be defined as the maximum e_E over all elements. The relative error, e_E^{rel}, is defined as

Table 2
Accuracy of the numerical integration of the J Integral, reported in the literature. * with respect to observations which have been reported to be accurate within 1–3% [24].

Reference	Edge crack in tension	Center through-going crack	Horizontal penny crack	Elliptical crack	Inclined penny crack
[60]					
Brick mesh	2.6%	0.5–10.2%	2–5%	2–5.6%	3–20%
[45]					
Brick mesh 20n	–	–	1.3%(±10%)	–	–
Brick mesh 27n	–	–	1.3%(±1%)	–	–
[35]					
Tetra mesh	2.5%(±2.5%)				~10%
[49]					
Reg tetra mesh irreg tetra mesh	0.5%* (±l%)	–	0.5%* (±l%)	2%* (+2%)	–
Irreg tetra mesh	0.5%* (±3%)	–	1%* (±3%)	2%* (±4%)	–

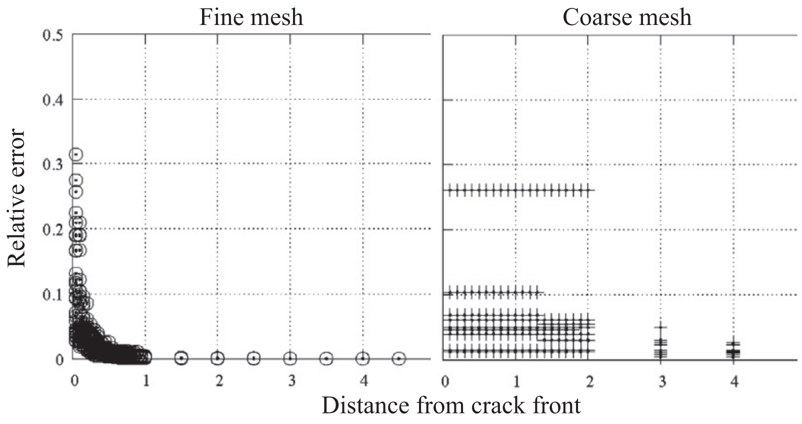

Fig. 11. Discretization error measured away from the tip varies between 3% and 28% for coarse mesh and between ⩽1% and 32% for a fine mesh. Notice that in this example, for the fine mesh, a choice of $r_{cyl} \geqslant 0.5$ carries a maximum discretization error of ⩽5%, while for the coarser mesh we a much larger choice of $r_{cyl} \geqslant 3.5$ is required for an error ⩽5%.

$$e_E^{rel} = \frac{e_E}{\sqrt{(\bar{\sigma}^h_{max})^2 - (\bar{\sigma}^h_{min})^2}} \quad (31)$$

where $\bar{\sigma}^h_{max}$ and $\bar{\sigma}^h_{min}$ are the overall maximum and minimum values of $\bar{\sigma}^h$.

Fig. 11 illustrates how the error increases as a function of proximity to the crack. This affects the minimum size of the integration domain – the smaller the radius, the larger the discretization error proportional to the total value of the integral. For a refined mesh, the error is more pronounced at the close vicinity of the tip, an indicator that sampling too close to the tip may conduce to large discretization errors. However, it is clear from the comparison that a finer mesh allows for proximal sampling beyond the capabilities of a coarser mesh. The definition of e_E^{rel} can be fed into meshing engines in order to create meshes that minimize the overall error thereby improving the overall mesh quality [29]. This behavior might be related to the difficulty of some finite element based methods to generate good stress intensity factor approximations while using small integration domains [38,35], as other authors using mesh-free boundary element methods have shown to compute stress intensity factors using small integration domains [55,37].

5. Numerical experiments: validation

The computation methods presented in Section 3.2 can be validated by comparing the predicted stress intensity factors against analytical solutions for three different geometries: (a) a through-going crack, (b) a horizontal penny crack in a block, and (c) a slanted penny crack in a block. In all cases, pure tension boundary conditions area applied (see Fig. 12).

The results are compared to analytical solutions which have been derived for the specific geometries and boundary conditions. All results are normalized by $\sigma\sqrt{\pi a}$. In all cases material properties are chosen as an elastic modulus of 10000 Pa, and a Poisson's ratio of 0.

5.1. Through-going crack embedded in a thick plate

In Fig. 12a, K is measured along the crack tip of a through-going crack, of radius 1, embedded in a thick plate of dimensions $60 \times 60 \times 60$. Plane-strain conditions are reproduced near the center of the plate, and results can be compared to analytical solutions for center-cracked plane strain 2D [67]. These results match the analytical solution within less than 1% for large r_{cyl} of 10 and 5, and $r_{cyl} \leqslant 5$ the error increases up to 5% for localized, single segment computations; see Fig. 13. The average K along the crack front only differs from the analytical solution by up to 2% for all cylinder sizes. Oscillations become more apparent for smaller cylinder sizes.

5.2. Horizontal penny crack

Fig. 12b illustrates a circular crack embedded in a $400 \times 400 \times 400$ cubic block which is subjected to tension parallel to its surface. The results are compared with a 3D analytical solution derived by Erdogan and Sih [58]. Fig. 14 is a plot of the stress intensity factors along the crack tip front for a fractures of radius 10. Oscillations can be mitigated by sampling over more than one segment, as originally suggested by Rajaram et al. [38]. Fig. 14 shows the effect of extending the 1D integration domain to include two and four more neighbors, and assuming J to be constant or have linear variation

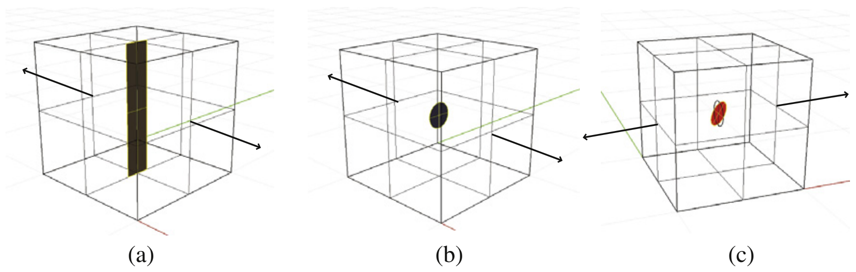

Fig. 12. Experimental setup: (a) a through-going crack, (b) a horizontal penny crack in a block, and (c) a slanted penny crack in a block.

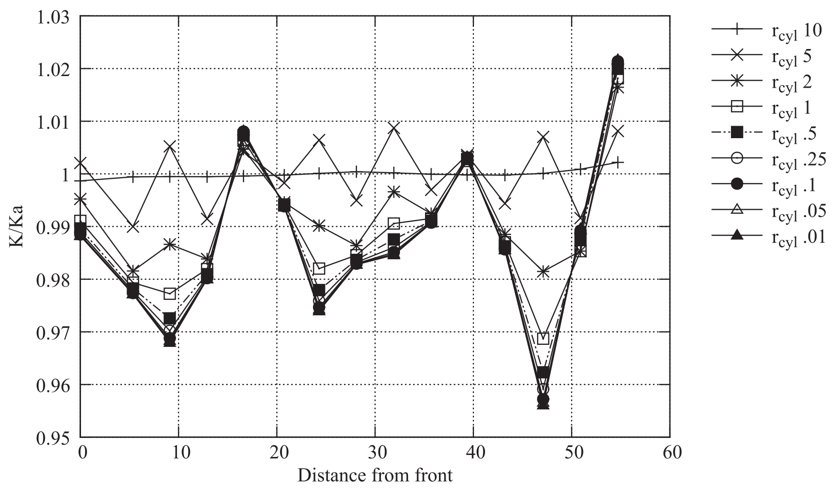

Fig. 13. Stress intensity factor calculation of a through-going crack embedded in a thick plate. Results converge <5% of the analytical solution.

along the tip front. Average values of K agree from 1.7 to 7% for r_{cyl} ranging from 0.7 to $0.8t_s$, results agree with the analytical solution within approximately 5%.

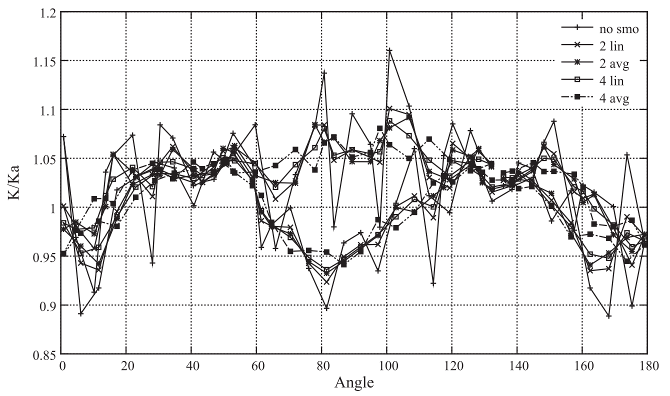

Fig. 14. Normalized stress intensity factor of a penny crack for a crack of radius 10 and an integration domain with $r_{cyl} = 0.7t_s$, for integration domains spanning over two and four neighboring segments, assuming linear and constant J variation. Oscillations in the front, aggravated by the small integration domain, can be reduced by assuming a linear variation of J along the crack front, and taking more than one segment into account for integration.

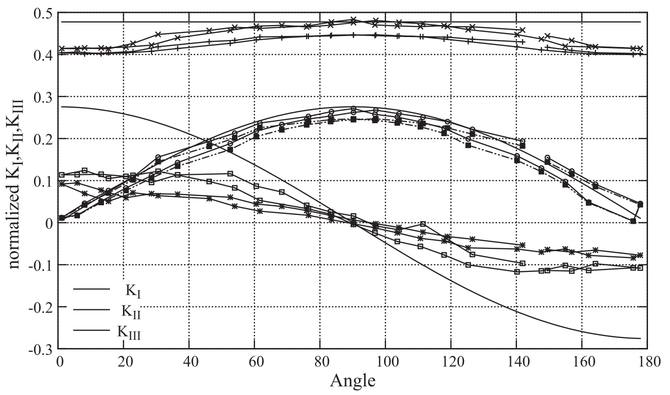

Fig. 15. Modal stress intensity factors of a slanted penny crack. Results are plotted for large cylinder ($r_{cyl} = 5, 10$) values. Approximation of the modal stress intensity factor K_{II} is not accurate for large integration domains as adjacent regions of the crack influence the integration domain.

5.3. Slanted penny crack

Fig. 12c shows a crack slanted at 30°, embedded in a 400 × 400 × 400 block, which is subjected to remote tensile stress. This experiment is used to verify the decomposition of K by comparing results against the analytical solution derived by Cherepanov [68]; Fig. 15 is a plot of the decomposed stress intensity factors using large integration domains of the order of the tip radius. For a fracture of radius 10 embedded in a 400 × 400 × 400 block, we initially integrate using $r_{cyl} = \rho/2$ and $r_{cyl} = \rho$. However, mode decomposition does not properly match the expected analytical solution. This is corrected by choosing a smaller integration domain, $r_{cyl} \approx 0.75t_s$, as shown in Fig. 16. We observe a good correlation between the analytic and numerical solution with an overall error of 5%.

All results are computed using quarter-point tetrahedral elements, and use only six integration points for each crack tip node to predict stress intensity factors. Thus, the approach is computationally inexpensive, and yields good approximations even though integration parameters are simplified.

6. Conclusions

A numerical framework for 3D discrete crack growth has been presented, which is completely automatic and uses arbitrary tetrahedral meshes. Fracture growth depends on three input criteria: a failure criterion, and criteria for propagation magnitude and direction. These are input to the growth model. Geometry is stored independently from the mesh in the form of faceted non-convex polyhedra for crack and matrix bodies, and parametric NURBS curves for crack tips. The stress intensity factor is computed along the crack tips in the reference space of a virtual unit cylinder, and is independent of the surrounding mesh configuration. Thus, the technique can be applied to pure tetrahedral meshes, as well as to hybrid meshes that combine prisms, hexahedra, and tetrahedra arbitrarily. Geometric information is used to guide adaptive mesh refinement. The technique can be applied as a fully local approach, sampling single tip segments. Alternatively, neighborhoods of tips can be sampled to reduce oscillations when relying on small integration domains. The method is validated with analytical solutions, and results agree within 2–5% with predicted values. This is achieved for single segment integration using only six integration points per tip segment. Cylinder size, r_{cyl}, must be small enough not to contain any external singularity, and can be approximated

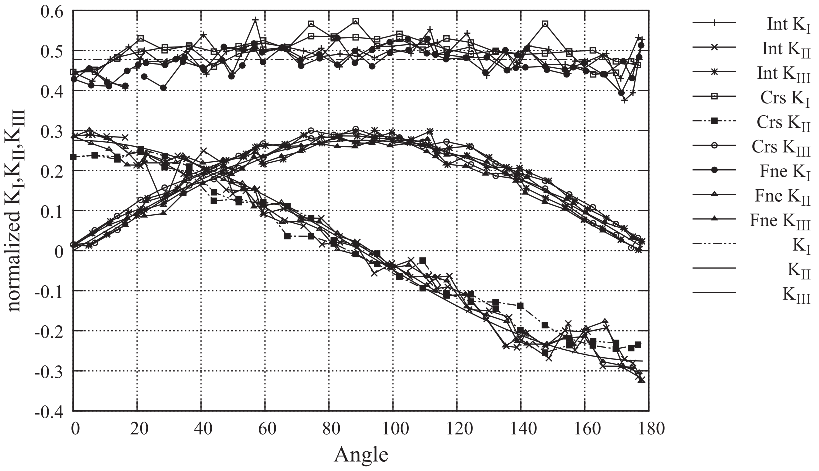

Fig. 16. Modal stress intensity factors of a slanted penny crack for a coarse, fine, and intermediate levels of refinement. A smaller cylinder ($r_{cyl} = 2$) yields a better approximation for modal decomposition, especially K_{II}, which appears independent of mesh refinement.

as $\min(0.25\rho, 0.75t_s, \Delta d_{p_n}/2)$. Additionally, the integration domain defined by r_{cyl} should include elements for which the relative discretization error $e_E^{rel} \leqslant 5\%$. Full mesh generation remains as a viable adaptive remeshing technique as it only consumes a fraction of the total iteration time and generates a high quality, geometry compliant mesh at each step. We presented a finite element based, straightforward 3D fracture propagation technique which exploits an efficient and accurate approach to approximate stress intensity factors locally.

Acknowledgements

The authors thank Rio Tinto for supporting this work, through the Rio Tinto Centre for Advanced Mineral Recovery at Imperial College. We would also like to thank Victor Saouma for valuable discussions and thoughtful comments on various aspects of this work.

References

[1] R. de Borst, J.J.C. Remmers, A. Needleman, M.-A. Abellan, Discrete vs smeared crack models for concrete fracture: bridging the gap, Int. J. Numer. Anal. Methods Geomech. 28 (2004) 583–607.
[2] M. Jirásek, Nonlocal models for damage and fracture: comparison of approaches, Int. J. Solids Struct. 35 (1998) 4133–4145.
[3] S. Bordas, V. Nguyen, C. Dunant, H. Nguyen-Dang, A. Guidoum, An extended finite element library, Int. J. Numer. Methods Engrg. 71 (2007) 703–732.
[4] E. Wyart, D. Coulon, T. Pardoen, J. Remacle, F. Lani, Application of the substructured finite element/extended finite element method to the analysis of cracks in aircraft thin walled structures, Engrg. Fract. Mech. 76 (1) (2009) 44–58.
[5] T. Rabczuk, T. Belytschko, A three-dimensional large deformation meshfree method for arbitrarily evolving cracks, Comput. Methods Appl. Mech. Engrg. 196 (2007) 2777–2799.
[6] S. Bordas, T. Rabczuk, G. Zi, Three-dimensional crack initiation, propagation, branching and junction in non-linear materials by an extended meshfree method without asymptotic enrichment, Engrg. Fract. Mech. 75 (5) (2008) 943–960.
[7] T. Rabczuk, T. Belytschko, Cracking particles: a simplified meshfree method for arbitrarily evolving cracks, Int. J. Numer. Methods Engng. 61 (2004) 2316–2343.
[8] B.J. Carter, P.A. Wawrzynek, A.R. Ingraffea, Automated 3-D crack growth simulation, Int. J. Numer. Methods Engrg. 47 (2000) 229–253.
[9] FRANC3D: Concepts/Users Guide. FRANC3D Version 2, 1998. Cornell University, New York. Available from: <http://www.cfg.cornell.edu>.
[10] X.B. Lin, R.A. Smith, Finite element modelling of fatigue crack growth of surface cracked plates. Part I: The numerical technique, Engrg. Fract. Mech. 63 (1999) 503–522.
[11] M. Schöllmann, M. Fulland, H.A. Richard, Development of a new software for adaptive crack growth simulations in 3D structures, Engrg. Fract. Mech. 70 (2003) 249–268.
[12] H.M. Nick, A. Paluszny, S.K. Matthai, M. Blunt, Influence of fracture aperture on solute transport in porous media, Geo Halifax, Halifax, Nova Scotia, Canada, September 20–24, 2009.
[13] A. Paluszny, S.K. Matthai, Impact of fracture development on the effective permeability of porous rocks as determined by 2D discrete fracture growth modeling, J. Geophys. Res. – Solid Earth 115 (2010).
[14] W.T. Riddell, A.R. Ingraffea, P.A. Wawrzynek, Experimental observations and numerical predictions of three-dimensional fatigue crack propagation, Engrg. Fract. Mech. 58 (4) (1997) 293–310.
[15] C. Timbrell, P.W. Claydon, G. Cook, Application of ABAQUS to analysis of 3d cracks and fatigue crack growth prediction, in: ABAQUS Users Conference Proceedings. Newport, RI, 1–3 June, 1994, pp. 527–541.
[16] G. Dhondt, Automatic three-dimensional cyclic crack propagation predictions with finite elements at the design stage of an aircraft engine, in: Applied Vehicle Technology Panel Symposium on Design Principles and Methods for Aircraft Turbine Engines (NATO-RTO). France: Toulouse, 1998, pp. 331–338.
[17] D. Bremberg, G. Dhondt, Automatic 3-D crack propagation calculations: a pure hexahedral element approach versus a combined element approach, Int. J. Fract. 157 (2009) 109–118.
[18] E. Gürses, C. Miehe, A computational framework of three-dimensional configurational-force-driven brittle crack propagation, Comput. Methods Appl. Mech. Engrg. 198 (2009) 1413–1428.
[19] G.P. Cherepanov, The propagation of cracks in a continuous media, J. Appl. Math. Mech. 31 (1967) 503–512.
[20] J.R. Rice, A path independent integral and approximate analysis of strain concentration by notches and cracks, J. Appl. Mech. (1968) 379–386.
[21] D.M. Parks, A stiffness derivative finite element technique for determination of crack tip stress intensity factors, Int. J. Fract. 10 (1974) 487–502.
[22] T.K. Hellen, On the method of virtual crack extensions, Int. J. Numer. Methods Engrg. (1975) 187–207.
[23] C.G. Huang, P.A. Wawrzynek, A.R. Ingraffea, On the virtual crack extension method for calculating the derivatives of energy release rates for a 3D planar crack of arbitrary shape under mode I loading, Engrg. Fract. Mech. 68 (2001) 925–947.
[24] I.S. Raju, J.C. Newman, Stress-intensity factors for a wide range of semi-elliptical surface cracks in finite-thickness plates, Engrg. Fract. Mech. 11 (1979) 817–829.
[25] M. Isida, H. Noguchi, Tension of a plate containing an embedded elliptical crack, Engrg. Fract. Mech. 20 (3) (1984) 387–408.
[26] S.K. Chan, I.S. Tuba, W.K. Wilson, On the finite element method in linear elastic mechanics, Engrg. Fract. Mech. 2 (1970) 1–17.
[27] H. Okada, H. Kawai, K. Araki, A virtual crack closure-integral method (VCCM) to compute the energy release rates and stress intensity factors based on quadratic tetrahedral finite elements, Engrg. Fract. Mech. 75 (2008) 4466–4485.
[28] L. Banks-Sills, Update: application of the finite element method to linear elastic fracture mechanics, Appl. Mech. Rev. (2010) 63.
[29] C.C. Pain, A.P. Umpleby, C.R.E. de Oliveira, A.J.H. Goddard, Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations, Comput. Methods Appl. Mech. Engrg. 190 (29-30) (2001) 3771–3796.
[30] S.J. Owen, A survey of unstructured mesh generation technology, in: Proceedings of 7th international mesh roundtable, Sandia National Laboratories, Dearborn, MI, 1998.
[31] B.M. Klingner, J.R. Shewchuk, Aggressive Tetrahedral Mesh Improvement, in: Proceedings of the 16th International Meshing Roundtable, Seattle, Washington, 2007, pp. 3–23.
[32] F. Labelle, J.R. Shewchuk, Isosurface stuffing: fast tetrahedral meshes with good dihedral angles, ACM Trans. Graph. 26 (3) (2007) 57.1–57.10. August 2007. Special issue on Proceedings of SIGGRAPH 2007.
[33] H. Si, Constrained Delaunay tetrahedral mesh generation and refinement, Finite Elem. Anal. Design 46 (1–2) (2010) 33–46.
[34] G.P. Nikishkov, S.N. Atluri, Calculation of fracture mechanics parameters for an arbitrary three-dimensional crack, by the 'equivalent domain integral' method, Int. J. Numer. Methods Engrg. 24 (1987) 1801–1821.
[35] J. Cervenka, V.E. Saouma, Numerical evaluation of 3-D SIF for arbitrary finite element meshes, Engrg. Fract. Mech. 57 (1997) 541–563.
[36] H. Ishikawa, H. Kitagawa, H. Okamura, J-Integral of a mixed mode crack and its application, in: Proceedings of the 3rd International Conference on Mechanical Behaviour of Materials, Pergamon, Oxford, vol. 3, 1979, pp. 447–455.
[37] R.H. Rigby, M.H. Aliabadi, Decomposition of the mixed-mode J-Integral – Revisited, Int. J. Solids Struct. 35 (17) (1998) 2073–2099.
[38] H. Rajaram, S. Socrate, D.M. Parks, Application of domain integral methods using tetrahedral elements to the determination of stress intensity factors, Engrg. Fract. Mech. 66 (2000) 455–482.
[39] J.A. Cottrell, T.J.R. Hughes, Y. Bazilevs, Isogeometric Analysis: Toward Integration of CAD and FEA, John Wiley & Sons, 2009.
[40] A. Paluszny, S.K. Matthai, Numerical modeling of discrete multi-crack growth applied to pattern formation in geological brittle media, Int. J. Solids Struct. 46 (2009) 3383–3397.
[41] A.R. Ingraffea, P.A. Wawrzynek, Finite Element Methods for Linear Elastic Fracture Mechanics, in: Comprehensive Structural Integrity, Elsevier Science Ltd, 2003.
[42] A. Paluszny, Numerical simulation of fracture pattern development and implications for fluid flow. PhD Thesis. Imperial College London, 2009.
[43] L.F. Martha, P.A. Wawrzynek, A.R. Ingraffea, Arbitrary crack representation using solid modeling, Engrg. Comput. 9 (1993) 63–68.
[44] K.J. Versprille, Computer-Aided Design Applications of the Rational B-Spline Approximation Form. PhD Thesis, Syracuse University, New York, 1975.
[45] F. Cazals, M. Pouget, Smooth surfaces, umbilics, lines of curvatures, foliations, ridges and the medial axis: selected topics, Int. J. Comput. Geometry Appl. 15 (5) (2005) 511–536.
[46] H. Prautzsch, W. Boehm, M. Paluszny, Bezier and B-Spline Techniques, Springer, Berlin, 2002.
[47] D.J. Filip, T.W. Ball, Procedurally representing lofted surfaces, IEEE Comput. Graph. Appl. (1989) 2733.
[48] R.D. Cook, D.S. Malkus, M.E. Plesha, Concepts and Applications of Finite Element Analysis, third ed., John Wiley & Sons, 1989.
[49] J.C. Jaeger, N.G.W. Cook, R.W. Zimmerman, Fundamentals of Rock Mechanics, fourth ed., Blackwell Publishing, 2007.
[50] L. Banks-Sills, Application of the finite element method to linear elastic fracture mechanics, Appl. Mech. Rev. 44 (1991) 447–461.
[51] D.A. Field, The legacy of automatic mesh generation from solid modeling, Comput. Aided Geometric Design 12 (1995) 651–673.
[52] J. Cervenka, Discrete crack modeling in concrete structures. PhD. Thesis, University of Colorado, Boulder, 1994.
[53] O.C. Zienkiewicz, J.Z. Zhu, The superconvergent patch recovery and a posteriori error estimates. Part 1. The recovery technique, Int. J. Numer. Methods Engrg. 33 (1992) 1331–1364.
[54] H. Rajaram, Three-dimensional fracture mechanics computations using tetrahedral finite elements. PhD Thesis, Massachusetts Institute of Technology, Cambridge, 1997.
[55] K.N. Shivakumar, I.S. Raju, An equivalent domain integral for three-dimensional mixed mode fracture problems, NASA CR-182021, 1990.
[56] D. Broek, Elementary Engineering Fracture Mechanics, Kluwer Academic Publishers, 1986.

[57] R.J. Charles, Static fatigue of glass, J. Appl. Phys. 29 (1958) 1549–1560.
[58] F. Erdogan, O.C. Sih, On the crack extension in plates under plane loading and plane shear, J. Basic Engrg. 85 (1963) 519–527.
[59] A.M. Al-Ajmi, R.W. Zimmerman, Relation between the Mogi and the Coulomb failure criteria, Int. J. Rock Mech. Mining Sci. 42 (2005) 431–439.
[60] W. Weber, G. Kuhn, An optimized predictor-corrector scheme for fast 3D crack growth simulations, Engrg. Fract. Mech. 75 (3-4) (2008) 452–460.
[63] S.K. Matthai, S. Geiger, S.G. Roberts, The complex systems platform CSP3D3.0: User's Guide. Technical report, ETH Zurich Research Reports, 2001.
[64] K. Stüben, A review of algebraic multigrid, J. Comput. Appl. Math. 128 (2001) 281–309.
[66] K. Kishimoto, N. Tkeuchi, S. Aoki, M. Sakata, Computational accuracy of the J-Integral by the finite element method, Int. J. Pres. Ves. Piping 44 (1990) 255–266.
[67] P.A. Wawrzynek, B.J. Carter, L. Banks-Sills, The M-Integral for computing stress intensity factors in generally anisotropic materials, NASA Consultant Reports, 2005.
[68] G.P. Cherepanov, Mechanics of Brittle Fracture, McGraw-Hill, New York, 1979.
[69] M. Wicke, D. Ritchie, B. Klingner, S. Burke, J. Shewchuk, J. O'Brien, Dynamic local remeshing for elastoplastic simulation, ACM T. Graphic 29 (2010) 4.