Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Incremental, persistent and concurrent finite element connectivity for mesh cutting and crack growth

Pedro Areias

pedro.areias@tecnico.ulisboa.pt

Instituto Superior Técnico

Nuno Silvestre

Instituto Superior Técnico

Research Article

Keywords:

Posted Date: November 27th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11788742/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Incremental, persistent and concurrent finite element connectivity for mesh cutting and crack growth

P. Areias[1,2*] and N. Silvestre[1,2]

[1*]Department of Mechanical Engineering (DEM), Instituto Superior Técnico, Universidade de Lisboa, Av. Rovisco Pais 1, 1049-001, Lisboa, Portugal.
[2]IDMEC, Instituto Superior Técnico, Universidade de Lisboa, Av. Rovisco Pais 1, 1049-001, Lisboa, Portugal.

*Corresponding author(s). E-mail(s): pedro.areias@tecnico.ulisboa.pt;

Abstract

In three-dimensional crack growth, the solver connectivity of a conforming mesh is usually recomputed at every increment. We update it after each local edit at a cost per edited element independent of the mesh size, plus a directory copy per commit. Counting relations (counts of the elements that contain each node pair) give exact update rules, with proofs, for inverse incidence, the sparsity pattern, assembly destinations and facet adjacency; unedited elements keep their destinations. Versions are committed and restored in constant time, so one rollback undoes a rejected growth step. A local Steiner-point-free cutter, proved equivalent to our batch cutter, inserts and grows cracks and cuts independent edge groups concurrently. With elements of 1/64 of the crack radius, the stress intensity factors, which the interaction integral averages along the front, differ from the equally averaged penny-crack solutions of Sneddon and of Kassir and Sih by at most 0.29 % of the normal-tension factor K_0. With a free-surface term, two surface cracks give factors within 3.14 and 1.53 % of Newman and Raju's values on the finest meshes. A corner crack grown by the Erdogan–Sih criterion on 6.5 million tetrahedra demonstrates cutting and versioning; its path has no reference solution. On 10^7 tetrahedra, updating the connectivity after a crack advance by node duplication takes 3.0 ms, against 849 ms for recomputation by our earlier framework and 300 ms for a tuned 16-thread recomputation. In the Paris-law examples, fitting, crack increment and commit take 0.5 to 1.6 % of a step.

Keywords: crack growth, mesh cutting, mesh data structures, sparse matrix assembly, incremental algorithms, stress intensity factors

Article Highlights

- Solver connectivity is updated per edit at a cost set by the edited neighborhood, plus a directory copy per commit.
- Mesh versions are committed and restored in constant time, so one rollback undoes a rejected crack-growth step.
- On the finest mesh, penny-crack stress intensity factors match averaged solutions to 0.29 % of the normal-tension value.

1 Introduction

Simulations of fatigue crack growth in three-dimensional components advance the crack front increment by increment [15]. The extended finite element method keeps the mesh but adds enriched unknowns as the front advances [51, 71], which changes the sparsity pattern. With a conforming mesh, every increment changes the mesh topology near the front. Remeshing codes regenerate the elements near the front,

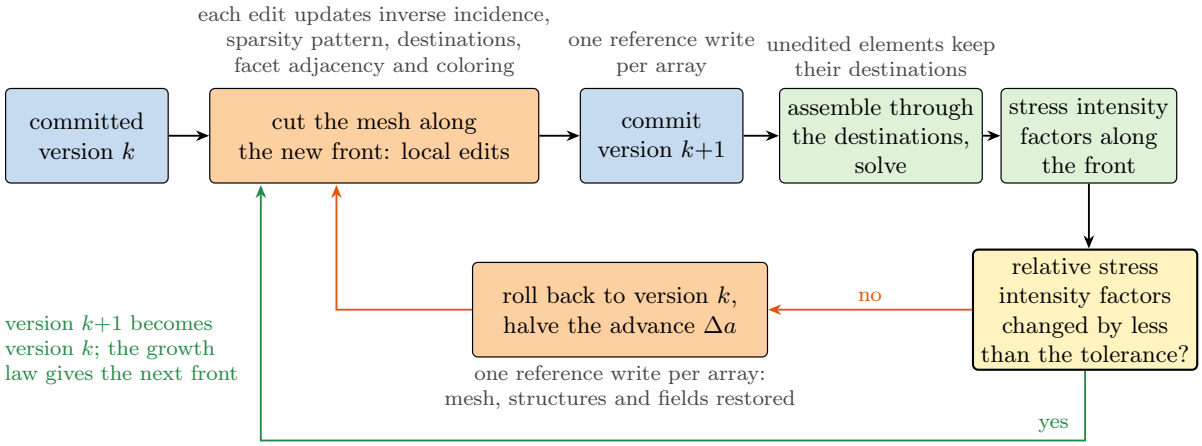


Fig. 1 One growth step on the maintained structures. In the examples with planar cracks the mesh is first fitted to the front (Section 8.1). The mesh is then cut along the new front by local edits (Algorithm 1), and the result is committed as version $k+1$. A step in which the relative stress intensity factors (each factor divided by the mean along the front) change by more than a tolerance is rolled back to version k and repeated with half the advance Δa (Section 8.3). An increment that fails the checks of Section 5.3 is also rolled back.

or the whole mesh, at each increment [18, 49, 60, 65]. Remeshing near the front also transfers the fields and recomputes the solver structures [18]. Crack insertion duplicates nodes and reassigns elements [53, 61] or superposes elements on phantom nodes [69]; local refinement replaces elements by children [9, 33, 39]; erosion deletes elements. Each of these changes invalidates the solver connectivity derived from the element–node lists: the elements at each node with their local positions (inverse incidence), the global sparsity pattern, the pattern slot of each element-matrix entry (assembly destinations), facet adjacency, boundary facets and the element coloring for threaded assembly without write conflicts. The change is local, but each of these structures spans the mesh, so recomputing them costs time proportional to the mesh size, at every increment. When a growth law rejects a step, the mesh, its derived structures and the nodal fields must also be restored exactly.

We maintain these structures under local edits of single elements instead of recomputing them (Figure 1). A crack increment is applied as a sequence of such edits, and each edit updates all the structures above. The result is committed as a version, and the stiffness matrix is assembled through the destinations that unedited elements keep. After the solve, the stress intensity factors of the new front decide whether the step is accepted. A rejected step is undone by one rollback, which restores the mesh, the structures and the fields of the previous version. The method combines counting relations, which give exact local update rules, with persistent versions that share unchanged storage. We know of no earlier method that maintains inverse incidence, the sparsity pattern, assembly destinations, facet adjacency and a coloring under element edits that are not drawn from a pool prepared before the run, and restores all of them, with the mesh and the fields, from a committed version.

Mesh software either updates explicit adjacency locally or derives it in bulk. Half-edge-type structures and their array-based half-facet form [24] store adjacency explicitly. Mesh databases update it locally: the distributed mesh database of Seol and Shephard [66], the modifiable arrays of Ibanez and Shephard [34, 35], and TopS and ParTopS, which insert cohesive elements locally and in parallel [26, 63]. Adaptation codes do the same: Mahmoud et al. [48] update patches on the GPU, and Gorman et al. [28] keep node–node and node–element lists current on independent sets. None of these maintains a sparsity pattern with stable slots (entry positions that edits do not move), assembly destinations or a coloring for assembly. Libraries that derive adjacency in bulk by transposition and intersection [46] or cone–support traversals [41, 45] describe no incremental update after an edit. Adaptive mesh and finite element libraries refine and coarsen meshes locally and in parallel [2, 17, 40]; after each adaptation the degree-of-freedom map and the sparsity pattern are recomputed on the new mesh. MM3 [8], our earlier framework of typed mesh relations, also derives adjacency in bulk. It caches the transpose and position tables and recomputes them after an edit, at reported costs of 56 ms (transpose) and 173 ms (with the position tables) for 10^6 quadrilaterals on a laptop.

Sparse-matrix software offers formats with slack for insertion [74] and methods that update factorizations [20, 59], but neither derives topology-based patterns nor preserves assembly destinations. The closest work on solver structures is the incremental matrix assembly of Acharya and Hyde [1] for preplanned edits of tetrahedral meshes. The elements that a simulation may use form a pool allocated before

the run. One fixed compressed-row layout holds the entries of every element of the pool, with the position of each element's entries precomputed. A count of active elements is kept at each entry, and the values are written in place into the solver's array. Assembly is 6 to 27 times faster than a full reassembly for a graph Laplacian and 2 to 17 times for linear elasticity, on meshes of up to 4.6×10^5 tetrahedra (10^4 for elasticity). Edits outside the pool are not covered, and no earlier version is kept. Here the edits are not drawn from a pool, and inverse incidence, facet adjacency, the coloring and committed versions are maintained with the pattern and the destinations.

Incremental view maintenance updates a materialized view from changes to its base relations [13, 31, 32]; counting the derivations of each entry makes deletions exact [32]. With integer multiplicities, products of relations are bilinear, so every product has an exact update formula [42, 43]; the construction extends to matrix products [57] and to computations on streams of changes [16]. Persistent data structures keep earlier versions by copying only the path from the root to a changed entry [23, 58]. Graph-processing and database systems use them so that readers work on a fixed version while a writer prepares the next one [11, 22, 47, 54]. Blelloch et al. represent triangulations persistently [14], but not the solver structures derived from them.

Mesh cutting also costs time proportional to the mesh size. The Steiner-point-free *batch cutter* of [7] builds on [6] and places new nodes only on edges of the existing mesh. On each call it sorts all edges and builds a new mesh, regardless of the size of the cut. Its reported cost is 11.7 s for 3.7×10^7 tetrahedra, with one cut per edge per call. MM3 crack insertion [8] scans every edge, refines through a closure loop, duplicates crack nodes and regenerates its caches. The local remeshing framework of Jiang et al. [37] operates on small neighborhoods, rolls back operations that violate invariants and processes disjoint neighborhoods in parallel under vertex locks, but maintains no assembly structures. Cutting the mesh along the crack gives conforming crack faces, and when the cut is local, the nodes, fields and assembly destinations outside the cut band are unchanged. The cutter proposed here costs time proportional to the number of elements the increment crosses, leaves a valid solver connectivity after the cut, reverts locally, and cuts separate cracks or regions concurrently.

We extend the typed relations of MM3 with counting relations and with persistent versions that share unchanged chunks. Inverse incidence, position tables, the node–node pattern, assembly destinations, facet adjacency, boundary facets and coloring are then updated after every unplanned local edit. The cost per edited element depends on the largest numbers of elements and of adjacent nodes at a node, but not on the size of the mesh (Theorem 6). The first edit after a commit adds a copy of the directories of the arrays (Section 4). The update rules read only the old and new rows of the edited element and the maintained structures at its nodes (Section 3). Pair counts make deletions exact, and stable slots preserve the destinations of unedited elements (Section 3). Copy-on-write chunks keep committed versions immutable, so readers never wait for the thread that edits, and a rejected step is undone by one rollback (Section 4). A local cutter built on these edits inserts and grows cracks (Section 5), and stress intensity factors are computed on the resulting meshes (Section 8). Maintenance shortens the total run time most where topology updates are a substantial fraction of each step. This is the case for steps with cheap solves, such as the model step of Section 9.6, and for algorithms that interleave queries with edits, such as recursive bisection (Section 9.4). Recomputation is cheaper when a batch of random node duplications on 10^6 elements edits more than 0.30 to 2.8 % of them (Section 9.3). In the Paris-law growth examples (Sections 8.3 and 8.4) the solve dominates, and fitting the mesh to the front, the crack increment and the commit take 0.5 to 1.6 % of a step. There maintenance saves little time; the committed versions provide the rollback of rejected steps, and the local cut keeps the nodal fields of the nodes outside the cut. The refined band that the crack crosses is chosen before the run; the edits inside it are not planned (Section 8). The implementation is open source [4].

Contributions.

We provide:

1. exact update rules, with proofs, for the transpose with node positions, the element position table, the counting node–node product with stable slots and the facet-sharer relation, and a local repair that keeps the coloring proper; unedited elements keep their assembly destinations, and the cost of an edit depends on the lengths of the longest transpose and pattern rows but not on the number of elements or nodes (apart from the first writes after a commit and the growth of the arrays);
2. persistent versions of all structures and registered attributes through copy-on-write chunks, with constant-time commit and rollback and lock-free snapshot reads;
3. a local Steiner-point-free cutter that reverts locally, inserts and grows cracks from two level sets and cuts independent edge groups concurrently, with proofs that the local cut gives the rows of the batch cutter of [7] and that concurrent cuts of independent groups give those of the sequential cut;

4. stress intensity factors by the domain interaction integral on the maintained structures, with a surface term for fronts that end on a free surface, verified against the solutions of Sneddon [68], Kassir and Sih [38] and Irwin [36] averaged along the front with the weight of the integral, and compared with the finite element values of Newman and Raju [55] and their empirical equation [56];
5. Paris-law growth in which the mesh is rolled back when the relative stress intensity factors (each divided by the mean along the front) change by more than a tolerance over a step, and, as a demonstration of cutting and versioning for which no reference solution exists, Erdogan–Sih growth of a corner crack on 6.5×10^6 tetrahedra, computed on one node of the Deucalion supercomputer.

Relative to our earlier work, the relations, their definitions and their evaluation from scratch are those of MM3 [8], and the templates and label rules of the cutter are those of the batch cutter [7]. The counting update rules and their proofs, the stable slots, the committed versions, the local cut with its revert, the crack increments by a walk from the previous front and the concurrent cut are new. After each batch every structure is compared with an independent program that evaluates the definitions from the rows (Section 7). Sections 2 to 7 give the relations, update rules, versions, cutting, implementation and verification, Section 8 the crack examples and Section 9 the measurements.

2 Relations and derived structures

Let $\mathcal{E} = \{0, \dots, n_E - 1\}$ be the element indices and $\mathcal{N} = \{0, \dots, n_N - 1\}$ the node indices. The primary relation A assigns to each element e a row $\mathsf{A}[e] = (a_{e,0}, \dots, a_{e,k-1})$ of k distinct nodes (at most k in a mixed mesh); row order gives local node numbering and element orientation. A deleted element has the empty row; elements with nonempty rows are *live*. Indices remain stable under edits. Global renumbering (compression) is a separate operation.

The transpose with node positions T assigns to each node v the list of pairs

$$\mathsf{T}[v] = \big((e, p) \ : \ a_{e,p} = v\big), \tag{1}$$

sorted by e; the first component identifies an element containing v (inverse incidence), and the second gives the local position of v. The element position table is

$$\mathrm{EP}(e, p) = \text{index of } (e, p) \text{ in } \mathsf{T}[a_{e,p}], \tag{2}$$

which permits direct reads along element → node → element. The counting relation of A is the incidence matrix $\mathsf{A}^{\#} \in \{0, 1\}^{n_E \times n_N}$ with ${\mathsf{A}^{\#}}_{ev} = 1$ exactly when v occurs in $\mathsf{A}[e]$. The node–node counting product

$$\mathsf{P}^{\#} = {\mathsf{A}^{\#}}^{\mathsf{T}} \mathsf{A}^{\#}, \qquad {\mathsf{P}^{\#}}_{vw} = \#\{e \ : \ v, w \in \mathsf{A}[e]\}, \tag{3}$$

counts the elements that contain both v and w; its support $\operatorname{supp} \mathsf{P}^{\#}$ is the sparsity pattern of the assembled operator at node granularity. The pattern is stored row by row; the position of column w in row v is its slot $s_v(w)$. The destination table

$$\mathrm{D}(e, i, j) = s_{a_{e,i}}(a_{e,j}) \tag{4}$$

maps local entry i, j to its pattern slot. Assembly of S from element matrices S_e uses $S[\mathrm{ptr}_{a_{e,i}} + \mathrm{D}(e, i, j)] \mathrel{+}= S_e[i, j]$, with ptr_v the start of row v, as in the clique table of MM3 [8]. For simplices the facet f of e is the face opposite local node f, with node set $W_{e,f} = \{a_{e,p} : p \neq f\}$. The facet-sharer relation is

$$\Sigma(W) = \bigcap_{v \in W} \mathsf{T}_{\mathcal{E}}[v], \tag{5}$$

where $\mathsf{T}_{\mathcal{E}}[v]$ is the set of elements in $\mathsf{T}[v]$, and the facet adjacency is $\mathrm{FA}(e, f) = \min\big(\Sigma(W_{e,f}) \setminus \{e\}\big)$, or -1 when the set is empty. Only elements with as many nodes as e count as sharers of a facet of e, so facets are not matched across element types. When every facet belongs to at most two elements, $\Sigma(W_{e,f}) \setminus \{e\}$ has at most one element. The boundary facets are the pairs (e, f) of live elements with $\mathrm{FA}(e, f) = -1$. A coloring $\mathrm{col} : \mathcal{E} \to \{0, 1, 2, \dots\}$ is proper when elements that share a node (*neighbors*) have different colors, so that elements of one color can be assembled concurrently without write conflicts [19].

MM3 [8] evaluates (1), (2), (3) and (4) from scratch, by counting sort, binary search, a marker-array product and a shared-element query, and evaluates them again on demand after an edit. An *edit* of element e replaces its row O by N; creation has $O = \emptyset$, deletion $N = \emptyset$, and node duplication for crack insertion replaces some entries of O by new node indices. A *batch* is a sequence of edits followed by a commit (Section 4).

3 Update rules

A one-row edit changes the structures of Section 2 through local update rules. Theorem 6 proves exactness and bounds the cost; Algorithm 1 gives the update.

3.1 Counting relations and the rank-one edit

An edit of e changes one row of the incidence matrix:

$$\Delta\mathsf{A}^{\#} = \boldsymbol{\iota}_e\, \boldsymbol{r}^{\mathsf{T}}, \qquad \boldsymbol{r} = \boldsymbol{\chi}_N - \boldsymbol{\chi}_O, \tag{6}$$

with $\boldsymbol{\iota}_e$ the unit vector of element e and $\boldsymbol{\chi}_X \in \{0,1\}^{n_N}$ the indicator vector of the node set X. Products of counting relations are bilinear: for $\mathsf{C} = \mathsf{XY}$,

$$\Delta\mathsf{C}^{\#} = \Delta\mathsf{X}^{\#}\,\mathsf{Y}^{\#} + \mathsf{X}^{\#}\,\Delta\mathsf{Y}^{\#} + \Delta\mathsf{X}^{\#}\,\Delta\mathsf{Y}^{\#}, \tag{7}$$

the delta rule of incremental view maintenance for joins [32, 42]. The Boolean relations of MM3 are the supports of these counting relations; a pair enters or leaves the support when its count crosses zero, which makes deletion exact [32].

Proposition 1 (Node–node product)**.** *For the edit* (6)*,*

$$\Delta\mathsf{P}^{\#} = \boldsymbol{\chi}_N\boldsymbol{\chi}_N^{\mathsf{T}} - \boldsymbol{\chi}_O\boldsymbol{\chi}_O^{\mathsf{T}}. \tag{8}$$

Proof. Apply (7) with $\mathsf{X} = \mathsf{A}^{\mathsf{T}}$, $\mathsf{Y} = \mathsf{A}$. Since ${\mathsf{A}^{\#}}^{\mathsf{T}}\boldsymbol{\iota}_e = \boldsymbol{\chi}_O$, the three terms are $\boldsymbol{r}\boldsymbol{\chi}_O^{\mathsf{T}}$, $\boldsymbol{\chi}_O\boldsymbol{r}^{\mathsf{T}}$ and $\boldsymbol{r}\boldsymbol{r}^{\mathsf{T}}$; their sum is $(\boldsymbol{\chi}_O + \boldsymbol{r})(\boldsymbol{\chi}_O + \boldsymbol{r})^{\mathsf{T}} - \boldsymbol{\chi}_O\boldsymbol{\chi}_O^{\mathsf{T}}$ and $\boldsymbol{\chi}_O + \boldsymbol{r} = \boldsymbol{\chi}_N$. □

The edit subtracts the old row's clique (all node pairs) and adds the new row's clique. Pairs common to both rows cancel. The cost is $\mathcal{O}(k^2)$ count updates, independent of n_E and n_N.

3.2 Transpose, positions and facets

Proposition 2 (Transpose and positions)**.** *An edit of e changes $\mathsf{T}[v]$ only for $v \in O \cup N$: the pair of e is removed for $v \in O \setminus N$, inserted in sorted order for $v \in N \setminus O$, and its second component is updated for $v \in O \cap N$.* EP *changes only at (e, p) for all p and at entries following an insertion or removal, whose indices shift by $+1$ or -1; the node positions stored in the shifted pairs locate these entries.*

Proof. Relation (1) is the transpose of A with positions attached; $\Delta\mathsf{A}^{\#}$ is supported on row e, so only the columns v with $(\boldsymbol{r})_v \neq 0$ change membership, and for $v \in O \cap N$ only the attached position can change. In a sorted row an insertion or removal at index i_0 shifts exactly the entries after i_0. For such an entry $(e', p_{e'})$ the stored $p_{e'}$ is the local position of v in $\mathsf{A}[e']$, so $\mathrm{EP}(e', p_{e'})$ is written without a search. □

Proposition 3 (Facet sharers)**.** *Let W be a nonempty node set. The edit of e changes $\Sigma(W)$ if and only if $W \subseteq O$ and $W \not\subseteq N$, or $W \subseteq N$ and $W \not\subseteq O$. Consequently* FA *changes only for e itself and for the elements $e' \neq e$ sharing a facet of O or of N that is not a facet of both.*

Proof. By (5), $e \in \Sigma(W)$ before the edit if and only if $W \subseteq O$, and after if and only if $W \subseteq N$; no other element changes membership. The facet entries of $e' \neq e$ depend on $\Sigma(W_{e',f})$ only. □

Before modifying the transpose rows, the update collects the elements that share a facet in O but not in N. It then recomputes the facet entries of these elements, of the elements sharing a new facet in N, and of e. For tetrahedra, each recomputation merges three sorted transpose rows. Facets common to O and N keep their entry and their boundary status. The boundary set is a sparse set (a list and a position table), so a facet enters or leaves it in amortized constant time.

Lemma 4 (Color repair)**.** *Let* col *be proper before the edit of e. If no other element containing a node of $N \setminus O$ has color* col(e)*,* col *is proper after the edit; otherwise, and for a created element, assigning to e the smallest color not used by its current neighbors makes it proper.*

Proof. Only the neighborhood of e changes. Elements sharing a node of $O \cap N$ with e were neighbors before and already differ from col(e). Removing neighbors cannot create a conflict. The remaining new neighbors share a node of $N \setminus O$. The second statement is the greedy rule restricted to e. □

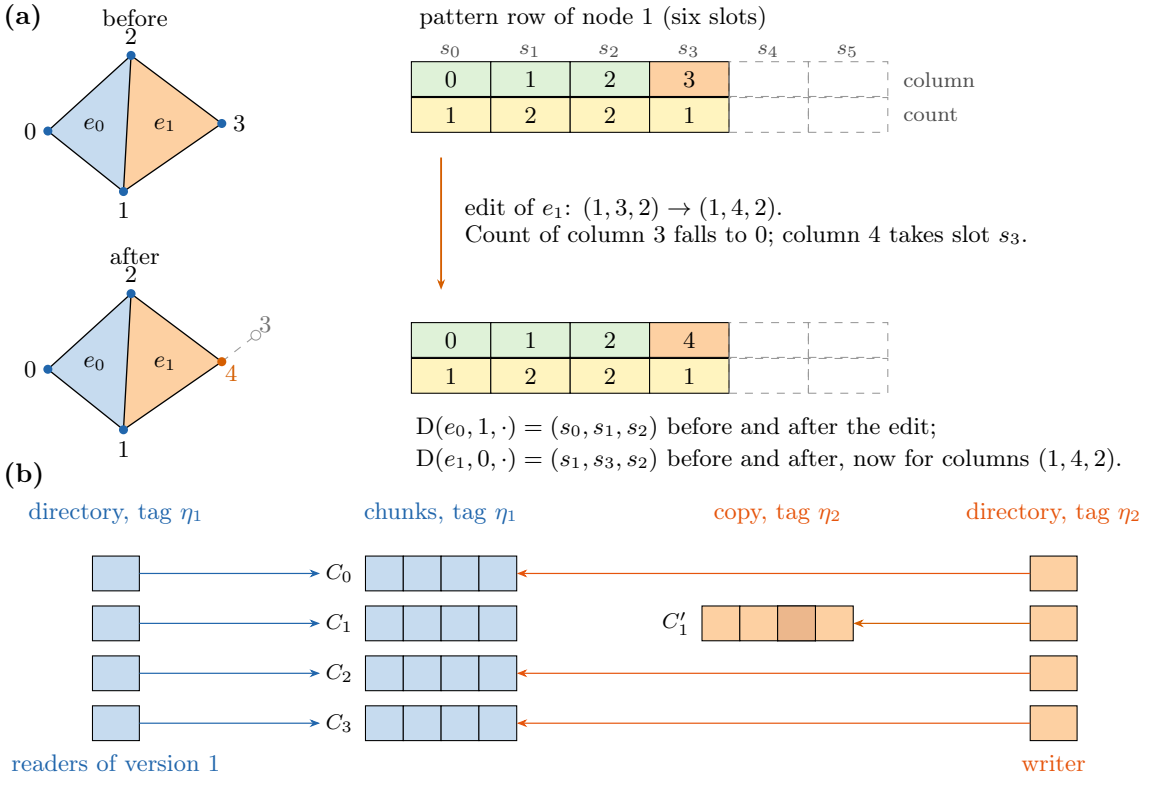


Fig. 2 (a) Row of node 1 of the pattern before and after the edit of e_1 from $(1, 3, 2)$ to $(1, 4, 2)$; free slots dashed, changed slot shaded. (b) Committed directory with its chunks, and working directory after the first write, which copied one chunk.

A color assigned by the repair is at most the number of neighbors of e at that time, as in a greedy coloring. If the initial coloring is greedy, as in Section 6, every color in use was assigned in this way, so the number of colors never exceeds one plus the largest number of neighbors that an element has during the computation. Randomized dynamic graph-coloring algorithms keep the number of colors within the same bound, one plus the largest vertex degree, under arbitrary edge insertions and deletions [12].

3.3 Stable slots and destinations

Each slot of a pattern row holds a column and its count (Figure 2(a)). A column whose count becomes positive takes its old slot if that slot still holds it; otherwise it takes a zero-count slot, or appends a slot if none is available. At count zero, the slot retains its column until another column reuses it. Each row occupies a contiguous segment with spare capacity in a gapped compressed-row array. When full, the row moves to a larger segment at the array end and its start ptr_v changes (relocation). Compaction reclaims abandoned segments while preserving row-relative slots; its cost is proportional to the size of the pattern, and it is not part of an edit.

Theorem 5 (Destination stability)**.** *The destination entries* $\mathrm{D}(e', \cdot, \cdot)$ *of an element* e' *remain unchanged and valid until* e' *is edited: the slot* $\mathrm{D}(e', i, j)$ *of row* $a_{e',i}$ *holds column* $a_{e',j}$ *with a positive count.*

Proof. While e' is live and unedited, every pair $(a_{e',i}, a_{e',j})$ has a count of at least one, because the element itself is counted and the maintained counts equal (3) by the induction in the proof of Theorem 6. A column occupies at most one slot of a row, since the whole row is searched for the column before a zero-count slot is taken, and a slot changes its column only when its count is zero. The slot of the pair therefore does not change. Relocation and compaction move whole rows and change ptr_v, not slots. The row-relative entries (4) of e' therefore stay valid until e' is edited. □

An edit of e writes at most k^2 destinations of e and leaves all other table entries unchanged. Pairs whose nodes keep their local positions need no count or slot update: a node duplication that changes one node of a tetrahedron updates seven of its sixteen pairs.

Theorem 6 (Exactness and cost)**.** *After any sequence of edits, the maintained* T, EP, *the slots of positive count of the pattern with their counts,* D, FA *and the boundary set equal their definitions* (1), (2), (3), (4) *and* (5) *evaluated on the current rows, and the coloring is proper. An edit costs* $\mathcal{O}\big(k^2\ell_{\max} + (k +$

$n_\Sigma)\, k\, d_{\max}\big)$, *where* $d_{\max}$ *is the length of the longest transpose row and* $\ell_{\max}$ *the largest number of slots in a pattern row, before or after the edit, and* n_Σ *the number of sharers whose facet entries are recomputed. In addition, each relocated row costs the copy of at most* $\ell_{\max}$ *entries, and the growth of the arrays costs an amortized constant per added element, node, pattern slot or boundary facet. Apart from the first writes after a commit (Proposition 7) and the growth of the arrays, the cost does not depend on* n_E *or* n_N.

Proof. Exactness follows by one induction over the edits. Assume that before an edit every structure equals its definition and every destination is valid. The decrements of Algorithm 1 then reach the slots of the old pairs; Propositions 1, 2 and 3 and Lemma 4 give the structures after the edit, and the argument of Theorem 5, applied to these counts, keeps every other destination valid. The cost has three terms. The transpose rows of $O \cup N$ with their position shifts, and the color repair, which reads the transpose rows of N, cost $\mathcal{O}(k\, d_{\max})$, which the third term dominates. The $2k^2$ count updates, each with a slot search in a row of at most $\ell_{\max}$ entries, cost $\mathcal{O}(k^2 \ell_{\max})$. The transpose-row merges for the facets of e and of its sharers cost $\mathcal{O}\big((k + n_\Sigma)\, k\, d_{\max}\big)$, and the boundary set changes in amortized constant time. A relocation copies at most $\ell_{\max}$ entries, and an edit relocates at most one row per node of the new row. The arrays grow geometrically, which costs an amortized constant per added entry. □

Besides k, the bound depends on the mesh only through $d_{\max}$, $\ell_{\max}$ and n_Σ. When every facet belongs to at most two elements before and after the edit, $n_\Sigma \le 2k$ and an edit costs $\mathcal{O}\big(k^2(d_{\max} + \ell_{\max})\big)$. A slot is appended only when every slot of the row holds a column of positive count, so the number of slots of a row never exceeds one plus the largest number of nodes adjacent to its node at one time. On a shape-regular family of conforming meshes the number of elements around a node, and with it $d_{\max}$ and $\ell_{\max}$, is bounded in terms of the shape-regularity constant alone. Newest-vertex bisection of a Kuhn mesh (Section 7.1) is conforming and produces tetrahedra of finitely many similarity classes [50, 70, 72], so the bisection-refined meshes satisfy these bounds before they are cut; deletion does not raise the counts, and node duplication does not raise $d_{\max}$. The cut templates of Section 5 do not bound the shape of the children. On cut meshes a cut is rejected when a pattern row could exceed $2^{10} - 1$ neighbors (Section 5.4), which bounds $\ell_{\max}$. On the cut meshes of Section 8.6, the longest transpose row has 80 to 108 elements, and the longest pattern row has 46 to 57 columns of positive count, a lower bound on $\ell_{\max}$.

Algorithm 1 gives the update order. The same update runs unchanged on the persistent arrays of Section 4.

4 Committed versions and rollback

Rejecting a growth step requires the mesh, its derived structures and the nodal fields to be restored exactly. Output may also read one committed version while the next increment is inserted. Versions that share unchanged storage serve both purposes. All structures of Sections 2 and 3 and all registered per-element or per-node attributes (coordinates, refinement tags, history variables) are stored in two-level arrays, in which a directory points to chunks of 2^{10} integers or 2^8 row references. The edits between two consecutive commits or rollbacks form an *epoch*, and the thread that edits is the *writer*. Chunks and directories carry owner tags, the epoch in which they were created or last copied. In epoch η, the writer modifies objects tagged η in place. On the first write to an object that earlier versions share, the writer copies it, together with its directory if that is shared too, and retags the copies. This is path copying [23] in a tree of depth two, as in the page directories of LLAMA [47] (Figure 2(b)).

A *commit* records the directories of all arrays as a version that is never modified afterward, makes it visible to readers in one atomic store, and advances the epoch. The writer never modifies objects reachable from a committed version. Any number of threads can therefore read a committed version without locks while the writer prepares the next one; this is the single-writer protocol of Aspen [22]. A *snapshot* is the reference through which a reader accesses a committed version. *Rollback* restores a committed version's directories as the working state and advances the epoch. Restoring the topology and the attributes registered before the commit takes one reference write per array. Versions that no reader holds are reclaimed by the runtime's garbage collector, which removes the reclamation problem of read–copy update [21].

Proposition 7 (Cost of versioning)**.** *Commit and rollback cost* $\mathcal{O}(n_a)$ *for* n_a *arrays, independent of the mesh. Within an epoch, the first write to a chunk costs* $\mathcal{O}(B)$ *for chunks of* B *entries and the first write to an array of* n_{ent} *entries costs* $\mathcal{O}(n_{\text{ent}}/B)$ *for the directory; later writes in the same epoch cost* $\mathcal{O}(1)$. *A version retains only the chunks and directories copied during its epoch, plus shared data; a compaction gives later versions new pattern arrays.*

The directory copies make the first edit after a commit cost time proportional to the mesh size, which favors one commit per batch. Transient mode, for runs that need neither snapshots nor rollback, keeps

Algorithm 1 Edit of element e from row O to row N.

Require: current structures; rows O (old) and N (new), distinct nodes; $f_{e'}$ is the local index in e' of the facet it shares with e; $W^O_{e,f}$ and $W^N_{e,f}$ are the node sets of facet f in O and in N; Σ counts only elements with as many nodes as e; κ_p is false where O_p or N_p does not exist

$\mathcal{F}_= \leftarrow \{f : W_{e,f}$ has the same node set in O and $N\}$; $\kappa_p \leftarrow (O_p = N_p)$
pending $\leftarrow \emptyset$
for $f \notin \mathcal{F}_=$ **do** ▷ old facets
    remove (e, f) from the boundary set if $\mathrm{FA}(e, f) = -1$
    pending $\leftarrow$ pending $\cup\{(e', f_{e'}) : e' \in \Sigma(W^O_{e,f}) \setminus \{e\}\}$
end for
for i, j with $\neg(\kappa_i \wedge \kappa_j)$ **do** ▷ old pairs, via the destinations
    $\mathsf{P}^{\#}[\mathrm{ptr}_{O_i} + \mathrm{D}(e, i, j)] \leftarrow \mathsf{P}^{\#}[\mathrm{ptr}_{O_i} + \mathrm{D}(e, i, j)] - 1$
end for
remove $(e, \cdot)$ from $\mathsf{T}[v]$ for $v \in O \setminus N$; shift EP behind it
$\mathsf{A}[e] \leftarrow N$
for p with $\neg\kappa_p$ **do**
    insert (e, p) into $\mathsf{T}[N_p]$ (or update p if $N_p \in O$); shift EP; set $\mathrm{EP}(e, p)$
end for
for i, j with $\neg(\kappa_i \wedge \kappa_j)$ **do** ▷ new pairs
    $s \leftarrow$ slot of N_j in row N_i (existing, reused or appended; relocate if full)
    $\mathsf{P}^{\#}[\mathrm{ptr}_{N_i} + s] \leftarrow \mathsf{P}^{\#}[\mathrm{ptr}_{N_i} + s] + 1$; $\mathrm{D}(e, i, j) \leftarrow s$
end for
for $f \notin \mathcal{F}_=$ **do** ▷ new facets
    $\mathrm{FA}(e, f) \leftarrow \min(\Sigma(W^N_{e,f}) \setminus \{e\})$ or -1; update the boundary set
    pending $\leftarrow$ pending $\cup\{(e', f_{e'}) : e' \in \Sigma(W^N_{e,f}) \setminus \{e\}\}$
end for
repair $\mathrm{col}(e)$ by Lemma 4
for $(e', f_{e'}) \in$ pending **do** recompute $\mathrm{FA}(e', f_{e'})$ and its boundary status
end for

a single epoch and copies nothing. In both modes an entry is reached in two dependent loads, directory then chunk; MM3 follows a list-of-lists indirection [8].

5 Local cutting

Cuts replace elements by children; crack increments also duplicate nodes. Both use Algorithm 1 on the structures of Sections 3 and 4.

5.1 Templates and batch equivalence

A cut takes an ordered set M of marked node pairs and places one new node on each pair. The Steiner-point-free templates of [7] replace each element containing a marked edge by children: a bar by two bars, a triangle by two to four triangles, a tetrahedron with $j = 1, \ldots, 6$ marked edges by two to eight tetrahedra. The marked edges determine the template case. Remaining choices (the diagonal of a quadrilateral face, of a pyramid base, of a prism and of the inner octahedron) compare global labels, such as $\max(i_1, i_3) > \max(i_2, i_4)$ for a quadrilateral (i_1, i_2, i_3, i_4) and $\lambda_1 > \lambda_2$ for the labels of the two midpoints of a triangle.

A point element (one node, used for boundary conditions) at an endpoint of a marked pair is copied to the new node of the pair. Each distinct pair receives one new node. As in the batch cutter [7], the new nodes are labeled $n_N, n_N + 1, \ldots$ in the order of first appearance of their pairs in M. The *shell* of a pair (v, w) is the set of elements containing both nodes, $\Sigma(\{v, w\})$ by (5), computed by merging transpose rows of v and w (shaded for triangles in Figure 3(a)).

Theorem 8 (Batch equivalence)**.** *Replacing each element of the union of the shells of M by its template children, the first child keeping the index of the parent and the others appended, and copying each point element at an endpoint of a marked pair to the new node of the pair, produces the multiset of rows of the batch cutter applied to the whole mesh with the same marks, the new nodes carrying the same labels.*

Proof. By the templates above, the children of an element depend only on its row and the midpoint labels of its marked edges. An element outside every shell has no marked edge and is copied by the batch cutter. Both procedures label the new nodes in this order. □

A local cut is a sequence of edits of Algorithm 1, so its cost scales with the number of split elements, and it maintains all structures of Section 2. Face compatibility requires a triangle and its adjacent tetrahedra to use the same label order; with new nodes labeled above every existing node the triangle rule $\lambda_1 > \lambda_2$ and the tetrahedron rule on the same face coincide. The cutter never reuses a node label: a reverted node keeps its label (Section 5.2).

5.2 Reverting a cut

A cut record stores marked pairs, new nodes, parent rows and child indices. Reverting $M_\mathrm{r} \subseteq M$ splits again each parent that contains a pair of M_r, now with marks $M \setminus M_\mathrm{r}$. New children reuse old child indices, with extra children appended and surplus children deleted. Point elements copied to reverted nodes are deleted; those nodes retain their labels but have no incident elements.

Proposition 9 (Local revert)**.** *If no element has been edited since the cut, the nonempty rows after reverting M_r equal, as a multiset, those of the batch cut of the mesh before the cut with the marks $M \setminus M_\mathrm{r}$, under the relabeling that maps the surviving new nodes by rank; the reverted nodes remain without elements.*

Proof. The relabeling keeps every surviving new node above every old node and preserves their relative order, so every comparison of a template has the same outcome. Parents without a pair of M_r keep their children; the others are split by the templates with the remaining marks. □

5.3 Crack insertion and growth

Two level sets define a crack increment: a surface function Φ_s and a region function Φ_r; the crack occupies $\{\Phi_\mathrm{s} = 0,\ \Phi_\mathrm{r} < 0\}$. A node assigned to the crack surface (a new node, an assigned endpoint or a duplicate) has sign 0; any other node has the sign of Φ_s, with $\Phi_\mathrm{s} \geq 0$ counted positive. Insertion proceeds in four stages, which Figure 3 illustrates on triangles.

1. The increment finds the crack edges, which join signs -1 and $+1$ and whose root of Φ_s lies in the crack region; the root parameter is computed from the endpoint with the smaller label.
2. It assigns an endpoint to the surface, without moving it, when it lies within edge parameter $\xi_\mathrm{p} = 0.05$ of a root, unless a tetrahedron of its star (the tetrahedra containing it) would then have four surface nodes. Candidates are processed in ascending label order, and each test includes the earlier assignments.
3. It cuts the remaining crack edges in ascending node-pair order and places each new node at the root with sign 0.
4. It removes the adjacency across crack faces (interior faces with three surface nodes); a surface node whose tetrahedral star then separates into two components is split, and the duplicate takes the positive-side component, except for the front nodes described below.

If a surface node's star remains connected, it is a front node when both signs occur; otherwise its opposite side has already been split off. If the star has two components and one contains both signs, the node is a front node on the domain boundary and is not split; such nodes occur where a front ends on a free surface (Section 8.4). An earlier increment can leave an unsplit node between split ones, and its star may then have more than two components; each component receives one node, and the first negative-side component keeps the original.

A walk starts at the tetrahedra around the previous front and follows the tetrahedra that contain crack edges, so the cost of an increment scales with the advance; the first increment, which has no previous front, scans all edges. An increment is rejected, and the mesh is rolled back to the committed version, when the star of a node has a component without a signed node, components on the same side of the surface, or more than eight components. Nodal fields transfer locally: a new node on edge (v, w) at parameter ξ takes $(1-\xi)u_v + \xi\, u_w$; a duplicate copies the original value. The transfer reproduces exactly any field that is linear on the pre-increment elements.

5.4 Concurrent cutting

Several cracks, or separate regions of one body, can be cut at the same time when their cuts touch disjoint sets of nodes. Groups $M_1, \ldots, M_{n_\mathrm{g}}$ of marked pairs are cut concurrently. The *touched nodes* $\mathcal{V}_i$ of group i are the nodes of the rows of its split elements, its endpoints and its new nodes.

Theorem 10 (Independent groups)**.** *If the sets $\mathcal{V}_i$ are pairwise disjoint, applying the groups on different threads, each with its own range of element indices, produces the rows of the cut of the concatenation $M_1 M_2 \cdots M_{n_\mathrm{g}}$ up to a permutation of the appended children, and every maintained structure equals its definition on these rows.*

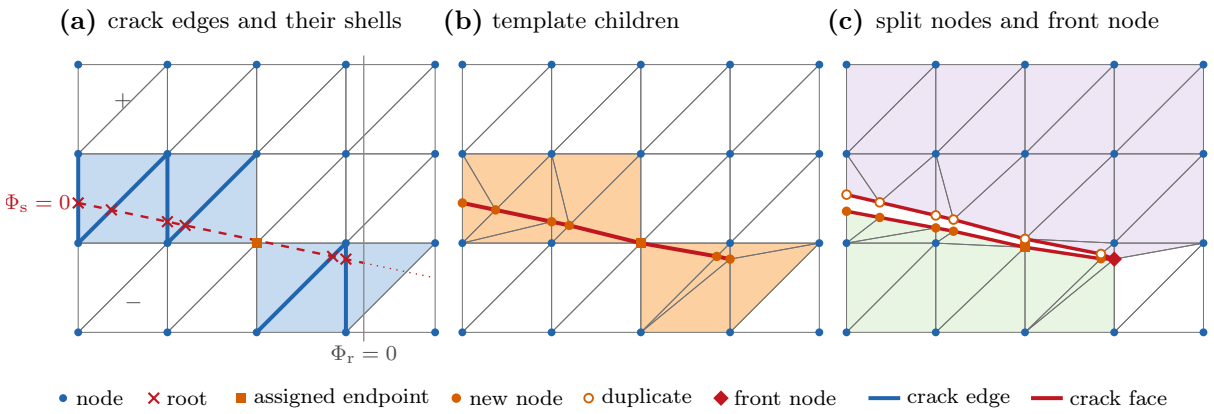


Fig. 3 Crack increment of Section 5.3, illustrated on triangles with the triangle templates of the cutter. (a) Signs of Φ_s, the crack edges that remain after the assignment (endpoints of signs -1 and $+1$, root inside the crack region $\Phi_r < 0$) and their shells; the endpoint within ξ_p of a root is assigned to the surface. (b) The crack edges are cut at their roots and the elements of the shells are replaced by template children; the diagonal of a quadrilateral starts at the new node with the larger label. (c) The adjacency across the crack faces is removed and the surface nodes are split, the duplicates taking the positive-side elements; the front node is not split. Elements whose signed nodes all have the same sign are shaded by side, and the crack is drawn open.

Proof. An edit of Algorithm 1 on element e of group i writes the transpose and pattern rows of the nodes of $O \cup N$, the position entries (e', p) of elements e' at a node of $O \cup N$, the destinations and color of e, and the facet entries of e and of elements sharing a facet of O or N. Each address is indexed by a node of $\mathcal{V}_i$ or by an element whose row lies in $\mathcal{V}_i$: a row by its node, a position entry by the node it refers to, a facet entry by the nodes of its facet, a destination or a color by its element. An element e' edited by another group contains only nodes of $\mathcal{V}_j$, $j \neq i$, so it shares no node with $\mathcal{V}_i$, and elements outside every shell are not edited. The addresses of different groups are therefore distinct, and what an edit of group i reads (transpose rows and the colors of neighbors) is written by no other group. Each group preserves the sequential edit order and disjoint groups commute; labels are assigned in concatenation order before the threads start, which preserves every sequential template decision. □

Groups whose touched sets intersect are merged transitively and applied by one thread, so Theorem 10 applies to any partition of M. Arrays cannot grow while threads write, so before the threads start the cutter reserves the element ranges and the pattern capacity and makes all array directories private to the working version. The first thread that writes to a shared chunk copies it, and the other threads wait for the copy. Pattern segments are allocated one thread at a time. Each thread keeps its own counters and changes to the boundary set, and these are merged when the threads finish. A pattern row holds at most $2^{10} - 1$ neighbors. A cut that could exceed this limit is rejected before any write: the row of an existing node v gains at most the midpoints of the parents containing v ($6|\mathsf{T}[v]|$ for tetrahedra), and the row of a new node holds at most the nodes and midpoints of its shell. The check costs $\mathcal{O}(1)$ while the stored upper bounds on row length and node degree are far from the limit; the concurrent cutter evaluates the bound for every row that a group touches. The result equals the sequential cut in rows, slots, destinations and boundary set, up to the permutation of the appended children; the placement of pattern segments and the order of the boundary list depend on thread timing. A cut that fails for any other reason, such as a crack increment rejected by the checks of Section 5.3, leaves the working version inconsistent, and the mesh is rolled back to a committed version.

6 Implementation

The implementation is in the open-source ManyToMany repository [4], a C# library for .NET 9 that also contains the MM3 library [8], the batch cutter [7] and the verification and measurement programs. Persistent two-level arrays hold the rows, the transpose rows, the position, destination, facet and color tables, the gapped pattern (no row crosses a chunk boundary) and the boundary set. Element creation, deletion and replacement apply Algorithm 1, and versions share immutable element rows. Coordinates, parents and transferred fields are registered as versioned node attributes. A snapshot reads every structure of a committed version.

The tuned recomputation derives the same structures in flat arrays: a counting-sort transpose over element chunks, per-thread marker arrays for the pattern and destinations, parallel loops for the facets and the boundary set, and a parallel greedy coloring of element blocks. Conflicts between blocks are

repaired in rounds. Each color depends only on earlier elements, so the coloring does not depend on thread scheduling. It is the fastest of the recomputations we implemented for each set of structures. The incremental mesh is initialized from it with the sequential greedy coloring in element order, and edits then repair the coloring by Lemma 4. For the solver, a committed pattern is copied into a flat array with the same slots (the *flat copy*), refreshed after a commit by copying only the chunks whose references changed.

The cutter ports the case logic of the batch cutter of [7] and implements the local cut, the revert, the concurrent cut and the crack increments of Section 5. Outside a concurrent cut a single thread edits; algorithms that query while they edit, such as recursive bisection, read the working version, which contains their own writes.

7 Verification

The code verification compares the maintained structures with their definitions in [4, 8], evaluated by an independent program, and the local cutter with the batch cutter of [7]. The stress intensity factors are compared with closed-form solutions for an infinite body on successively refined meshes in Section 8.2 and with the finite element values of Newman and Raju in Section 8.4. Section 7.1 defines the meshes and workloads used here and in Section 9.

7.1 Meshes and workloads

The meshes are Kuhn (Freudenthal) triangulations of the unit cube [27, 44] with n^3 cubes, each split into six tetrahedra. The values $n = 12, 26, 55, 119$ give 1.0×10^4 to 1.0×10^7 elements; verification tests use smaller meshes. Three workloads change the topology. *Crack advance*: a planar crack on $x = \lfloor n/2 \rfloor / n$ grows in $+y$; each of the $n + 1$ steps duplicates the nodes of one front row and reassigns the elements on the positive side. *Recursive bisection*: Maubach's newest-vertex bisection [50, 70, 72] bisects, at each of 40 steps, the elements of level below three whose centroid lies in a ball of radius $2.5/n$ moving along the axis of the box. Bisecting an element first bisects, recursively, the elements around its refinement edge that have a different refinement edge, so queries alternate with edits. *Erosion*: at each of 40 steps the elements whose centroid enters the same ball are deleted. In all three workloads the number of edits per step grows in proportion to n.

7.2 Verification of the maintained structures

The independent program reconstructs all structures from the committed rows by direct enumeration of the node sets and brute-force facet matching. It checks transpose rows, positions, facet adjacency and boundary facets exactly; pattern slots against pair counts, without duplicate columns; destinations against their defining property; and the coloring for properness. Destination blocks of unedited elements are compared between consecutive versions (Theorem 5), and integer element matrices assembled through the destinations must equal direct summation.

Randomized batches duplicate nodes, permute rows, relabel entries to existing nodes (non-manifold configurations), delete and create elements. After each batch the mesh is committed and every structure is compared with the independent program. A random earlier version (up to 64 are retained) is then checked to be unchanged, and every 37th batch the mesh is rolled back to one of the retained versions. Over five seeds and four mesh sizes, 160 000 edits, 837 rollbacks, 31 000 immutability checks and 6.5 million comparisons of destination blocks produce no mismatch. Neither do the three workloads of Section 7.1 on Kuhn meshes with $n = 4, 8, 12$ (up to 111 730 bisections, conformity checked through the boundary facets) nor surface-crack growth with the front ending on a body face (Section 8.4).

The parallel and sequential tuned recomputations agree exactly on all structures except the coloring, on Kuhn meshes of 1.6×10^5 and 1.3×10^6 elements in natural and random order and on mixed meshes. The parallel coloring is identical across runs with 2 and 16 threads and uses 39 or 40 colors; the greedy coloring in element order uses 30 or 31 (39 in random order).

7.3 Verification of the cutter

Each cutting round is compared with the batch cutter of [7] (Fortran templates translated to C#) applied to the whole mesh. The meshes are mixed meshes of tetrahedra, boundary triangles, bars and point elements, with random edges marked in a sub-box (0.5 to 97 %) or edges crossing a sphere; zero to six marked edges per tetrahedron occur. Every round agrees on node count, new-node parents and the row multiset of each element type (Theorem 8); all structures match the independent program and the mesh remains conforming. Partial and complete reverts reproduce the batch cut with the remaining marks and the mesh before the cut (Proposition 9).

The crack-edge walk agrees with a full edge scan: rows, node count and coordinates match a global evaluation for an inclined penny crack, an edge crack and a curved crack on meshes of $n = 8$, 13 and 21. A field linear on the initial elements is transferred through ten increments with an error of at most 2.0×10^{-15}.

Concurrent cuts are compared with the sequential cut of the concatenated groups on a copy of the mesh, for groups in grid cells separated by four element sizes (independent), by one (partly merged) and with a group spanning the box (all merged). All rounds of five runs with 2, 4, 8 and 16 threads agree, up to 3.2 million cut edges per run, in node count and parents, rows and ancestors, every structure, conformity, edit counts, slot reuse, relocation and earlier-version immutability. They also agree after reverting single groups and after repeating a cut with another thread count following a rollback.

In the cutter verification no edge is marked whose shell has a node with more than 200 neighbors, since repeated cuts of one region raise node degrees without bound. On meshes with $n = 4$, 6 and 9, repeated cuts of opposite edges in the star of one node grow the pattern row of that node from 15 to 51, 195 and 771 columns. The fourth cut, with a bound of 3 075 columns, is rejected by both the sequential and the concurrent cutter before any write. The working version stays identical to the committed one, and the structures after a later cut match the independent program.

8 Examples

We extract stress intensity factors by a domain interaction integral (Section 8.1) and compare them with closed-form solutions for penny-shaped cracks (Section 8.2). Paris-law growth of an elliptical crack (Section 8.3) and of surface cracks (Section 8.4) uses one cutter increment per step (Section 5) and follows the loop of Figure 1. A mixed-mode corner crack, grown toward the boundary by the Erdogan–Sih criterion, demonstrates cutting and versioning along a curved path (Section 8.5). Section 8.6 gives the element shapes and row lengths of the cut meshes.

In all examples the body Ω is linear elastic and contains the crack surface $\mathcal{C}$. Let $\mathcal{C}_h$ be the union of the crack faces of the current tetrahedral mesh, which approximates $\mathcal{C}$, and let V_h be the space of vector fields that are linear on each tetrahedron, continuous across every face shared by two tetrahedra, and satisfy the six point constraints given below. Because the crack nodes are duplicated, the fields of V_h can jump across $\mathcal{C}_h$, and $V_h \subset H^1(\Omega \setminus \mathcal{C}_h)^3$. The displacement $\boldsymbol{u}_h \in V_h$ satisfies

$$\int_\Omega \operatorname{Sym}\nabla \boldsymbol{v} : \mathbb{C} : \operatorname{Sym}\nabla \boldsymbol{u}_h \,\mathrm{d}V = \int_{\partial\Omega_\mathrm{t}} \sigma\,(\boldsymbol{n}_\mathrm{t} \cdot \boldsymbol{v})\,\mathrm{d}S \qquad \text{for all } \boldsymbol{v} \in V_h, \tag{9}$$

with $\mathbb{C}$ the isotropic elasticity tensor with Young's modulus E and Poisson's ratio ν, $\partial\Omega_\mathrm{t}$ the two loaded faces, $\boldsymbol{n}_\mathrm{t}$ their outward normal and σ the applied tension. The other faces and the crack faces are free of traction. The six constraints remove the rigid-body motions and are statically determinate, so the self-equilibrated loads produce no reactions.

The stiffness matrix of (9) is assembled through the destination table (4) in 3×3 blocks per pattern slot. Its rows are gathered in parallel: row v sums the blocks of the pairs of $\mathsf{T}[v]$ in element order, so that the matrix equals that of a sequential element loop bit for bit. The conjugate-gradient solver with block-Jacobi preconditioning reads the flat copy and uses the residual tolerance $10^{-10}\,\|\boldsymbol{f}^\mathrm{ext}\|$ ($\boldsymbol{f}^\mathrm{ext}$ the nodal loads) unless stated otherwise. Mesh changes between solves use Algorithm 1. The penny and elliptical cracks use the cube $[-1, 1]^3$ with $E = 1$ and $\nu = 0.3$, loaded by a uniaxial tension $\sigma = 1$ along z on the faces $z = \pm 1$. Six point constraints at three corners remove the rigid-body motions (the surface crack of Section 8.4 uses a plate). The Kuhn mesh of spacing $h_0 = 0.1$ is refined by recursive bisection [50] until no element is longer than $1.5\,h(\boldsymbol{x})$, with $h = h_\mathrm{min} + 0.35 \max(0, d - r_q)$, r_q the tube radius of Section 8.1 and d the distance to the region the crack fronts occupy. Refined tetrahedra have a radius ratio $3r_\mathrm{in}/r_\mathrm{circ}$ (r_in and r_circ the inradius and circumradius) of at least 0.641; Section 8.6 gives the ratios after fitting and cutting. Recursive bisection does not refine template children, so the band around the crack path is refined before the first increment. The band must therefore contain the fronts of the whole sequence and is chosen before the run; the edits themselves are not planned, since each increment follows from the stress intensity factors of the previous solution. Each example grows one crack with the sequential cutter; concurrent cutting is measured in Section 9.7. The times of this section are single measurements of one run, on the workstation of Section 9.1 (AMD Ryzen 9 9950X, 16 cores) or on one node of the Deucalion supercomputer (A64FX, 48 cores, or two EPYC 7742, 128 cores).

On a mesh without a crack the computed displacement reproduces the exact solution, which is linear, to within 7×10^{-11}.

8.1 Stress intensity factors on the maintained structures

A planar crack lies in the plane $\boldsymbol{n}\cdot\boldsymbol{x}=\zeta$ with front $\rho=R(\psi)$, a star-shaped periodic cubic spline in plane polar coordinates. Before insertion, the mesh is fitted to the front in a band of $3h_{\min}$ around the plane. A node within $0.2h_{\min}$ of the surface $\rho=R(\psi)$ moves onto it radially when no neighbor has moved and no tetrahedron of its star changes orientation or loses more than 90 % of its volume. Edges crossing the surface more than a tenth of their length from either endpoint are cut at the crossing. The crack is then inserted as in Section 5, with the region $\rho<R(\psi)+h_{\min}/4$.

At a front point parametrized by ψ, the stress intensity factors (the factors, for short) follow from the interaction integral [75] in the domain form of [52, 67]. The virtual extension field is $\boldsymbol{V}=q\,\boldsymbol{N}$. Here $\boldsymbol{N}$ is the outward in-plane normal at the nearest front point, and q is the product of a tube factor $1-r/r_q$ (r the distance to the front; the tube is the set of points within r_q of the front) and a hat, interpolated linearly in each tetrahedron. The hat is piecewise linear in the curve parameter ψ, equal to 1 at the front point and 0 at the parameter distance $\omega=\max(r_q/\bar{R},\,2\pi/64)$ from it, with $\bar{R}$ the mean radius of the front. The interaction integral is

$$I\int_\Gamma q\,\mathrm{d}s=\int_\Omega\Big(\sigma_{ij}H_{ik}+\sigma^{\mathrm{aux}}_{ij}u_{i,k}-\sigma_{pl}\varepsilon^{\mathrm{aux}}_{pl}\delta_{jk}\Big)V_{k,j}\,\mathrm{d}V+\int_\Omega\Big(\sigma^{\mathrm{aux}}_{ij,j}u_{i,k}+\sigma_{ij}\big(H_{ik,j}-H_{ij,k}\big)\Big)V_k\,\mathrm{d}V. \quad (10)$$

Here Γ is the front, u_i and σ_{ij} the components of $\boldsymbol{u}_h$ and of its stress $\mathbb{C}:\mathrm{Sym}\nabla\boldsymbol{u}_h$ (the scalar σ remains the applied tension), I the interaction integral for one auxiliary mode and $\boldsymbol{\varepsilon}^{\mathrm{aux}}=\mathrm{Sym}\boldsymbol{H}$ the auxiliary strain. The auxiliary gradient $\boldsymbol{H}$ is the plane-strain or antiplane near-tip gradient of unit K_{I}, K_{II} or K_{III} in the frame (normal, binormal, tangent) of the nearest front point, rotated to global axes, and $\boldsymbol{\sigma}^{\mathrm{aux}}=\mathbb{C}:\mathrm{Sym}\boldsymbol{H}$ is the two-dimensional asymptotic stress, traction-free on the crack plane [29, 30, 73]. The integrals over planar crack faces therefore vanish. Because the front-curvature terms are omitted, $\boldsymbol{H}$ is not compatible; the last integral of (10) accounts for the resulting non-equilibrium and incompatibility, using central differences of $\boldsymbol{H}$. Quadrature uses four points per tetrahedron. The factors are $K_{\mathrm{I}}=E'I_{\mathrm{I}}/2$, $K_{\mathrm{II}}=E'I_{\mathrm{II}}/2$, $K_{\mathrm{III}}=\mu I_{\mathrm{III}}$ with $E'=E/(1-\nu^2)$ and μ the shear modulus. Each factor is therefore an average along the front with the weight of the hat. Where a factor varies along the front, this average differs from its value at the front point, and Sections 8.2 and 8.3 compare it with the reference averaged in the same way. The J-integral of the same domain, which needs no auxiliary field, differs from the energy release rate computed from the factors by at most 4.0, 2.4, 1.0 and 0.55 % for normal loading on the penny-crack meshes with $h_{\min}=h_0/4$, $h_0/8$, $h_0/16$ and $h_0/32$.

Two changes adapt the interaction integral to a front that ends on a free surface. First, where the tube meets a free part $\partial\Omega_{\mathrm{f}}$ of the boundary, with outward normal $\boldsymbol{n}_{\mathrm{f}}$, the auxiliary stress is not traction-free, and the divergence theorem gives the surface term

$$\Psi_{\mathrm{f}}=\int_{\partial\Omega_{\mathrm{f}}}\big(\sigma^{\mathrm{aux}}_{ij}n_{\mathrm{f},j}u_{i,k}-\sigma_{pl}\varepsilon^{\mathrm{aux}}_{pl}n_{\mathrm{f},k}\big)V_k\,\mathrm{d}S, \quad (11)$$

because the traction of the actual solution vanishes there. The factors are those of (10) with Ψ_{f} subtracted; Ψ_{f} is evaluated by a three-point rule on the boundary triangles of the tube, and the crack faces and the loaded faces are excluded from $\partial\Omega_{\mathrm{f}}$. Where the front meets the free surface the singularity is not of inverse-square-root order [10]; the values reported at the surface point are averages over a hat of finite width, not pointwise values. Second, the tube factor is 1 for $r\le\gamma r_q$ (the *plateau*), with $\gamma=0.5$, and falls linearly to zero at r_q, replacing the linear factor $1-r/r_q$. The plateau excludes near-front elements, where the stress is least accurate, from the tube-factor gradient terms. The surface and corner cracks (Sections 8.4 and 8.5) use both changes.

8.2 Penny-shaped cracks

A crack of radius $a=0.2$ through the origin has the normal $(\sin\beta,0,\cos\beta)$. For $\beta=0$ the reference is $K_{\mathrm{I}}=K_0=2\sigma\sqrt{a/\pi}$ [68]; for $\beta=30^\circ$ it is the solution of Kassir and Sih [38] for the normal stress $\sigma\cos^2\beta$ and the shear $\tau=\sigma\sin\beta\cos\beta$ on the crack plane, $K_{\mathrm{II}}=4\tau\sqrt{a/\pi}\cos\theta/(2-\nu)$ and $K_{\mathrm{III}}=4(1-\nu)\tau\sqrt{a/\pi}\sin\theta/(2-\nu)$, with θ measured from the direction of the shear. These solutions are for an infinite body; the faces of the cube are $5a$ from the center of the crack. The crack plane for $\beta=0$ contains lattice nodes of the Kuhn mesh, and the crack is assembled from faces of the mesh. For $\beta=30^\circ$ the increment cuts 8 170 crack edges on the mesh of Figure 4 ($h_{\min}=h_0/8$), which shows the crack faces opening and sliding.

Figure 5 gives the displacement jump across the crack faces, compared with the closed form $8(1-\nu^2)\sigma\cos^2\beta\sqrt{a^2-r^2}/(\pi E)$ (r the distance from the center of the crack) for the opening and $16(1-$

Table 1 Penny-shaped cracks: largest difference of the stress intensity factors from the solutions averaged over the hat of the interaction integral, over sixteen front points, in % of K_0 (linear tube factor, $r_q = 0.075$, $\omega = 21.5°$); tetrahedra and unknowns of the meshes with $\beta = 30°$.

$h_{\min}/a$	tetrahedra	unknowns	$\beta = 0$: K_{I}	$\beta = 30°$: K_{I}	K_{II}	K_{III}
0.125	341 620	195 084	3.15	2.21	0.97	1.23
0.0625	1 291 000	724 779	0.73	0.94	0.61	0.51
0.03125	6 779 092	3 753 192	0.34	0.55	0.24	0.14
0.015625	47 393 057	25 971 639	0.29	0.12	0.11	0.02

Table 2 Penny-shaped cracks at $h_{\min} = h_0/8$ ($h_{\min}/a = 1/16$): largest difference of the stress intensity factors from the solutions averaged over the hat, over sixteen front points, in % of K_0, for the linear tube factor and for the plateau $\gamma = 0.5$ and three tube radii; K_{II} and K_{III} for $\beta = 30°$.

r_q	tube factor	$\beta = 0$: K_{I}	$\beta = 30°$: K_{I}	K_{II}	K_{III}
0.05	linear	2.11	1.75	1.18	0.74
	plateau	0.48	0.56	0.37	0.29
0.075	linear	0.73	0.94	0.61	0.51
	plateau	0.35	0.48	0.33	0.19
0.11	linear	0.34	0.58	0.33	0.15
	plateau	0.41	0.55	0.27	0.16

$\nu^2)\tau\sqrt{a^2 - r^2}/(\pi(2-\nu)E)$ for the sliding. The computed opening and sliding are 1.3 and 1.5 % below the closed forms at the center and differ from them by at most 2.7 and 3.9 % of the central value for $r < 0.9a$. Figure 6 gives the three factors along the front against the pointwise solution. Along the circular front K_{II} and K_{III} vary as $\cos\theta$ and $\sin\theta$, so that their averages over the hat are the pointwise values times $S = [\sin(\omega/2)/(\omega/2)]^2$; K_{I} is constant along the front. Table 1 gives the largest difference from these averaged solutions over sixteen front points, signed, with θ measured from the direction of the shear as in Figure 6, for the linear tube factor with $r_q = 0.075$ ($\omega = 21.5°$, $S = 0.988$).

The errors fall from 3.15 to 0.29 % ($\beta = 0$) and from 2.21, 0.97 and 1.23 to 0.12, 0.11 and 0.02 % (the three factors, $\beta = 30°$) as $h_{\min}$ halves three times; the ratios of successive entries are irregular, and no order of convergence is inferred from them. Against the pointwise solutions, K_{II} and K_{III} differ by 1.56, 1.20, 0.83 and 0.51 % and by 1.65, 0.92, 0.56 and 0.42 % on the four meshes; the averaging alone accounts for 0.59 and 0.42 % of K_0. At $h_{\min} = h_0/16$ the errors are 0.34 % ($\beta = 0$) and 0.19, 0.14 and 0.05 % ($\beta = 30°$) with $r_q = 0.11$ ($\omega = 31.5°$), and 0.94 % and 1.05, 0.56 and 0.38 % with $r_q = 0.05$ ($\omega = 14.3°$), on 13.2 and 3.6 million tetrahedra. To separate the hat from the tube, the solutions with $r_q = 0.075$ are also evaluated with half-widths $\omega = 5.6$, 11.3, 21.5, 32.2 and 43.0°. At $h_{\min} = h_0/8$ the error of K_{II} is 1.13, 1.31, 0.61, 0.57 and 0.51 % against the averaged solution and 1.17, 1.47, 1.20, 1.90 and 2.86 % against the pointwise one; at $h_{\min} = h_0/16$ it is 0.82, 0.49, 0.24, 0.13 and 0.09 % and 0.86, 0.66, 0.83, 1.46 and 2.44 %. K_{III} behaves alike, and K_{I}, which the averaging does not change, differs by 2.10 and 2.11 % with the two narrowest hats at $h_{\min} = h_0/8$ and by 0.88 to 0.95 % with the others.

Table 2 compares the linear tube factor with the plateau of Section 8.1 at $h_{\min} = h_0/8$. Against the averaged solutions, the plateau lowers the largest of the four errors from 2.11, 0.94 and 0.58 % to 0.56, 0.48 and 0.55 % for $r_q = 0.05$, 0.075 and 0.11; at $h_{\min} = h_0/16$ and $r_q = 0.075$ it gives 0.40 % ($\beta = 0$) and 0.38, 0.27 and 0.10 % ($\beta = 30°$). For a crack of half the radius in the same cube ($a = 0.1$, faces at $10a$, $r_q = 0.0375$, linear factor), the errors at $h_{\min}/a = 1/8$ are 3.77 % ($\beta = 0$) and 2.67, 1.19 and 1.39 % ($\beta = 30°$). At $h_{\min}/a = 1/16$ they are 1.49 % and 1.39, 0.73 and 0.71 %. These errors exceed those of the crack with $a = 0.2$ at the same $h_{\min}/a$ (Table 1), although the faces are twice as far away; the mesh away from the refined band is also twice as coarse relative to a ($h_0 = a$).

A second finite element code checks the assembly through the maintained destinations: SimPlas [5] (4-node tetrahedra, linear analysis) solves the same inclined penny crack on the Kuhn mesh with $n = 12$ ($h_0 = 1/6$, $h_{\min} = h_0/4$, 150 996 tetrahedra, $\beta = 30°$), using the same loads and constraints. The SimPlas displacements differ from those computed here by 2.6×10^{-9} of the largest displacement, a difference at the precision of the SimPlas output.

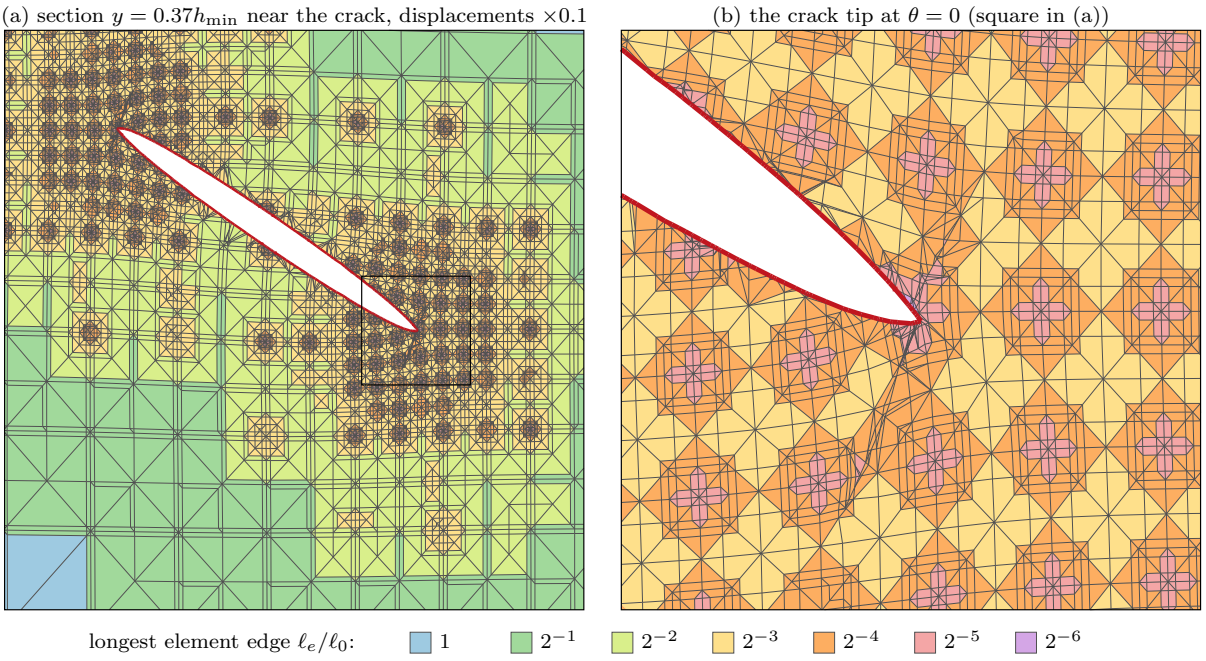


Fig. 4 Inclined penny-shaped crack ($\beta = 30°$, $h_{\min} = h_0/8$, 1 291 000 tetrahedra): section $y = 0.37h_{\min}$ of the deformed mesh, displacements scaled by 0.1, section polygons colored by the longest edge ℓ_e of their tetrahedron relative to that of the initial lattice, ℓ_0, crack faces red; (a) the neighborhood of the crack, (b) the front at $\theta = 0$.

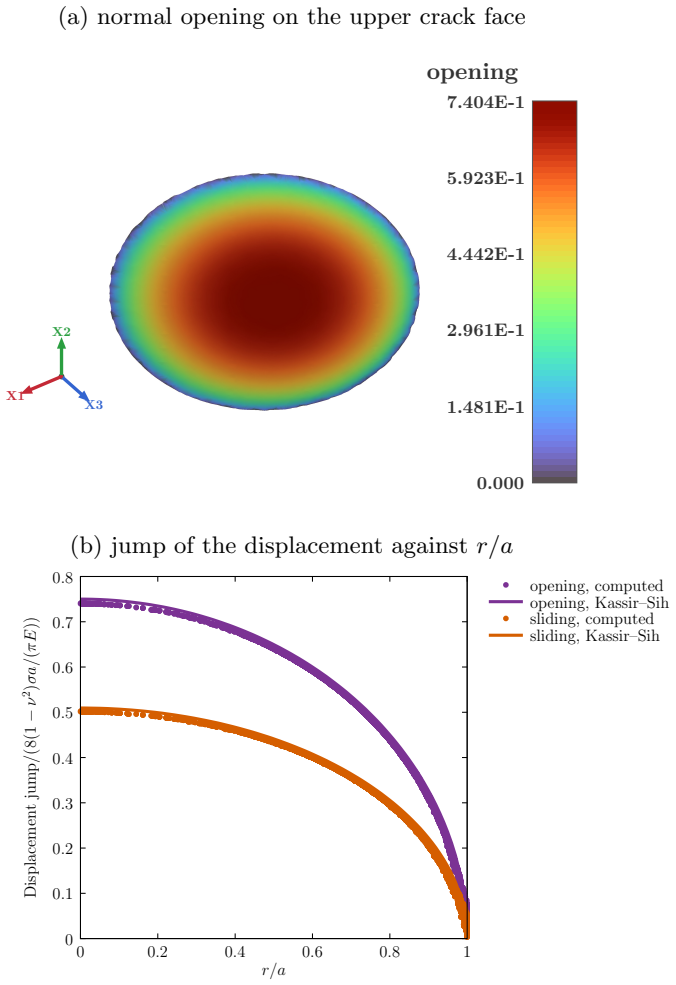


Fig. 5 Inclined penny-shaped crack ($\beta = 30°$, $h_{\min} = h_0/8$): jump of the displacement across the crack faces divided by $8(1-\nu^2)\sigma a/(\pi E)$. (a) Normal component on the deformed upper face (displacements scaled by 0.1; the face is drawn up to the front circle), drawn by the IST Vector Postprocessor [3]; (b) opening and sliding at every pair of crack nodes against the solution of Kassir and Sih.

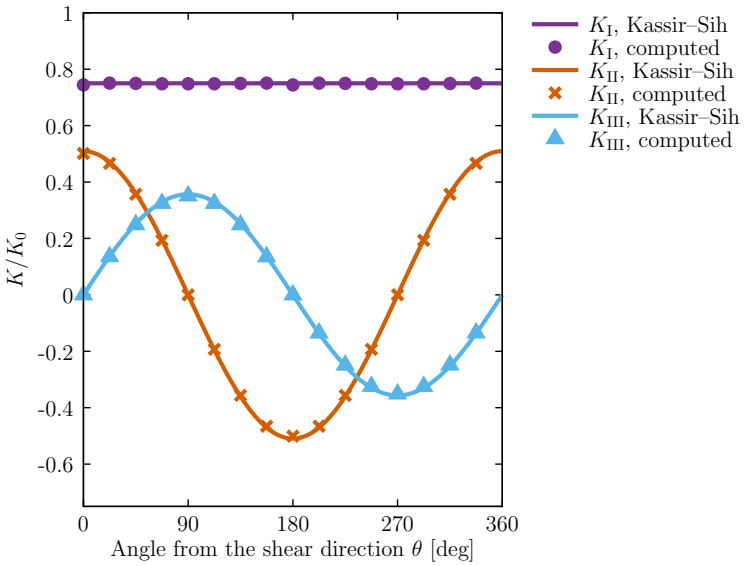


Fig. 6 Inclined penny-shaped crack ($\beta = 30°$, $h_{\min} = h_0/16$): stress intensity factors along the front against the solution of Kassir and Sih.

8.3 Fatigue growth of an elliptical crack

The elliptical crack tests Paris-law growth with rejected steps. An elliptical crack with semi-axes $a = 0.25$ (along x) and $b = 0.15$ lies in the plane $z = 0.37h_{\min}$, offset from the lattice planes, so every increment cuts edges; the mesh size in the band the fronts cross is $h_{\min} = h_0/8$. At the first front, K_{I} is compared with the solution of Irwin [36] for an infinite body, $K_{\mathrm{I}} = \sigma\sqrt{\pi b}\,(\sin^2\varphi + (b/a)^2\cos^2\varphi)^{1/4}/\mathrm{E}(\kappa)$, with φ the parametric angle and $\mathrm{E}(\kappa)$ the complete elliptic integral of the second kind of modulus $\kappa = \sqrt{1 - b^2/a^2}$ (the faces of the cube are $4a$ from the center). On 3 396 115 tetrahedra the largest difference along the first front is 1.80 % from the solution of Irwin averaged over the hat ($\omega = 18.0°$) and 1.97 % from its pointwise values (Figure 7(b)).

The front advances by a Paris law [62]: the 32 knots of the front move along the normal by $\Delta a_j = \Delta a\,(K_j/K_{\max})^m$, $m = 4$, with K_j the value of K_{I} at knot j, so that Δa is the advance of the point of largest K_{I}. Each step fits the mesh to the new front (Section 8.1), inserts the crack increment by a walk from the previous front (Section 5.3), commits, assembles, solves from the previous displacement field and evaluates K_{I} at 32 knots. Stable node indices preserve existing nodal displacements; new nodes interpolate from their parents. If the relative factors $K_j/\overline{K}$ ($\overline{K}$ the mean of the K_j) change by more than 0.04 over a step, a rollback restores the pre-step version and the step is repeated with $\Delta a/2$, down to $\Delta a = h_{\min}$, at which a step is accepted. After an accepted step, Δa is multiplied by 1.5, up to 0.025. The advances use the factors of the current front, and the change of the relative factors over a step serves as a heuristic indicator for the step size; it is not an estimate of the error of the front update. Only the ratios of the advances enter; the number of cycles is not computed. The tube radius is $r_q = 0.06$.

With $\Delta a = 0.025 = 2h_{\min}$ the eight steps after the first front are accepted at the first attempt (relative changes of 0.019 to 0.038). The crack tends to the circular shape expected under uniform tension: the aspect ratio b/a grows from 0.60 to 0.93 and the ratio of the largest to the smallest K_{I} along the front falls from 1.29 to 1.04 (Figure 7(a,b)). A step cuts 3 865 to 7 199 crack edges and 5 143 to 9 299 fitting edges, and the front has 522 to 920 nodes, each within $h_{\min}/2$ of the curve. Each cut edge edits its entire shell, so these steps are larger than the node-duplication steps of Section 7.1.

Figure 7(c) gives the cost of each step on the Ryzen 9 9950X. For the eight steps after the first front, fitting and crack increment take 0.30 to 0.55 s, commit 4 to 14 µs (one call per step) and the refresh of the flat copy 0.5 to 14 ms. The tuned parallel recomputation of all structures from the rows (Section 6) would take 0.68 to 0.87 s. Assembly takes 0.15 to 0.31 s on 16 threads. The solution takes 51 to 75 s (1 810 to 2 215 iterations at relative residual 10^{-10}; starting from the previous solution saves few iterations), and factor evaluation at the 32 knots takes 6.2 to 7.4 s. Fitting, crack increment and commit take 0.52 to 0.66 % of a step, and the mesh grows from 3.40 to 3.96 million tetrahedra.

On an A64FX node of Deucalion (48 cores), with the initial and largest advance $\Delta a = 0.05$, three of ten attempts change the relative factors by 0.068, 0.048 and 0.044; rollback restores the mesh and the front data in 6.5, 0.048 and 0.027 ms. The steps repeated with half the advance are accepted, and the crack reaches $b/a = 0.92$ over seven accepted fronts, including the first front. On that node the fitting

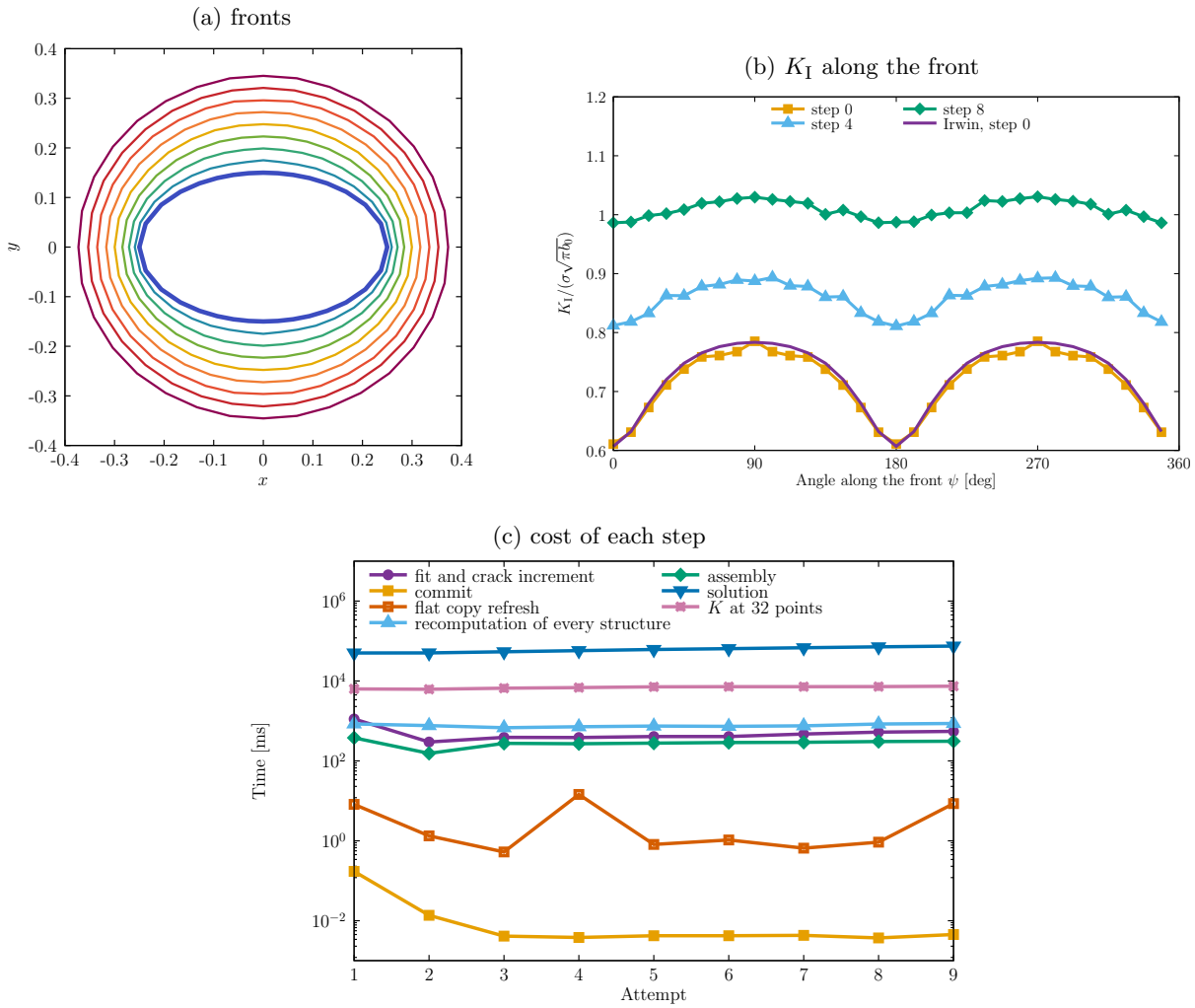


Fig. 7 Paris-law growth of an elliptical crack ($h_\mathrm{min} = h_0/8$): (a) fronts of the accepted steps (first front thick, in blue); (b) K_I along the front at steps 0, 4 and 8, and the solution of Irwin for the first front; (c) cost of each attempt (every attempt is accepted).

and crack increment of a rejected attempt take 3.8 to 5.7 s; without stored versions, undoing it would also need a saved copy of the mesh and fields and a recomputation of all structures, which takes 2.0 to 2.6 s.

8.4 Surface crack of Newman and Raju

The factors at a front that ends on a free surface are compared with the finite element values of Newman and Raju. A semi-elliptical crack of depth a and half-length c lies at the face $y = -t$ of the plate $[-1, 1] \times [-t, 0] \times [-L, L]$, of thickness t, half-width $b = 1$ and half-length L. The symbol b is the plate half-width here and a semi-axis in Section 8.3. We use L for Newman and Raju's h because h denotes element size. The plate is loaded by the tension $\sigma = 1$ on $z = \pm L$ with six point constraints at three corners; the computed displacement of the uncracked plate reproduces the exact, linear solution to within 5×10^{-11}. The crack lies in the plane $z = 0.37 h_\mathrm{min}$, offset from the lattice planes so that every increment cuts crack edges, and is centered on the face. The closed front curve $\rho = R(\psi)$ uses the representation of Section 8.1. Its arc $0 \le \psi \le \pi$ lies in the body and ends on the face at $\psi = 0$ and $\psi = \pi$ (Figure 8). The interaction-integral hat is normalized over the front arc inside the body.

The surface term (11) and the plateau of Section 8.1 are used with $r_q \approx a/3$, a the crack depth, unless stated otherwise. For an embedded ellipse with semi-axes 0.2 and 0.12 in the cube ($h_\mathrm{min} = h_0/8$, $r_q = 0.04$, 2 529 426 tetrahedra) the plateau lowers the largest difference from the solution of Irwin averaged over the hat from 3.70 to 1.86 % (from 3.81 to 1.98 % from its pointwise values).

Newman and Raju [56] fitted $K_\mathrm{I} = \sigma\sqrt{\pi a/Q}\, F_\mathrm{s}$ to the finite element results of Raju and Newman [64], with $Q = 1 + 1.464(a/c)^{1.65}$ for $a/c \le 1$ and F_s a function of a/c, a/t, c/b and the parametric angle φ of the front point ($x = c\cos\varphi$, $y + t = a\sin\varphi$). Table I of NASA TP-1578 [55] lists $F = K_\mathrm{I}/(\sigma\sqrt{\pi a/Q})$ for $c/b = c/L = 0.2$ and $\nu = 0.3$ at $2\varphi/\pi = 0, 1/8, \ldots, 1$. Two of its cases serve as references: $a/c = 0.6$ and $a/t = 0.2$ ($a = 0.12$, $c = 0.2$, $t = 0.6$), and $a/c = 1$ and $a/t = 0.4$ ($a = c = 0.2$, $t = 0.5$), both

Table 3 Surface cracks of Newman and Raju: difference of F from Table I of NASA TP-1578 [55] over $2\varphi/\pi = 0, 1/8, \ldots, 1$ (% of the tabulated value; largest in magnitude, and signed differences, computed minus tabulated, at the deepest and surface points); tetrahedra and unknowns of the meshes ($h_0 = 0.1$, $r_q = 0.04$ and 0.065, $\gamma = 0.5$).

case	$h_{\min}/a$	tetrahedra	unknowns	largest	deepest	surface
$a/c = 0.6$, $a/t = 0.2$	0.139	850 360	466 215	4.11	−3.19	+1.07
	0.104	1 923 599	1 060 590	4.61	−2.40	+1.33
	0.083	3 098 864	1 698 609	4.74	−4.74	+1.55
	0.069	4 082 741	2 217 990	3.66	−3.66	+1.30
	0.052	10 859 707	5 935 686	3.56	−3.56	+1.47
	0.035	26 321 087	14 225 667	3.14	−2.89	+1.60
$a/c = 1$, $a/t = 0.4$	0.125	495 962	279 204	2.27	+2.27	+0.60
	0.083	1 052 955	580 269	1.75	+1.75	−0.38
	0.062	2 505 757	1 384 551	1.34	+1.34	−0.43
	0.042	5 682 059	3 087 378	1.59	+1.59	−0.26
	0.031	15 932 759	8 716 593	1.53	+1.53	−0.13

with $b = L = 1$. Table 3 gives the largest difference of F from the tabulated values over the nine angles, and Figure 9 shows F along the front. These are pointwise values; averaged over the hat, the equation of Newman and Raju differs from its pointwise values by at most 0.77 and 0.30 % between the surface points and by 0.07 % or less at the deepest point.

For $a/c = 0.6$ the largest difference is 3.66, 3.56 and 3.14 % on the three finest meshes, after a rise from 4.11 to 4.74 % over the three coarsest. The equation of Newman and Raju, fitted to the same finite element results, differs from the table by up to 4.0 %. On the finest mesh the computed F lies 2.4 to 3.1 % below the tabulated values from $2\varphi/\pi = 3/8$ on, and 1.7 to 2.3 % below the equation, which is within 1.2 % of the tabulated values there. At the surface point F is 1.6 % or less above the table, where the equation is 4.0 % above it. An energy balance checks the factors. The front is advanced from $a = 0.12$ to 0.15 (c from 0.2 to 0.219). The decrease of the potential energy under dead load, $\Delta(\boldsymbol{f}^{\text{ext}} \cdot \boldsymbol{u})/2$ ($\boldsymbol{u}$ the displacements), then agrees with the swept-area integral of the energy release rate G, evaluated with G averaged between the two fronts, to within 1.6 % for $h_{\min} = h_0/8$ and 0.6 % for $h_{\min} = h_0/12$ (where the advance reaches $a = 0.149$).

For $a/c = 1$ the largest difference is 1.3 to 1.8 % on the four finest meshes ($h_{\min}/a \le 0.083$), at the deepest point on every mesh. On the finest mesh the computed F at the deepest point agrees with the equation of Newman and Raju to within 0.18 %, and the table lies 1.7 % below the equation. At the surface point the difference is 0.6 % or less.

The embedded ellipse with semi-axes 0.2 and 0.12, computed with the same tube and plateau and refined from $h_{\min} = h_0/8$ (1.86 %) to $h_0/12$, is within 1.58 % of the solution of Irwin averaged over the hat (1.69 % of its pointwise values) on 5 132 891 tetrahedra. On the mesh with $h_{\min} = h_0/3$ for $a = 0.15$ and $c = 0.3$ in the plate $t = 1$, $L = 2$ (266 592 tetrahedra, 1 092 crack-face nodes), the displacements of SimPlas [5] for the same loads and constraints differ from those computed here by 1.3×10^{-9} of the largest displacement. In a growth sequence with $h_{\min} = h_0/4$ and $\Delta a = 0.03$ (533 496 tetrahedra), six steps are accepted and four attempts, whose relative factors change by 0.084 to 0.162, are rolled back. Every committed version agrees with the independent program, and after each rollback every structure of the working version is identical to that of the restored version.

The growth sequence of Figure 10 starts from a crack with $a = 0.15$ and $c = 0.3$ ($a/c = 0.5$, $a/t = 0.15$) in the plate $t = 1$, $L = 2$ ($h_{\min} = h_0/8$, $r_q = 0.05$, 2 284 992 tetrahedra). The surface term (11) changes the factor at the surface point: with the linear tube factor, K_{I} at the surface point is 8.7 % below the equation of Newman and Raju without the term and 3.1 % with it. The largest difference at other points is 4.5 % in both cases. The factors K_{II} and K_{III}, whose exact values vanish by symmetry, have computed magnitudes below 0.7 % of K_{I}. The J-integral of the same domain, evaluated without a free-surface term, differs from the energy release rate computed from the factors by at most 1.6 % at the first front with $h_{\min} = h_0/6$ and the plateau.

The crack grows by the Paris law with $m = 3$ and the plateau $\gamma = 0.5$. The knots of the half front move along the normal by $\Delta a_j = \Delta a\,(K_j/K_{\max})^m$ with $\Delta a = 0.02$. Factors are evaluated at knots inside the body ($0 \le \psi \le \pi$) and mirrored about $x = 0$. The rollback rule of Section 8.3 with the tolerance 0.06 triggers no rollback: the relative factors change by 0.019 to 0.042 per step. Twelve steps after the first front take a/c from 0.50 to 0.788, a from 0.15 to 0.383 and c from 0.30 to 0.486, inside the range of the equation of Newman and Raju (Figure 10(a,b)). The reference evolution starts from the same (a, c): each

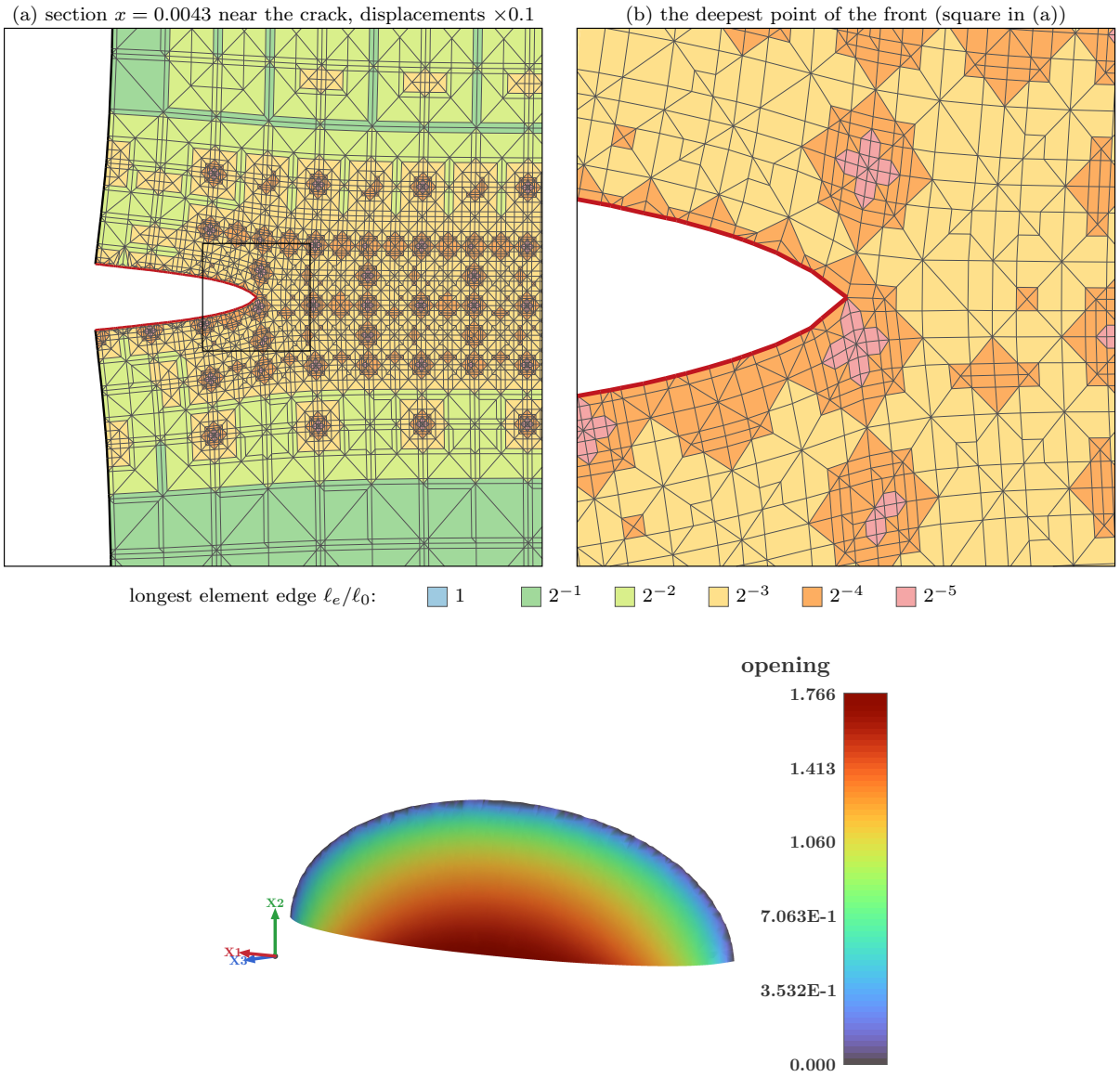


Fig. 8 Surface crack ($a = 0.15$, $c = 0.3$, $t = 1$, $h_{\min} = h_0/6$, 1 093 547 tetrahedra), displacements scaled by 0.1: section $x = 0.0043$ through the deepest point of the front, polygons colored by ℓ_e/ℓ_0 as in Figure 4, crack faces red and plate faces black ((a) the neighborhood of the crack, (b) the front); below, the normal jump of the displacement across the crack on the upper face divided by $8(1-\nu^2)\sigma a/(\pi E)$ (the face is drawn up to the front curve), drawn by the IST Vector Postprocessor [3].

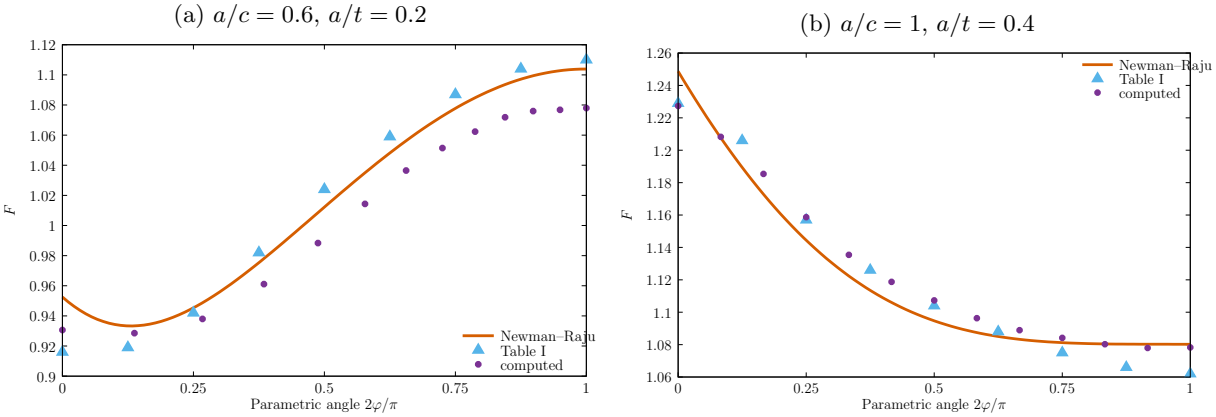


Fig. 9 Surface cracks of Newman and Raju: $F = K_{\mathrm{I}}/(\sigma\sqrt{\pi a/Q})$ along the front against the parametric angle for the finest mesh of Table 3, Table I of NASA TP-1578 [55], and the equation of Newman and Raju [56].

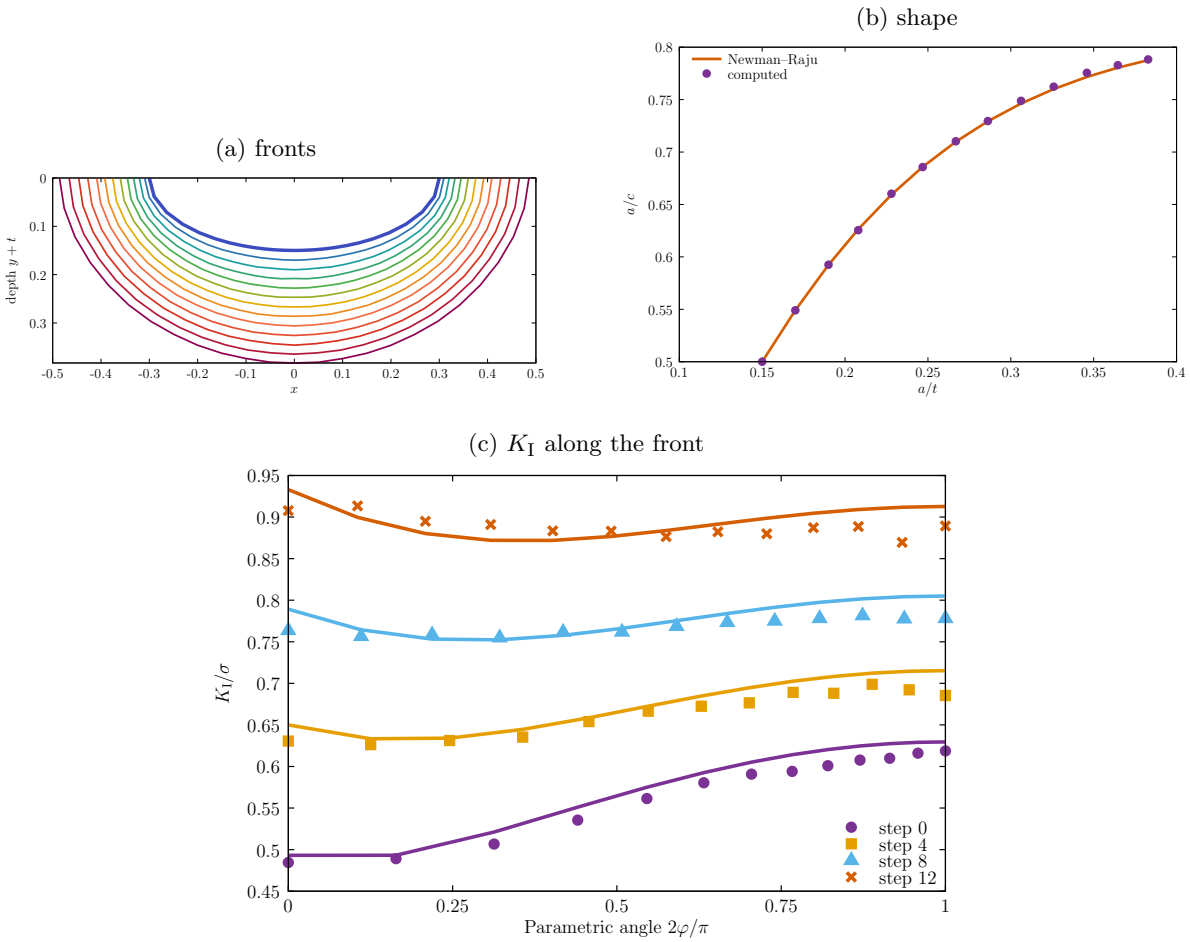


Fig. 10 Paris-law growth of a surface crack ($m = 3$, $h_{\min} = h_0/8$): (a) fronts of the accepted steps (first front thick, in blue); (b) a/c against a/t of the computed fronts and of the Newman–Raju-driven evolution of (a, c); (c) K_{I} along the front at steps 0, 4, 8 and 12 and the equation of Newman and Raju.

step increases a by the computed deepest-point advance and c by that advance times $(K_{\mathrm{C}}/K_{\mathrm{A}})^m$, with K_{A} and K_{C} the values of the equation at the deepest and surface points for the current (a, c). The half-length of the reference evolution is within 0.51 % of the computed half-length at every step. The largest difference of the factors from the equation along each front is 3.05 to 5.09 % (Figure 10(c)); the law uses only the ratios $K_j/K_{\max}$, so a common error of the factors does not enter. The comparison tests the ratio of the advances of the two points; it is not a separate solution of the growth problem.

On an A64FX node of Deucalion (48 cores), fitting takes 0.67 to 1.23 s, the crack increment 0.72 to 1.49 s (1 800 to 3 881 crack edges), commit 9 to 50 µs and the refresh of the flat copy of the pattern 0.8 to 10 ms. The tuned parallel recomputation of all structures from the rows would take 1.34 to 1.98 s. The assembly takes 0.25 to 0.31 s, the solution 87 to 142 s (2 110 to 2 747 iterations) and the factors at 25 points 19 to 25 s. Fitting, crack increment and commit take 1.3 to 1.6 % of a step, and the mesh grows from 2.28 to 2.73 million tetrahedra. With the coarser $h_{\min} = h_0/6$ (same crack and plate; 1.09 to 1.36 million tetrahedra), the attempt at $a = 0.188$ changes the relative factors by 0.073, and a rollback restores the mesh and the front data in 6.6 ms. The step is accepted when repeated at the minimum advance $h_{\min}$, and thirteen steps after the first front reach $c = 0.498$, with a half-length within 0.54 % of the evolution driven by the equation.

8.5 Corner-crack growth under mixed-mode loading

A quarter-circular crack of radius $a_0 = 0.1$ lies at the corner of the block $[0, 1]^2 \times [-1, 1]$, in the plane whose normal is inclined by 40° from the load axis, which gives mixed-mode loading. The loading and the constraints are those of Section 8.4, with $E = 1$ and $\nu = 0.3$; the faces $x = 0$ and $y = 0$ are free. The crack grows toward the faces $x = 1$ and $y = 1$. The front is a polyline of 64 knots, each with a frame $(\boldsymbol{N}, \boldsymbol{B}, \boldsymbol{T})$: $\boldsymbol{N}$ is the direction of the last advance, $\boldsymbol{B}$ the normal of the crack surface and $\boldsymbol{T} = \boldsymbol{N} \times \boldsymbol{B}$. At each knot, the factors come from the interaction integral of Section 8.1 evaluated in this frame, with the surface term of (11) on the four lateral faces. The tube radius is the larger of 0.15 times the mean front radius and six local element sizes, with the plateau $\gamma = 0.5$. The integral over the crack faces vanishes

only for planar faces; it is omitted here although the faces are curved, and the penny-crack verification covers only planar faces. Unlike in the examples above, the mesh is not fitted to the front: the increment cuts the edges that cross the level set, and the crack surface, made of element faces, lies within a fraction of an element of the level set.

The Erdogan–Sih criterion [25] gives the kink angle ϑ, measured from $\boldsymbol{N}$ toward $\boldsymbol{B}$, and the advance Δa_j at knot j:

$$\vartheta_j = 2\arctan\frac{K_{\mathrm{I}} - \sqrt{K_{\mathrm{I}}^2 + 8K_{\mathrm{II}}^2}}{4K_{\mathrm{II}}}, \qquad \Delta a_j = \Delta a\Big(\frac{K_{\mathrm{e},j}}{K_{\mathrm{e,max}}}\Big)^m, \quad K_{\mathrm{e}}^2 = K_{\mathrm{I}}^2 + K_{\mathrm{II}}^2, \tag{12}$$

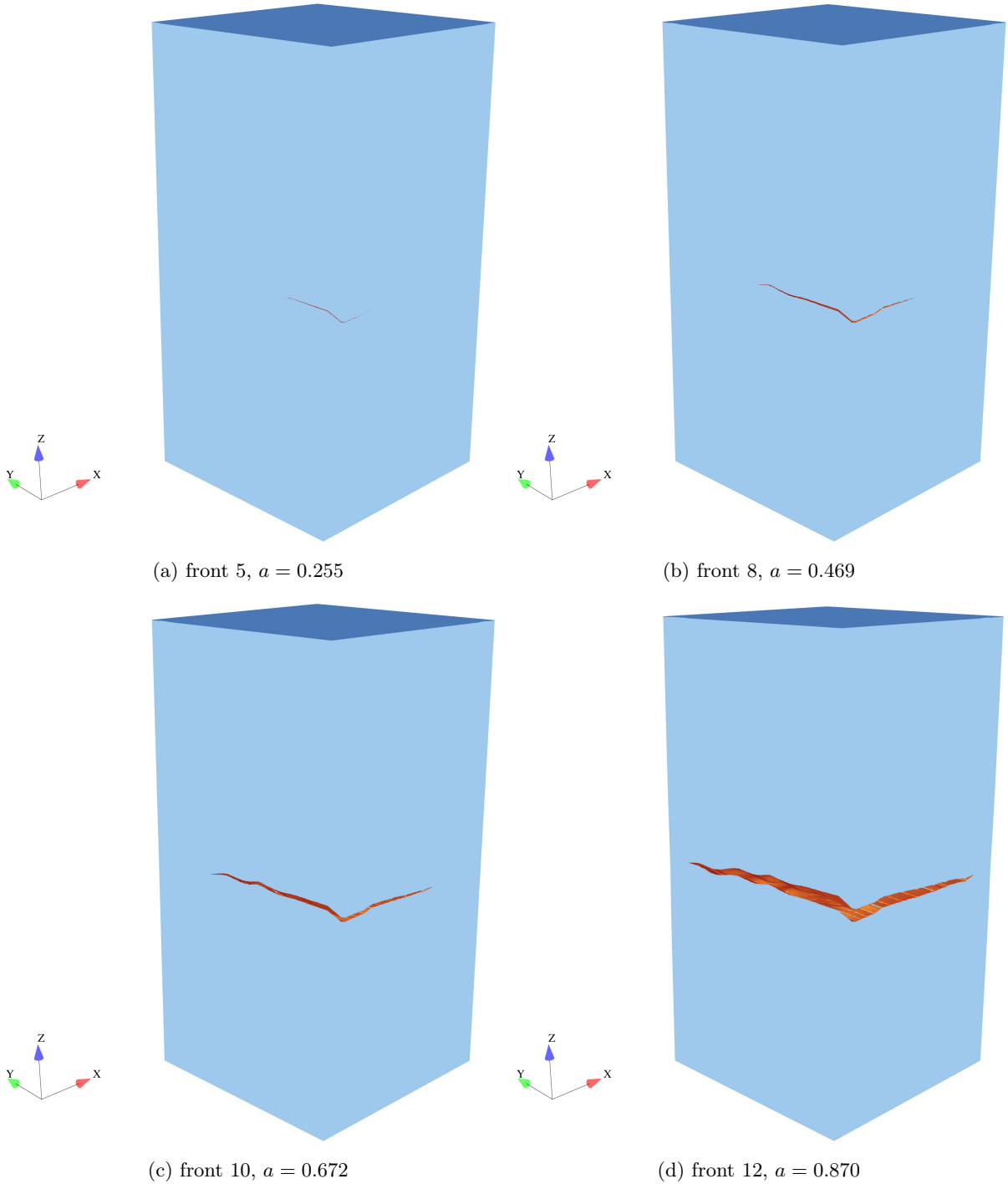
with $m = 3$ and Δa equal to 0.25 times the mean front radius. K_{III} does not enter the criterion. Knots within one hat width of a free face use values from one hat width inside. Kink angles are limited to 45° and constrained by a minimum curvature radius of twelve element sizes. Advances and the new polyline are smoothed along the front. The new knots lie at $\boldsymbol{P}_j + \Delta a_j(\cos\vartheta_j \boldsymbol{N}_j + \sin\vartheta_j \boldsymbol{B}_j)$ with $\boldsymbol{P}_j$ the position of knot j and the end knots on the faces. The crack surface is a graph $\zeta = w(s_1, s_2)$ over the initial plane, built from cubic Hermite strips between successive fronts (tangent to the previous advance at the old front and to the new direction at the new one). Behind each earlier front, the level set of the increment retains the values with which the nodes there were cut. The increment is that of Section 5.3, including the splitting of a node whose star has more than two components into one node per component.

The Kuhn mesh of spacing 0.05 is refined by recursive bisection to the size $0.043(\rho + 0.05)$ in the slab $-0.3 \le z \le 0.2$, $\rho \le 1.05$, with ρ the distance from the corner edge, and graded outside the slab. The mesh has 6 387 904 tetrahedra, with radius ratio of at least 0.641. Applied to the penny crack of Section 8.2 ($\beta = 30°$, $h_{\min}/a = 1/16$, $r_q = 0.1$, plateau, 1 769 612 tetrahedra), the same front and interaction integral differ from the solution of Kassir and Sih averaged over their hat ($\omega = 28.6°$) by 1.2, 1.7 and 3.2 % of K_0 in the three factors. There is no reference value for the corner crack, and its factors are verified only through the penny crack above. The example demonstrates the cutter and the committed versions under a mechanically driven, approximate growth procedure toward two free faces; it does not validate the crack path.

Thirteen fronts, with the mean front radius growing from $a = 0.100$ to 0.870, bring the crack within 0.115 of the boundary (Figures 11 and 12, Table 4). The crack leaves its inclined plane and turns toward the plane normal to the load: the fronts remain between $z = -0.081$ and $z = 0.010$, whereas the initial plane would reach $z = -0.73$ at $x = 0.87$. All insertions succeed on the first attempt, every committed version agrees with the independent program, and the upper crack face remains connected. The first increment kinks by up to 20°, and the 45° limit is never reached (largest kink 21.3°, Table 4). The factor K_{II} reaches 0.16 on the first front, against K_{I} of 0.24 to 0.27, and stays below 0.4 afterward; K_{I} grows from 0.24–0.27 to 3.0–4.0. The magnitude $|K_{\mathrm{III}}|$ is below 0.33 up to front 8 and reaches 1.3 at the end of the front on the face $x = 0$, where the two-dimensional criterion, which neglects K_{III}, is least justified. The tube radius grows from 0.036 to 0.228; at fronts 11 and 12 (0.213 and 0.228) it exceeds the distances from the front to the faces $x = 1$ (0.19 and 0.115) and $y = 1$ (0.21 and 0.13), which therefore cut the tube. The run on one node of Deucalion (two EPYC 7742, 128 cores) solves meshes of 6.42 to 6.52 million tetrahedra with 3.54 to 3.66 million unknowns and takes 2.5 h in total. The solutions take 2.2 h, 454 to 985 s each (3 732 to 4 710 conjugate-gradient iterations at 10^{-9}), the factors at 64 knots 3 to 8 s, the insertions 0.4 to 6.2 s, and mesh generation 65 s.

8.6 Element shapes and row lengths of the cut meshes

Table 5 gives the radius ratio $3r_{\mathrm{in}}/r_{\mathrm{circ}}$ of the tetrahedra of four stored meshes. The penny-crack mesh of Figure 4 is fitted and cut; the corner-crack meshes of fronts 0, 6 and 12 (Table 4) are cut without fitting. Before fitting and cutting, the smallest ratio is 0.641. After cutting, the median is 0.70 on all four meshes. On the penny-crack mesh, 0.59 % of the tetrahedra have a ratio below 0.1 and the smallest ratio is 1.1×10^{-3}. On the corner-crack meshes the share below 0.1 grows from 0.035 to 0.20 % over the twelve increments, and the smallest ratio falls from 2.3×10^{-3} to 4.8×10^{-4}. The shape controls are the assignment of near-root endpoints to the surface (Section 5.3) and, in the fitting, the volume condition and the rule that cuts an edge only where the crossing is more than a tenth of its length from both endpoints (Section 8.1); no shape improvement follows a cut. The table also gives the longest transpose row, $d_{\max}$, and the number of columns of positive count in the longest pattern row (the node and the nodes adjacent to it). This number is a lower bound on $\ell_{\max}$ of Theorem 6, which also counts the zero-count slots. The longest transpose row has 80 to 108 elements and the longest pattern row 46 to 57 such columns, against the limit of $2^{10} - 1$ neighbors (Section 5.4).



(a) front 5, $a = 0.255$

(b) front 8, $a = 0.469$

(c) front 10, $a = 0.672$

(d) front 12, $a = 0.870$

Fig. 11 Corner crack (inclination 40°, 6.5×10^6 tetrahedra): deformed block (displacements scaled by 0.005) at fronts 5, 8, 10 and 12 of the Erdogan–Sih growth; a is the mean front radius.

9 Performance

Timings use the workloads of Section 7.1 unless stated otherwise.

9.1 Measurement conditions

Measurements were taken on an AMD Ryzen 9 9950X (16 cores, 32 threads, two 32 MB L3 caches, 91 GB of memory) under Linux with .NET 9.0.2. The runtime settings are those of the measurement programs released with MM3 [8]: one hardware thread per core, the server collector and no concurrent collection.

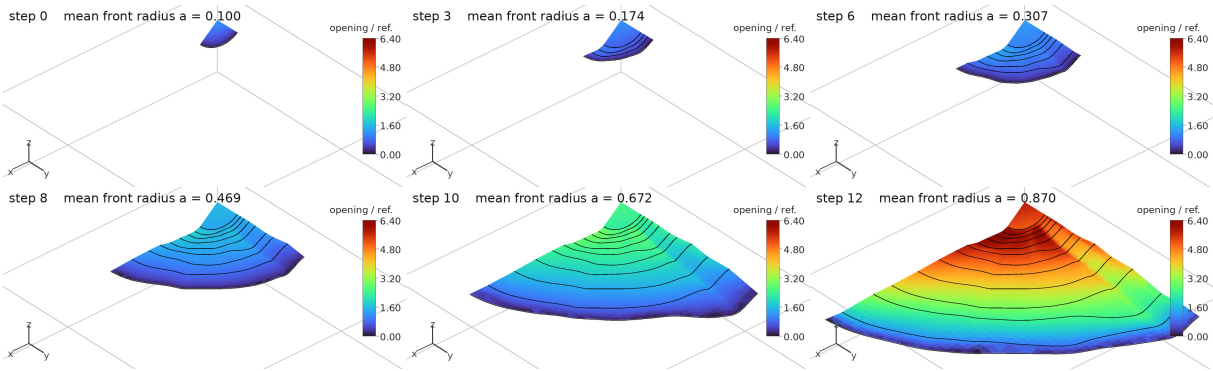


Fig. 12 Corner crack (inclination 40°, 6.5×10^6 tetrahedra), fronts 0, 3, 6, 8, 10 and 12 (labeled as steps in the panels) of the Erdogan–Sih growth: upper face of the crack colored by the normal opening divided by $8(1-\nu^2)\sigma a/(\pi E)$, a the mean front radius of the front shown (0.100, 0.174, 0.307, 0.469, 0.672, 0.870); the earlier fronts in black. The color is clipped at the prescribed front of each step; the mesh is not fitted to the front, and the cut follows element faces within a fraction of an element of it.

Table 4 Corner crack: tetrahedra and unknowns of the solved step, conjugate-gradient iterations and solution time on 128 cores, range of K_I along the front, largest $|K_\mathrm{II}|$ and largest kink angle ϑ of the increment that follows (64 knots, after smoothing and angle limiting).

front	a	tetrahedra	unknowns	iterations	solution [s]	K_I	max $\lvert K_\mathrm{II}\rvert$	ϑ [°]
0	0.100	6 415 498	3 542 478	3 851	705	0.24–0.27	0.16	19.9
1	0.121	6 421 526	3 549 321	3 732	462	0.28–0.38	0.12	21.3
2	0.143	6 427 791	3 556 380	3 796	585	0.33–0.47	0.09	3.3
3	0.174	6 434 249	3 563 724	3 868	590	0.38–0.52	0.13	6.0
4	0.211	6 441 709	3 572 052	3 917	454	0.40–0.61	0.08	11.8
5	0.255	6 450 819	3 582 462	4 026	619	0.46–0.65	0.08	13.1
6	0.307	6 462 104	3 595 110	4 101	524	0.67–0.75	0.28	7.0
7	0.378	6 474 753	3 609 372	4 242	587	0.77–0.89	0.17	17.2
8	0.469	6 487 889	3 624 075	4 327	545	1.01–1.17	0.11	11.0
9	0.581	6 504 706	3 642 924	4 444	724	1.29–1.93	0.39	13.8
10	0.672	6 515 136	3 654 846	4 494	719	1.47–1.87	0.27	15.8
11	0.803	6 520 373	3 660 780	4 638	985	2.24–2.91	0.33	7.6
12	0.870	6 523 413	3 664 224	4 710	573	3.03–4.04	0.20	6.7

Table 5 Radius ratio $3r_\mathrm{in}/r_\mathrm{circ}$ of the tetrahedra of four cut meshes: smallest value and share of the tetrahedra below 0.1 and below 0.2 [%]; longest transpose row d_max (elements at a node) and longest pattern row (columns of positive count). The penny-crack mesh ($\beta = 30°$, $h_\mathrm{min} = h_0/8$) is fitted and cut; the corner-crack meshes are cut without fitting.

mesh	tetrahedra	smallest	below 0.1	below 0.2	d_max	columns
Penny-shaped crack	1 291 000	1.1×10^{-3}	0.59	1.59	108	57
Corner crack, front 0	6 415 498	2.3×10^{-3}	0.035	0.13	80	46
Corner crack, front 6	6 462 104	6.7×10^{-4}	0.11	0.38	86	46
Corner crack, front 12	6 523 413	4.8×10^{-4}	0.20	0.67	88	47

The baselines are the original MM3 and the tuned recomputation. The *MM3 set* is the transpose, element positions, node positions, clique destinations and the node–node product, computed by MM3 with its own parallel routines. The tuned recomputation covers the MM3 set and all structures (adding facet adjacency, boundary set and coloring), every phase on 16 threads. The tuned recomputation of all structures is also timed sequentially, with the greedy coloring in element order. Each timing is the minimum of three samples after full garbage collections; the tables report the median of these minima over the sampled steps. At 10^7 elements the sampled recomputations lie between 0.96 and 1.24 times their median for MM3, 0.78 and 1.05 times for the MM3 set and 0.93 and 1.08 times for all structures, over the three workloads. Times include collector pauses; at most 10 % of the timed regions of a run contain a collection. Cuts allocate inside the timed region and are timed without collector pauses. Appendix A gives the pauses of the cuts and a check of the MM3 baseline on this machine.

Table 6 Time per step [ms] of the three workloads: incremental maintenance (pers.: commit after every step; trans.: transient mode, no versions) against the MM3 recomputation and the tuned 16-thread recomputations of the MM3 set and of all structures; n_E is the initial number of elements, $\tilde{n}_{\text{ed}}$ the median number of element edits per step, and MM3/pers. the ratio of the MM3 recomputation time to the persistent time.

			incremental		recomputation			
Workload	n_E	$\tilde{n}_{\text{ed}}$	pers.	trans.	MM3	MM3 set	all	MM3/pers.
Crack	1.04×10^4	108	0.176	0.096	1.23	0.262	1.64	7
	1.05×10^5	234	0.385	0.223	8.18	1.88	9.59	21
	9.98×10^5	495	0.970	0.569	79.0	20.3	93.2	81
	1.01×10^7	1 071	3.01	1.37	849	300	705	282
Bisection	1.04×10^4	384	1.25	–	2.07	0.221	3.98	2
	1.05×10^5	1 008	2.11	–	7.88	2.38	17.2	4
	9.98×10^5	2 436	4.80	–	85.6	20.9	104	18
	1.01×10^7	3 970	7.69	–	869	301	699	113
Erosion	1.04×10^4	24	0.079	–	1.17	0.237	1.52	15
	1.05×10^5	50	0.104	–	7.22	1.89	9.35	70
	9.98×10^5	121	0.187	–	77.5	20.7	100	414
	1.01×10^7	246	0.735	–	840	296	688	1 143

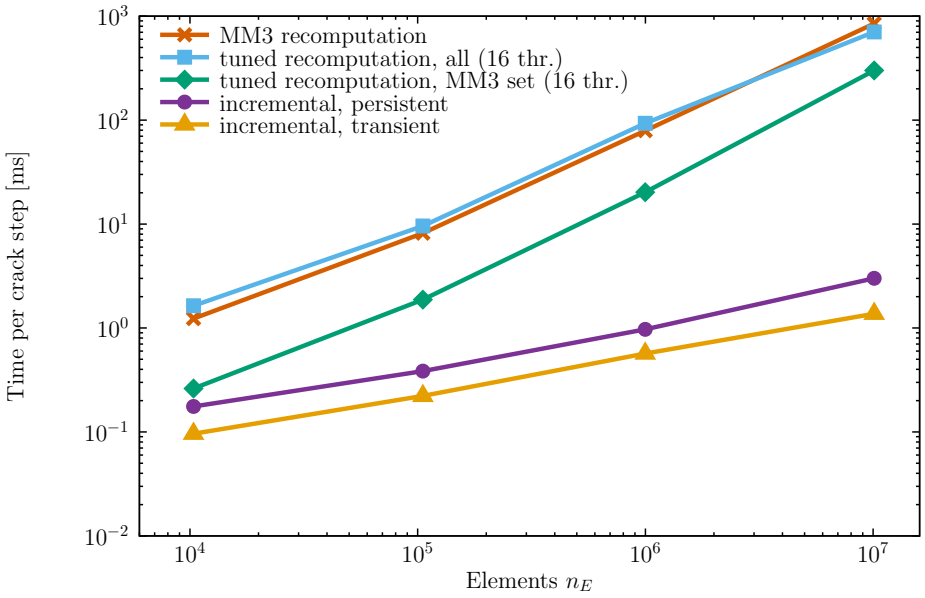


Fig. 13 Crack advance: time per step against the number of elements. The recomputation times grow with the mesh. The transient cost grows with the front, whose length is proportional to the number n of cubes per side; the persistent cost adds the directory copies.

9.2 Edit sequences

Incremental cost follows the number of edits (Table 6, Figure 13). From 10^4 to 10^7 elements, crack edits per step rise from 108 to 1 071 and the persistent step from 0.18 to 3.01 ms, while the MM3 recomputation rises from 1.23 to 849 ms and the tuned MM3-set recomputation from 0.26 to 300 ms. At 10^7 elements the connectivity update of a crack step is 282 and 100 times cheaper than these two baselines. Bisection, at 3 970 edits per step including recursive neighbor bisections, is 113 and 39 times cheaper, and erosion, at 246 deletions per step, 1 143 and 403 times cheaper. For crack advance the cost per edit, the time per step over the mean number of edits per step, is 1.7 to 2.8 µs in persistent mode and 1.0 to 1.3 µs in transient mode. It grows slowly with the mesh because of larger arrays and, in persistent mode, the directory copies of Proposition 7. At 10^7 elements the first edit after a commit costs 1.8 ms against 16 µs in transient mode (Section 9.5), which explains the persistent-mode overhead of 1.6 ms per step (3.01 against 1.37 ms). Computing the structures from scratch is cheaper only for bisection on the smallest mesh, where a step edits 384 elements, 3.7 % of the initial mesh: 0.22 ms for the tuned MM3-set recomputation against 1.25 ms.

Table 7 Share of the time inside Algorithm 1 per phase, crack advance and bisection on 10^6 elements (instrumented run, timer overhead included, workload logic excluded; last row: time per edit).

Phase	crack, transient	crack, persistent	bisection, persistent
Facets (old, new, sharers)	46.6 %	32.9 %	62.5 %
Pattern counts and destinations	26.5 %	37.7 %	13.6 %
Transpose and positions	13.1 %	19.3 %	12.9 %
Coloring	4.8 %	2.5 %	5.7 %
Row, checks	9.0 %	7.6 %	5.3 %
Time per edit [µs]	0.98	2.18	1.96

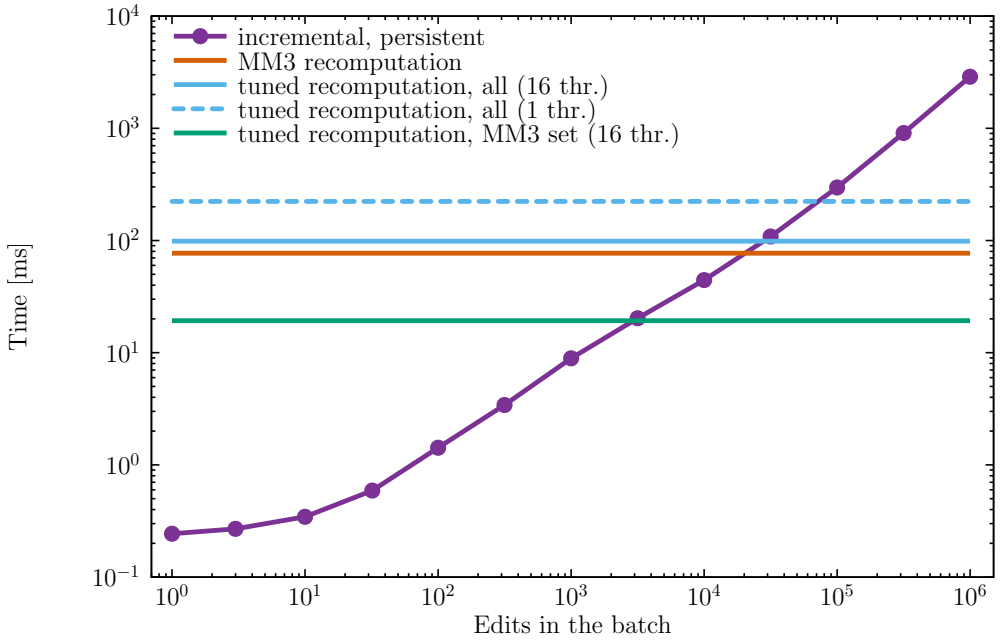


Fig. 14 A single batch of random node duplications on 9.98×10^5 elements, committed at the end, against the recomputations of the same mesh.

Table 7 breaks down the time inside Algorithm 1. Facets cost most in transient crack advance and in persistent bisection, since each edit recomputes the facets of the old and new rows and of their sharers by three-way merges of transpose rows. Persistence raises the share of pattern, destination and transpose updates, whose chunks are copied on first write in each epoch.

9.3 Batch size

Figure 14 measures one batch of random node duplications on 9.98×10^5 elements, each in a different, uniformly sampled element; each point is the minimum of three samples from the same version, restored by rollback. A batch costs about 0.2 ms for the commit and the directory copies, plus 10.8 µs per edit at 316 edits, 6.4 µs at 3 162 and 2.9 to 3.0 µs from 10^5 edits on. It stays cheaper than the tuned MM3-set recomputation up to 0.30 % of the elements, than the MM3 recomputation up to 2.1 %, than the tuned 16-thread recomputation of all structures up to 2.8 % and than its sequential version up to 7.3 % (linear interpolation between the sampled sizes). Above these fractions computing from scratch is cheaper, and initializing an incremental mesh from the recomputed structures adds to the cost of recomputation.

9.4 Interleaved queries and edits

Recursive bisection queries the elements around an edge before each bisection, so recomputing the transpose on the first query after each edit computes it almost once per edit. Figure 15 runs the same algorithm, with the same decisions, on the incremental mesh and on an MM3 many-to-many relation, which recomputes in this way; the ball has radius 0.25 and the final meshes are identical. The incremental version costs 2.9 to 3.5 µs per bisection from 1.3×10^3 to 1.05×10^5 elements; the MM3 version grows from 7.2 to 383 µs (497 µs with collector pauses). Refining the mesh of 1.05×10^5 elements (55 716 bisections) takes 0.16 s against 21 s with MM3 (28 s with 6.4 s of pauses).

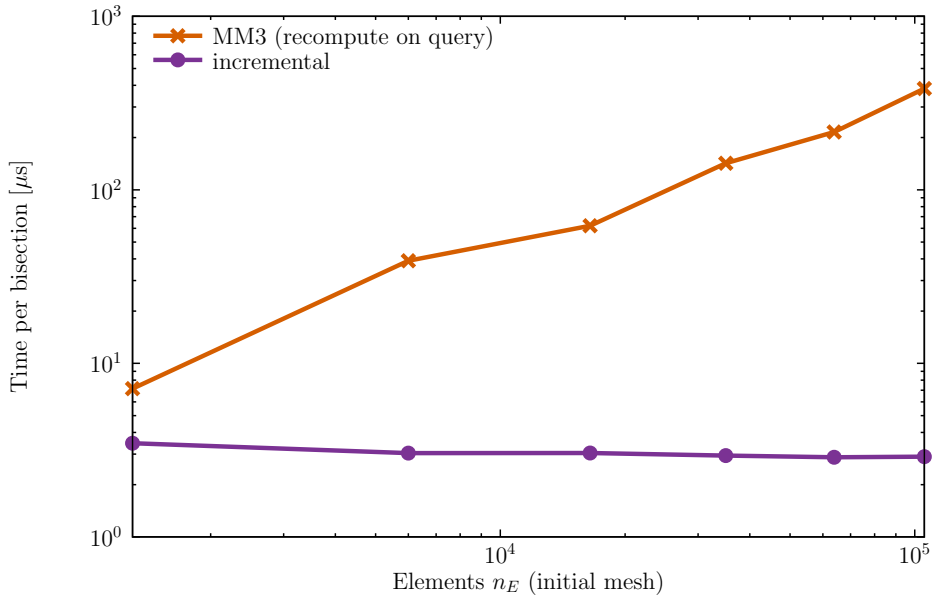


Fig. 15 Recursive bisection of the elements in a ball of radius 0.25: time per bisection when the edge queries read the incremental structures and when MM3 recomputes the transpose on the first query after each edit; times exclude collector pauses.

Table 8 Commit, rollback, first-edit and memory costs of versioning at two mesh sizes, with one transient-mode edit as reference.

Quantity	9.98×10^5 el.	1.01×10^7 el.
Commit without edits, mean of 10^4 calls	0.14 µs	0.14 µs
Rollback to a committed version, mean of 10^4 calls	0.08 µs	0.07 µs
First edit after a commit	157 µs	1.8 ms
One edit after a full collection, transient mode	5.1 µs	16 µs
Managed heap of the process, latest version	583 MB	5 417 MB
Retained memory per crack-advance version	2.6 MB	10.1 MB
Retained memory per edit	5.2 kB	9.4 kB

Table 9 One crack step on 9.98×10^5 elements followed by the assembly of a P1 operator and 100 Jacobi-preconditioned conjugate-gradient iterations [ms, medians over the crack steps; sequential assembly and solver]. The incremental topology time includes the refresh of the flat copy of the pattern (0.07 ms).

Path	topology	assembly	solve	step
Incremental (gapped pattern, destinations)	1.40	56.2	121	178
MM3 recomputation (compact compressed row, cliques)	117	57.2	126	300

9.5 Versioning and memory

The measurements of Table 8 agree with Proposition 7. A commit without edits and a rollback cost a fraction of a microsecond at both sizes. A commit after a step of the elliptical-crack growth costs 4 to 14 µs, and the first edit after a commit copies directories proportional to the mesh size. Each retained version adds the directories and chunks copied in its epoch, 5.2 to 9.4 kB per edit for the crack, whose front cuts across the element numbering. Including element rows, the incremental mesh uses 232 and 231 bytes per element at 10^6 and 10^7 elements, MM3 405 and 403 (rows, transpose, position tables, clique destinations, node–node product) and the tuned recomputation of all structures 206 (154 derived, 52 rows). The heap row of Table 8 is the managed heap of the whole process; the retained-memory rows are differences between heap readings.

Readers of committed versions do not wait for the writer. With one to fifteen reader threads, reads of the latest committed version reach 2.7×10^7 to 8.3×10^8 per second while a writer advances a crack on 9.98×10^5 elements and commits each step (Appendix B).

Table 10 Concurrent cutting of g^3 independent groups (Ryzen 9 9950X, minimum of five samples, each from the same committed version): number of cut edges, sequential cut and concurrent cut with 1 to 16 threads, without collector pauses, ms.

elements	g^3	cut edges	sequential	1	2	4	8	16
10^6	64	82 856	993	1 186	616	331	191	156
10^6	512	162 556	1 921	2 335	1 212	658	378	278
10^7	64	390 296	4 817	5 481	2 901	1 578	919	663
10^7	512	777 088	9 495	11 094	5 751	3 144	1 779	1 285

9.6 A model step

Table 9 measures a model implicit step: a crack advance by node duplication, the assembly of a P1 stiffness-plus-mass operator through the destinations, and exactly 100 conjugate-gradient iterations. The topology time falls from 39.0 % of the MM3 step to 0.8 % of the incremental one; assembly costs the same on both paths. The topology time of each step is a single sample after a full collection, and the medians over the steps therefore exceed the minima of Table 6 on the same mesh (0.970 ms and 79.0 ms). Conjugate-gradient iterations on snapshot reads take 198 ms, 58 % more than on the compact array of the recomputed pattern, because snapshot reads traverse two-level directories and zero-count slots. The solver therefore reads the flat copy of the pattern (Section 6), refreshed after every commit by copying the changed chunks: a median of 173 of 3 954, in 0.07 ms (2.6 ms for the first, complete copy). With the flat copy, iterations take 121 ms against 126 ms, and the step is 1.68 times faster overall; the ratio tends to 1 as the solve dominates. Acharya and Hyde report frame times 1 to 3 % shorter when the solve dominates and 18 to 20 % shorter in a synthetic dynamic elasticity loop [1]. On 16 threads the assembly takes 18.3 ms gathering row by row through the transpose, which reproduces the element loop bit for bit, and 48.5 ms looping over the color classes, sorting included.

9.7 Local and concurrent cutting

Table 10 gives the cost of a concurrent cut of g^3 groups of edges crossing spheres of radius $0.35/g$ centered in the cells of the unit box, with no merged group (the independent case of Section 7.3), without collector pauses. On one thread the concurrent cut costs 1.14 to 1.22 times the sequential cut. On 16 cores of the Ryzen 9 9950X it is 6.3 and 6.9 times faster with 64 and 512 groups at 10^6 elements and 7.3 and 7.4 times at 10^7 (5.0 to 9.0 with the pauses, which differ between the two cuts). On a Deucalion node with two EPYC 7742 processors (128 cores, 16 memory channels against two of the 9950X, server collector with 16 heaps) the sequential cut of 10^7 elements takes 13.1 s with 64 groups and 24.4 s with 512. There the concurrent cut of 64 and 512 groups is 10.8 and 11.6 times faster on 32 threads and 11.7 and 12.1 times faster on 64 (Figure 16). With 64 collector heaps the concurrent cut of 512 groups takes 2.1 s on 64 threads (2.0 s with 16 heaps) and the sequential cut 26.9 s. At 2.5×10^7 elements the 512 groups take 46.3 s sequentially and 4.5 s on 128 threads.

Table 11 compares the sequential local cut with the batch cutter of [7] (its C# translation) followed by a tuned 16-thread recomputation of all structures, on the same marks. For a single group of 302 to 6 122 cut edges the batch path takes 0.58 to 6.0 s, 37 to 269 times the local cost, which follows the number of cut edges. With all groups, the local cut is 1.48 times faster when the shells of the marked edges hold 5.9 % of the elements (10^7 elements, 64 groups), and the batch path 1.2 to 2.3 times faster when they hold 11.7 to 25 % (the other three rows). The concurrent cut of Table 10 on 16 threads takes 156 to 1 285 ms for the same marks. The sequential cuts of Tables 10 and 11, from separate runs, differ by at most 1.5 % (4 817 and 4 779 ms at 10^7 elements and 64 groups, 993 and 980 ms at 10^6).

10 Conclusions

Local Steiner-point-free cutting inserts and grows cracks in a conforming tetrahedral mesh, and a rejected increment is rolled back and repeated. After every edit the transpose with node positions, the node–node pattern, assembly destinations, facet adjacency and boundary facets equal their definitions and the coloring is proper. The cost per edited element depends on the longest transpose and pattern rows, not on the size of the mesh; in persistent mode the first edit after a commit adds a copy of the directories whose size is proportional to the mesh size (1.8 ms at 10^7 elements). Pair counts make deletions exact and keep the destinations of unedited elements, and copy-on-write chunks let one rollback undo a rejected step while readers never wait for the writer. The independent program found no mismatch in the randomized

Table 11 Local cut against the batch cutter of [7] followed by the tuned 16-thread recomputation of all structures, on the same marks (Ryzen 9 9950X, minimum of three samples, without collector pauses, ms); “one” is the first group alone, and the ratio is (batch + recomputation)/local.

elements	g^3	marks	cut edges	local	batch	recomputation	ratio
10^6	64	all	82 856	980	619	130	0.76
		one	1 274	15.5	506	74.9	37.4
10^6	512	all	162 556	1 920	692	160	0.44
		one	302	4.43	529	83.8	138
10^7	64	all	390 296	4 779	6 138	946	1.48
		one	6 122	78.4	5 231	751	76.3
10^7	512	all	777 088	9 514	6 821	1 035	0.83
		one	1 550	21.5	5 183	617	269

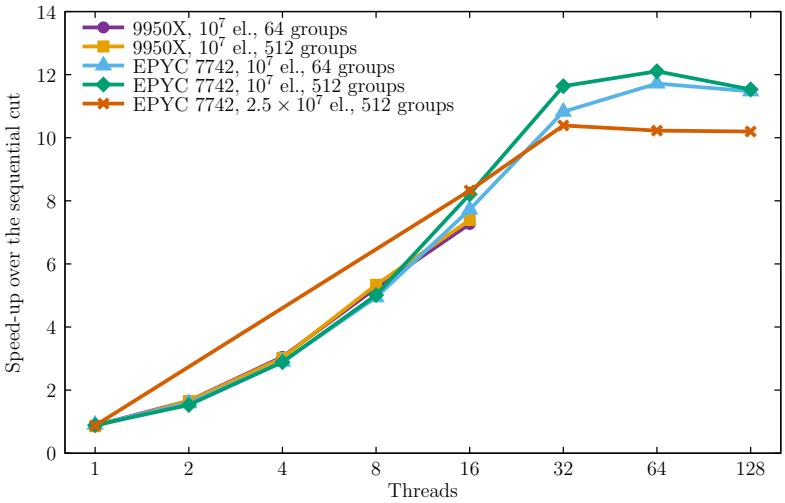


Fig. 16 Concurrent cutting of independent groups: speed-up over the sequential cut on the Ryzen 9 9950X (16 cores) and two EPYC 7742 (128 cores), minimum of five samples (three at 2.5×10^7 elements), without collector pauses.

edits, the workloads, the cutter tests and the committed versions of the surface-crack and corner-crack growth sequences.

The interaction integral returns factors averaged along the front. Against the penny-crack solutions averaged in the same way, they differ by at most 0.29 % of K_0 at $h_{\min}/a = 1/64$; against the pointwise solutions by at most 0.51 %, the averaging alone shifting the second and third factors by up to 0.59 and 0.42 %. On the finest meshes F differs from Table I of NASA TP-1578 by at most 3.14 and 1.53 %, and the free-surface term brings the factor at the surface point from 8.7 to 3.1 % of the equation of Newman and Raju. The corner crack demonstrates cutting and versioning along a curved mixed-mode path. Its factors have no reference value and omit the integral over the curved crack faces, and the criterion neglects K_{III}, so the path is not validated.

On 10^7 tetrahedra, updating the connectivity after a crack advance by node duplication takes 3.0 ms, against 849 ms for the MM3 recomputation and 300 ms for the tuned 16-thread recomputation. A complete step gains only where topology updates are a substantial part of it: a model step with 100 conjugate-gradient iterations on 10^6 elements runs 1.68 times faster, and recursive bisection, which queries between edits, refines 1.05×10^5 elements in 0.16 s against 21 s. In the Paris-law growth examples the solve dominates, and fitting, crack increment and commit take 0.5 to 1.6 % of a step; there the versions provide the rollback and the local cut keeps the fields outside it. Independent groups are cut 6.3 to 7.4 times faster than sequentially on 16 cores. Recomputation is faster when a batch of random node duplications edits more than 0.30 to 2.8 % of 10^6 elements, and batch cutting plus recomputation was faster when the cut shells held 11.7 to 25 % of the elements.

The baselines are our own programs, timed on one workstation and on two nodes of Deucalion. A pattern row holds at most 1 023 neighbors, the facet rule assumes simplices, the refined band is chosen before growth starts, recursive bisection cannot refine template children, and each example grows one crack with the sequential cutter. The templates do not bound the shape of the children: after cutting,

0.59 % of the tetrahedra of the penny-crack mesh and 0.20 % of those of the last corner-crack mesh have a radius ratio below 0.1 (Section 8.6).

A deeper directory tree would make the first copy after a commit proportional to the touched subtrees, and refinement rules that accept template children would allow refinement along a growing front. Whether the update rules extend, with numeric values in place of counts, to transfer and constraint operators for adaptive fracture analyses is left to future work.

Acknowledgments. The authors acknowledge Fundação para a Ciência e a Tecnologia (FCT) for its financial support via LAETA (project https://doi.org/10.54499/UID/50022/2025). This work was also funded by national funds through FCT under the project DEEPFRAC, 2024.14907.PEX (https://doi.org/10.54499/2024.14907.PEX). This work was supported by FCT I.P. under project 2026.002.CNCA.IST_UL at the Deucalion supercomputer, jointly funded by EuroHPC JU and Portugal.

Declarations

Funding. This work was supported by Fundação para a Ciência e a Tecnologia (FCT) via LAETA, project UID/50022/2025 (https://doi.org/10.54499/UID/50022/2025), by national funds through FCT under the project DEEPFRAC, 2024.14907.PEX (https://doi.org/10.54499/2024.14907.PEX), and by FCT I.P. under project 2026.002.CNCA.IST_UL at the Deucalion supercomputer, jointly funded by EuroHPC JU and Portugal.

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

Ethics approval and consent. Not applicable.

Code availability. The implementation and the verification and measurement programs are available in the ManyToMany repository, https://github.com/PedroAreiasIST/ManyToMany [4], under the GPL-3.0 license. The scripts that regenerate the tables and figures are in the same repository (directory `benchmarks/Step2`).

Data availability. The corner-crack EnSight Gold files (thirteen fronts) are available from the corresponding author on request.

Author contributions. P. Areias: Conceptualization, Data curation, Formal analysis, Funding acquisition, Project administration, Software, Validation, Visualization, Writing – original draft. N. Silvestre: Funding acquisition, Investigation, Methodology, Writing – review & editing.

Use of generative AI. During the preparation of this work the authors used Anthropic Claude to assist with writing, editing and coding. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Appendix A Collector pauses and the MM3 baseline

To check the MM3 baseline on the workstation of Section 9.1, we repeated its reported cache recomputation. After one edit, the recomputation takes 0.26, 9.6 and 108 ms for the transpose and 0.86, 25 and 228 ms with the position tables at 10^4, 10^5 and 10^6 quadrilaterals (medians of three runs). At the largest size these times contain collector pauses of 75 and 133 ms; without them they are 32 and 92 ms. The values reported for the i9-13900HX in [8] are 0.50, 4.7 and 56 ms and 1.1, 12 and 173 ms.

Cutting allocates inside the timed region, so collector pauses enter the elapsed time, and the pauses depend on the state of the heap. With the marks ordered by group or by edge, the sequential cut of 10^7 elements with 64 groups takes 6.0 s, with six collections and 1.2 s of pauses, in a process holding only the mesh. With a second mesh of the same size in memory, it takes 4.8 to 5.0 s without a collection. Without the pauses, the minimum over the samples, which may come from different runs, is 4.8 to 4.9 s in both states. At 10^6 elements every run of the sequential cut has one collection, of 164 to 170 ms. Edits and recomputations are therefore timed with their pauses, which occur in at most 10 % of the timed regions (Section 9.1); cuts are timed without them, and their pauses are stated separately.

Appendix B Reads of committed versions

Other threads read a committed mesh when output overlaps the next increment. Table B1 measures single-thread throughput with random probes, in M/s (10^6 operations per second). MM3 stores one-to-many relations as lists of lists and computes facet adjacency from shared node counts. Snapshot element and transpose reads are 1.4 to 2.5 times faster than the lock-free reads of the MM3 one-to-many relation, which follow more indirections. The MM3 many-to-many relation, which takes a reader–writer lock per

Table B1 Single-thread read throughput [10^6 operations/s], random probes; an operation reads a row, one facet neighbor (snapshot), the four facet neighbors of an element (MM3 adjacency) or a 4×4 destination block.

Operation	9.98×10^5 el.	1.01×10^7 el.
Snapshot: element row	80.7	45.3
Snapshot: transpose row	340	66.4
Snapshot: facet neighbor	457	119
Snapshot: destination block (4×4)	88.8	75.9
MM3 one-to-many: element row, no lock	50.0	32.2
MM3 one-to-many: transpose row, no lock	135	38.0
MM3 many-to-many: transpose row, read lock per call	19.4	6.60
MM3 many-to-many: transpose row copied under the lock	13.2	4.47
MM3 adjacency: four facet neighbors	0.67	0.50

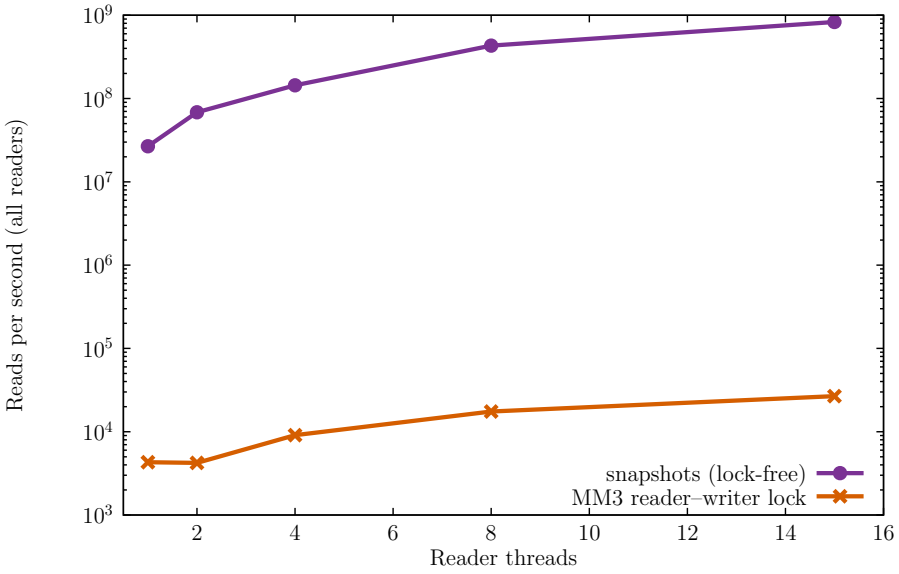


Fig. B1 Reads per second of all reader threads while one writer advances a crack on 9.98×10^5 elements.

call, reaches 4.5 to 19 M/s. MM3 computes the four facet neighbors of an element from shared counts at 0.50 to 0.67 M/s; a snapshot reads one stored facet neighbor at 119 to 457 M/s.

Figure B1 measures a writer advancing the crack on 9.98×10^5 elements and committing each step, with 1 to 15 readers querying a transpose row or a facet neighbor of the latest version at a random index generated in the reader. The generated index lowers the throughput relative to Table B1, which reads precomputed probe arrays. The readers reach 26.7 M/s with one thread and 831 M/s with fifteen, that is 26.7 and 55.4 M/s per thread. Without a writer each reader reaches 76.6 and 75.5 M/s at one and fifteen threads, and with a writer that commits every millisecond without editing 75.7 and 75.1 M/s. The loss therefore comes from the edits, not from the commits; it falls from 65 % with one reader to 27 % with fifteen. Pinning every thread to its own core gives the same rates, and repeated runs differ by at most 15 %. The writer's median step stays between 0.97 and 1.29 ms. With MM3, every edit invalidates the transpose and the next reader recomputes it under the write lock: the readers obtain 4.2 to 27 thousand reads per second and the writer's median step takes 0.59 to 1.59 s.

References

[1] Acharya M, Hyde D (2026) Incremental matrix assembly for preplanned dynamic tetrahedral meshes. arXiv preprint 2605.11673v2, https://doi.org/10.48550/arXiv.2605.11673

[2] Anderson R, Andrej J, Barker A, et al (2021) MFEM: a modular finite element methods library. Comput Math Appl 81:42–74. https://doi.org/10.1016/j.camwa.2020.06.009

[3] Areias P (2026) IST Vector Postprocessor. Version 1.0, Instituto Superior Técnico, Universidade de Lisboa

[4] Areias P (2026) ManyToMany: a high-performance scientific computing and finite element library for .NET. https://github.com/PedroAreiasIST/ManyToMany, GNU General Public License v3.0

[5] Areias P (2026) SimPlas. http://www.simplassoftware.com, Portuguese Software Association (ASSOFT) registry number 2281/D/17

[6] Areias P, Rabczuk T (2017) Steiner-point free edge cutting of tetrahedral meshes with applications in fracture. Finite Elem Anal Des 132:27–41. https://doi.org/10.1016/j.finel.2017.05.001

[7] Areias P, Soeiro J (2025) A simple and efficient mesh cutting algorithm and software library based on many-to-many relations. Adv Eng Softw 205:103919. https://doi.org/10.1016/j.advengsoft.2025.103919

[8] Areias P, Sardinha M, Silvestre N (2026) A multi-level framework for type-safe topological structures with efficient connectivity queries. Adv Eng Softw 222:104296. https://doi.org/10.1016/j.advengsoft.2026.104296

[9] Arnold D, Mukherjee A, Pouly L (2000) Locally adapted tetrahedral meshes using bisection. SIAM J Sci Comput 22(2):431–448. https://doi.org/10.1137/s1064827597323373

[10] Bažant ZP, Estenssoro LF (1979) Surface singularity and crack propagation. Int J Solids Struct 15(5):405–426. https://doi.org/10.1016/0020-7683(79)90062-3

[11] Bernstein P, Goodman N (1983) Multiversion concurrency control—theory and algorithms. ACM Trans Database Syst 8(4):465–483. https://doi.org/10.1145/319996.319998

[12] Bhattacharya S, Chakrabarty D, Henzinger M, et al (2018) Dynamic algorithms for graph coloring. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, p 1–20, https://doi.org/10.1137/1.9781611975031.1

[13] Blakeley J, Larson P, Tompa F (1986) Efficiently updating materialized views. In: Proceedings of the 1986 ACM SIGMOD international conference on Management of data - SIGMOD '86. ACM Press, pp 61–71, https://doi.org/10.1145/16894.16861

[14] Blelloch G, Burch H, Crary K, et al (2001) Persistent triangulations. J Funct Program 11(5):441–466. https://doi.org/10.1017/S0956796801004087

[15] Branco R, Antunes F, Costa J (2015) A review on 3D-FE adaptive remeshing techniques for crack growth modelling. Eng Fract Mech 141:170–195. https://doi.org/10.1016/j.engfracmech.2015.05.023

[16] Budiu M, Chajed T, McSherry F, et al (2023) DBSP: automatic incremental view maintenance for rich query languages. Proc VLDB Endow 16(7):1601–1614. https://doi.org/10.14778/3587136.3587137

[17] Burstedde C, Wilcox L, Ghattas O (2011) p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. SIAM J Sci Comput 33(3):1103–1133. https://doi.org/10.1137/100791634

[18] Carter B, Wawrzynek P, Ingraffea A (2000) Automated 3-D crack growth simulation. Int J Numer Methods Eng 47(1–3):229–253. https://doi.org/10.1002/(SICI)1097-0207(20000110/30)47:1/3⟨229::AID-NME769⟩3.0.CO;2-2

[19] Çatalyürek Ü, Feo J, Gebremedhin A, et al (2012) Graph coloring algorithms for multi-core and massively multithreaded architectures. Parallel Comput 38(10-11):576–594. https://doi.org/10.1016/j.parco.2012.07.001

[20] Davis T, Hager W (1999) Modifying a sparse Cholesky factorization. SIAM J Matrix Anal Appl 20(3):606–627. https://doi.org/10.1137/s0895479897321076

[21] Desnoyers M, McKenney P, Stern A, et al (2012) User-level implementations of read-copy update. IEEE Trans Parallel Distrib Syst 23(2):375–382. https://doi.org/10.1109/TPDS.2011.159

[22] Dhulipala L, Blelloch G, Shun J (2019) Low-latency graph streaming using compressed purely-functional trees. In: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation. ACM, pp 918–934, https://doi.org/10.1145/3314221.3314598

[23] Driscoll J, Sarnak N, Sleator D, et al (1989) Making data structures persistent. J Comput Syst Sci 38(1):86–124. https://doi.org/10.1016/0022-0000(89)90034-2

[24] Dyedov V, Ray N, Einstein D, et al (2015) AHF: array-based half-facet data structure for mixed-dimensional and non-manifold meshes. Eng Comput 31(3):389–404. https://doi.org/10.1007/s00366-014-0378-6

[25] Erdogan F, Sih GC (1963) On the crack extension in plates under plane loading and transverse shear. J Basic Eng 85(4):519–525. https://doi.org/10.1115/1.3656897

[26] Espinha R, Celes W, Rodriguez N, et al (2009) ParTopS: compact topological framework for parallel fragmentation simulations. Eng Comput 25(4):345–365. https://doi.org/10.1007/s00366-009-0129-2

[27] Freudenthal H (1942) Simplizialzerlegungen von beschränkter Flachheit. Ann Math 43(3):580–582. https://doi.org/10.2307/1968813

[28] Gorman G, Rokos G, Southern J, et al (2015) Thread-Parallel anisotropic mesh adaptation. In: New Challenges in Grid Generation and Adaptivity for Scientific Computing. Springer International Publishing, p 113–137, https://doi.org/10.1007/978-3-319-06053-8_6

[29] Gosz M, Moran B (2002) An interaction energy integral method for computation of mixed-mode stress intensity factors along non-planar crack fronts in three dimensions. Eng Fract Mech 69(3):299–319. https://doi.org/10.1016/S0013-7944(01)00080-7

[30] Gosz M, Dolbow J, Moran B (1998) Domain integral formulation for stress intensity factor computation along curved three-dimensional interface cracks. Int J Solids Struct 35(15):1763–1783. https://doi.org/10.1016/S0020-7683(97)00132-7

[31] Gupta A, Mumick I (1999) Maintenance of materialized views: Problems, techniques, and applications. In: Materialized Views: Techniques, Implementations, and Applications. The MIT Press, p 145–158, https://doi.org/10.7551/mitpress/4472.003.0016

[32] Gupta A, Mumick I, Subrahmanian V (1993) Maintaining views incrementally. In: Proceedings of the 1993 ACM SIGMOD international conference on Management of data. ACM, pp 157–166, https://doi.org/10.1145/170035.170066

[33] Heister T, Wheeler M, Wick T (2015) A primal-dual active set method and predictor-corrector mesh adaptivity for computing fracture propagation using a phase-field approach. Comput Methods Appl Mech Eng 290:466–495. https://doi.org/10.1016/j.cma.2015.03.009

[34] Ibanez D, Shephard M (2017) Modifiable array data structures for mesh topology. SIAM J Sci Comput 39(2):C144–C161. https://doi.org/10.1137/16m1063496

[35] Ibanez D, Seol E, Smith C, et al (2016) PUMI: parallel unstructured mesh infrastructure. ACM Trans Math Softw 42(3):1–28. https://doi.org/10.1145/2814935

[36] Irwin G (1962) Crack-extension force for a part-through crack in a plate. J Appl Mech 29(4):651–654. https://doi.org/10.1115/1.3640649

[37] Jiang Z, Dai J, Hu Y, et al (2022) Declarative specification for unstructured mesh editing algorithms. ACM Trans Graph 41(6):1–14. https://doi.org/10.1145/3550454.3555513

[38] Kassir M, Sih G (1966) Three-dimensional stress distribution around an elliptical crack under arbitrary loadings. J Appl Mech 33(3):601–611. https://doi.org/10.1115/1.3625127

[39] Kim H, Kim H (2024) An adaptive continuous–discontinuous approach for the analysis of phase field fracture using mesh refinement and coarsening schemes and octree-based trimmed hexahedral meshes. Comput Mech 74(5):1171–1196. https://doi.org/10.1007/s00466-024-02472-6

[40] Kirk B, Peterson J, Stogner R, et al (2006) libMesh: a C++ library for parallel adaptive mesh refinement/coarsening simulations. Eng Comput 22(3-4):237–254. https://doi.org/10.1007/s00366-006-0049-3

[41] Knepley M, Karpeev D (2009) Mesh algorithms for PDE with Sieve I: mesh distribution. Sci Program 17(3):215–230. https://doi.org/10.1155/2009/948613

[42] Koch C (2010) Incremental query evaluation in a ring of databases. In: Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM, pp 87–98, https://doi.org/10.1145/1807085.1807100

[43] Koch C, Ahmad Y, Kennedy O, et al (2014) DBToaster: higher-order delta processing for dynamic, frequently fresh views. VLDB J 23(2):253–278. https://doi.org/10.1007/s00778-013-0348-4

[44] Kuhn H (1960) Some combinatorial lemmas in topology. IBM J Res Dev 4(5):518–524. https://doi.org/10.1147/rd.45.0518

[45] Lange M, Mitchell L, Knepley M, et al (2016) Efficient mesh management in Firedrake using PETSc DMPlex. SIAM J Sci Comput 38(5):S143–S155. https://doi.org/10.1137/15m1026092

[46] Logg A (2009) Efficient representation of computational meshes. Int J Comput Sci Eng 4(4):283–295. https://doi.org/10.1504/IJCSE.2009.029164

[47] Macko P, Marathe V, Margo D, et al (2015) LLAMA: efficient graph analytics using large multiversioned arrays. In: 2015 IEEE 31st International Conference on Data Engineering. IEEE, pp 363–374, https://doi.org/10.1109/ICDE.2015.7113298

[48] Mahmoud A, Porumbescu S, Owens J (2025) Dynamic mesh processing on the GPU. ACM Trans Graph 44(4):1–19. https://doi.org/10.1145/3731162

[49] Maligno A, Rajaratnam S, Leen S, et al (2010) A three-dimensional (3D) numerical study of fatigue crack growth using remeshing techniques. Eng Fract Mech 77(1):94–111. https://doi.org/10.1016/j.engfracmech.2009.09.017

[50] Maubach J (1995) Local bisection refinement for N-simplicial grids generated by reflection. SIAM J Sci Comput 16(1):210–227. https://doi.org/10.1137/0916014

[51] Moës N, Gravouil A, Belytschko T (2002) Non-planar 3D crack growth by the extended finite element and level sets—Part I: mechanical model. Int J Numer Methods Eng 53(11):2549–2568. https://doi.org/10.1002/nme.429

[52] Moran B, Shih C (1987) Crack tip and associated domain integrals from momentum and energy balance. Eng Fract Mech 27(6):615–642. https://doi.org/10.1016/0013-7944(87)90155-X

[53] Mota A, Knap J, Ortiz M (2008) Fracture and fragmentation of simplicial finite element meshes using graphs. Int J Numer Methods Eng 73(11):1547–1570. https://doi.org/10.1002/nme.2135

[54] Neumann T, Mühlbauer T, Kemper A (2015) Fast serializable multi-version concurrency control for main-memory database systems. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. ACM, pp 677–689, https://doi.org/10.1145/2723372.2749436

[55] Newman JC, Raju IS (1979) Analysis of surface cracks in finite plates under tension or bending loads. NASA Technical Paper 1578, NASA Langley Research Center, URL https://ntrs.nasa.gov/citations/19800005257

[56] Newman JC, Raju IS (1981) An empirical stress-intensity factor equation for the surface crack. Eng Fract Mech 15(1–2):185–192. https://doi.org/10.1016/0013-7944(81)90116-8

[57] Nikolic M, ElSeidy M, Koch C (2014) LINVIEW: incremental view maintenance for complex analytical queries. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data. ACM, pp 253–264, https://doi.org/10.1145/2588555.2610519

[58] Okasaki C (1998) Purely Functional Data Structures. Cambridge University Press, https://doi.org/10.1017/CBO9780511530104

[59] Pais M, Yeralan S, Davis T, et al (2012) An exact reanalysis algorithm using incremental Cholesky factorization and its application to crack growth modeling. Int J Numer Methods Eng 91(12):1358–1364. https://doi.org/10.1002/nme.4333

[60] Paluszny A, Zimmerman R (2011) Numerical simulation of multiple 3D fracture propagation using arbitrary meshes. Comput Methods Appl Mech Eng 200(9–12):953–966. https://doi.org/10.1016/j.cma.2010.11.013

[61] Pandolfi A, Ortiz M (2002) An efficient adaptive procedure for three-dimensional fragmentation simulations. Eng Comput 18(2):148–159. https://doi.org/10.1007/s003660200013

[62] Paris P, Erdogan F (1963) A critical analysis of crack propagation laws. J Basic Eng 85(4):528–533. https://doi.org/10.1115/1.3656900

[63] Paulino G, Celes W, Espinha R, et al (2008) A general topology-based framework for adaptive insertion of cohesive elements in finite element meshes. Eng Comput 24(1):59–78. https://doi.org/10.1007/s00366-007-0069-7

[64] Raju IS, Newman JC (1979) Stress-intensity factors for a wide range of semi-elliptical surface cracks in finite-thickness plates. Eng Fract Mech 11(4):817–829. https://doi.org/10.1016/0013-7944(79)90139-5

[65] Schöllmann M, Fulland M, Richard H (2003) Development of a new software for adaptive crack growth simulations in 3D structures. Eng Fract Mech 70(2):249–268. https://doi.org/10.1016/S0013-7944(02)00028-0

[66] Seol E, Shephard M (2006) Efficient distributed mesh data structure for parallel automated adaptive analysis. Eng Comput 22(3–4):197–213. https://doi.org/10.1007/s00366-006-0048-4

[67] Shih C, Moran B, Nakamura T (1986) Energy release rate along a three-dimensional crack front in a thermally stressed body. Int J Fract 30(2):79–102. https://doi.org/10.1007/BF00034019

[68] Sneddon I (1946) The distribution of stress in the neighbourhood of a crack in an elastic solid. Proc R Soc Lond A 187(1009):229–260. https://doi.org/10.1098/rspa.1946.0077

[69] Song J, Areias P, Belytschko T (2006) A method for dynamic crack and shear band propagation with phantom nodes. Int J Numer Methods Eng 67(6):868–893. https://doi.org/10.1002/nme.1652

[70] Stevenson R (2008) The completion of locally refined simplicial partitions created by bisection. Math Comput 77(261):227–241. https://doi.org/10.1090/S0025-5718-07-01959-X

[71] Sukumar N, Chopp D, Moran B (2003) Extended finite element method and fast marching method for three-dimensional fatigue crack propagation. Eng Fract Mech 70(1):29–48. https://doi.org/10.1016/S0013-7944(02)00032-2

[72] Traxler C (1997) An algorithm for adaptive mesh refinement in n dimensions. Computing 59(2):115–137. https://doi.org/10.1007/BF02684475

[73] Walters M, Paulino G, Dodds R (2005) Interaction integral procedures for 3-D curved cracks including surface tractions. Eng Fract Mech 72(11):1635–1663. https://doi.org/10.1016/j.engfracmech.2005.01.002

[74] Wheatman B, Xu H (2018) Packed compressed sparse row: A dynamic graph representation. In: 2018 IEEE High Performance extreme Computing Conference (HPEC). IEEE, pp 1–7, https://doi.org/10.1109/HPEC.2018.8547566

[75] Yau J, Wang S, Corten H (1980) A mixed-mode crack analysis of isotropic solids using conservation laws of elasticity. J Appl Mech 47(2):335–341. https://doi.org/10.1115/1.3153665