Solid Modeling Aspects of Three-Dimensional Fragmentation

A. Pandolfi and M. Ortiz

1 Dipartimento di Ingegneria Strutturale, Politecnico di Milano, Milano, Italy; 2 Graduate Aeronautical Laboratories, California Institute of Technology, Pasadena, CA, USA

Abstract. The ways in which the topology and geometry of a three-dimensional finite-element model may evolve as a consequence of fracture and fragmentation are enumerated, and the actions which may be taken in order to update the boundary representation of the solid so as to faithfully reflect that evolution are described. Arbitrary topological and geometrical evolution of a three-dimensional solid, not necessarily restricted to an evolution of its surface, are addressed. Solids are described by their boundary representation (BRep) and a surface and volume triangulation. Fracture processes are modeled by the introduction of cohesive elements at element interfaces. Simple rules are shown to enable the simulation of strikingly complex crack patterns. The scope and versatility of the approach is illustrated with the aid of selected examples of application.

Keywords. Automatic meshing; Boundary representation; Finite element; Solid modeling; Three-dimensional Fragmentation; Topological Evolution

1. Introduction

The application of most meshing algorithms requires a complete topological and geometrical description of the domain of analysis. For example, the domain may be described as a collection of bodies, each comprising several subbodies. For manifold solids, the boundary of each subbody may then be described as a collection of oriented shells, faces and edges, and their adjacencies [1–3]. This boundary representation (BRep) of the domain must be supplemented with a geometrical description of the faces and edges which make up the various shells of the boundary. A number of techniques are presently available for effecting such geometrical description, including analytical representations, e.g. using rational Bezier patches [4,5]; piecewise or finite-element interpolation [6,7]; and the use of subdivision surfaces [8–10].

Meshing algorithms use solid models variously. For example, advancing front algorithms [11–17] begin by triangulating the surface of the domain, and subsequently propagate the mesh into the interior by the addition of one tetrahedral element at a time. Nodal insertion algorithms [18,19] work by introducing nodes within the domain in accordance with a prescribed nodal density function derived, e.g. from error estimation. The nodes are subsequently triangulated, e.g. by recourse to a constrained Delaunay algorithm [7,20–32]. In a notable variation upon this latter theme, full subdivision meshes, hexahedral or otherwise, are fitted into the domain and the intervening space left between the subdivision mesh and the boundary is patched using stencils [7,33–38].

In all of these methods, it becomes necessary at some point during the application of the algorithm to determine whether a point lies within or without the domain of analysis, which in turn necessitates a complete topological and geometrical description of the body. Advancing front methods further require an initial triangulation of the surface. Methods of surface triangulation generally require the computation of intersections between the surfaces and objects such as rays and circles (e.g. [5,7]), which in turn calls for a full geometrical description of the surface. Other basis operations rely on solid modeling data as well. For instance, both advancing front algorithms and constrained Delaunay require the determination of intersections between trial tetrahedra and either the surface or the meshing front. The determination of these intersections may be based on a triangulation of the surface.

Ultimately, a principal objective of computational mechanics is to marry the principles of mechanics with the tools of computational geometry and functional analysis. The interplay between geometry and
mechanics becomes vividly apparent in applications where the topology of the domain may change, often extensively, in the course of calculations. Examples of these applications include wetting or seizure, whereby two surfaces merge as a result of surface tension or local melting [39-41]; and fracture and fragmentation, which may result in a runaway proliferation of bodies in the model [42,43]. The complexity of fragmentation processes has been amply documented in the experimental literature (see, e.g., [44-47], for a sampler of representative works). The accurate simulation of these processes often requires the bridging of multiple scales, which strongly suggests the need for continuous adaptive remeshing. However, as noted earlier, the ability to remesh may be critically dependent on maintaining an updated topological and geometrical description of the model at all times. In particular, every change in the topology and geometry of the reference configuration of the model needs to be dutifully recorded as a change in the solid model, which thus evolves in time.

Camacho and Ortiz [42,43] have established the feasibility of: (1) accounting explicitly for individual cracks as they nucleate, propagate, branch and possibly link up to form fragments; and (2) simulating explicitly the granular flow which ensues following widespread fragmentation. In Camacho and Ortiz’s approach, cracks are allowed to form and propagate along element boundaries in accordance with a cohesive-law model [48,49]. Clearly, it is incumbent upon the mesh to provide a rich enough set of possible fracture paths, an issue which may be addressed within the framework of adaptive meshing. In contrast to other approaches [48,49] which require interfacial elements to be inserted at the outset along potential fracture paths, Camacho and Ortiz [42] adaptively create new surface as required by the cohesive model by duplicating nodes along previously coherent element boundaries. The nodes are subsequently released in accordance with a tension-shear cohesive law. These simple rules permit the simulation of strikingly complex crack patterns.

In the Camacho and Ortiz approach, the requisite cohesive laws which encapsulate the fracture behavior of the solid are embedded into cohesive finite elements. These elements are surface-like and are compatible with general bulk finite element discretizations of the solid, including those which account for plasticity and large deformations. De Andrés et al. [50] have extended the formulation to three dimensions, and Ortiz and Pandolfi [30] have developed a class of finite-deformation three-dimensional cohesive elements.

Fracture and fragmentation may be modeled by a variety of other means. Potapov [51,52] has developed a method in which elements are held together by compliant surfaces which fracture upon the attainment of a critical tensile strength. A similar approach has been developed by Rice and Ting [53] within the framework of explicit dynamics. However, these approaches are not specifically designed to model an evolutionary geometry.

The boundary element method is particularly appealing as a framework for analyzing solids with evolving geometries as it reduces the representational requirements to the description of the boundary. Wawrzynek and Ingraffea [54,55], and Marthe et al. [56,57] have addressed the topological and geometrical aspects of crack modeling, and how such aspects affect the representation of the solid. In their approach, fracture is simulated based on a geometric representation of the structure – via solid modeling – and flaw initiation and growth is tracked by the application of local remeshing algorithms. However, the scope of the boundary element method is essentially limited to linear problems and homogeneous solids. While some of these limitations may be overcome by discretizing the interior of the solid, the chief representational advantage of the boundary element method, namely, the fact that it solely requires the description of the surface of the solid, is lost in such extensions. A purely surface-based method of linear analysis is unable to predict key aspects of fracture and fragmentation processes such as: the effect of material defects and inhomogeneity; the spontaneous nucleation of flaws in the interior of the solid, e.g. by tensile waves resulting in spallation; and the influence of material nonlinearity on fracture.

The principal objective of the present paper is the enumeration of the ways in which the topology and geometry of a three-dimensional finite-element model may evolve as a consequence of fracture and fragmentation, and to describe such actions as may be taken to update the boundary representation of the solid so as to faithfully reflect that evolution. Thus, our approach is similar in spirit to that of Wawrzynek and Ingraffea [54,55], and Marthe et al. [56,57], but here we concern ourselves with a completely arbitrary topological and geometrical evolution of a three-dimensional solid, not necessarily restricted to an evolution of its surface. The solid is described by its boundary representation and a surface and volume triangulation. We restrict our attention to fracture processes which are modeled by the introduction of cohesive elements at element interfaces [42,50,58].

2. Bound in 3D

For the n case of domains
omorphing angulation nonmanif point, do ever, in e cases whi of nonman

The b Rep or descriptio ency of boundary regarded ture in th
The organization of the paper is as follows. The essential elements of the boundary representation of a solid required in subsequent discussions are briefly summarized in Section 2. Section 3 describes specialized data structures designed to facilitate geometrical operations. Section 4 describes relevant aspects of cohesive elements as they bear on the simulation of fracture and fragmentation. Section 5 enumerates the geometrical and topological changes which are induced by the introduction of a cohesive element in the mesh. Finally, Section 6 collects selected examples of applications which illustrate the scope and versatility of the approach.

2. Boundary Representation Concepts in 3D

For the most part, we restrict our discussion to the case of manifold solids and presume that the domains of interest are topological polyhedra homeomorphic to simplicial complexes which define triangulations of the domains (e.g. [1–3]). Inevitably, nonmanifold situations, such as shell pinched at a point, do indeed arise during fragmentation. However, in our experience the number of nonmanifold cases which may arise is limited and a full treatment of nonmanifold solids is not required.

The boundary representation of a domain, or BRep, consists of a topological and geometrical description of the connectivity, incidence and adjacency of vertices, edges, and faces defining the boundary of the bodies (Fig. 1). A BRep may be regarded as an embedding of the topological structure in the euclidean space \( \mathbb{R}^3 \).

The incidence and adjacency relations between the various objects in the BRep may be formulated as a graph (e.g. [2,59]) (Fig. 2). The root of the graph points to a collection of bodies. The number of bodies can grow considerably during calculations as a result of fragmentation. Each body may comprise several subbodies, e.g. made out of different materials. By assumption each subbody defines a 3-manifold with boundary and, consequently, the boundary of each subbody defines a 2-manifold without boundary.

The connected components of the boundaries of the subbodies are termed shells. It follows, therefore, that each shell defines a connected 2-manifold without boundary. A complete classification of such objects is known from differential topology [59]. The equivalence classes are represented by spheres with a finite number of handles. The number of handles appended to the sphere is known as the genus of the surface. The shells can be oriented consistently so as to unambiguously define an interior and an exterior for each subbody.

The shells may be partitioned into faces defining regions whose boundary must be preserved by the meshers. Thus, the boundaries of the faces represent salient geometric features of the shell such as ridges or sharp edges. The trivial case of a shell which consists of one single face is also possible. The faces are also regarded as oriented surfaces, but this

---

Fig. 1. Example of boundary representation (BRep).

Fig. 2. The structure of the BRep graph.
orientation is assigned arbitrarily. One face may be shared by two shells, e.g. at a material boundary, in which case it appears in each shell with opposite orientations.

Each face may be regarded as a 2-manifold with boundary. The boundary of a face is itself a 1-manifold without boundary. The connected components of the boundary of a face are known as loops. It is known from differential topology [59] that a loop is topological equivalent to the unit circle.

The loops may be partitioned into edges whose end vertices must be preserved by the mesher. The trivial case of a loop which consists of one single edge is also possible. The edges are regarded as oriented segments, but this orientation is assigned arbitrarily. One edge may be shared by two loops, in which case it appears in each loop with opposite orientations. This completes the topological definition of the BRep.

In addition to topological information, the BRep encompasses geometrical data as well. These data describe the actual shape of the boundary, and can take the form of analytical expressions, implicit or parametric, describing the edges and faces; or a recursive definition such as subdivision rules; or a representation based on a discretization of the edges and faces, e.g. by means of finite elements. This latter situation is typically encountered in the course of remeshing, in which case the surface is defined by the old mesh.

Throughout subsequent discussions we shall specifically assume that the interior and the surface of the domain of analysis is triangulated, and that, therefore, the geometry of the model is defined by the mesh itself. Fracture processes will be modeled through the introduction of cohesive elements at element interfaces, with an attendant creation of new surface. The problem which we address then is how the introduction of that new surface modifies the BRep.

3. Data Structures

The general boundary representation just described provides a suitable basis for automatic meshing. The algorithm used to generate all meshes presented subsequently has been described elsewhere [7]. The algorithm begins by meshing all edges in the BRep, followed by a triangulation of the faces using an advancing front method. The density of edge, surface and interior nodes is determined by a prescribed nodal density function defined, e.g. by error esti-

![Diagram of Tetrahedron data structure](image)

**Fig. 3. Tetrahedron data structure: it points to the four adjacent facets and the four adjacent tetrahedra.**

![Diagram of winged facet and segment](image)

**Fig. 4. (a) Winged facet, and (b) Winged segment data structures.**

Finally, a tetrahedral mesh is defined by a constrained Delaunay triangulation. The aim of this latter step is to construct a triangulation whose restriction to the boundary is given, and whose tetrahedra satisfy the circumsphere condition (see also [32]). Once the mesh is constructed, the tetrahedra are converted into ten-node elements by the addition of nodes to the midpoints of all tetrahedral edges.

As the mesh is constructed, a number of data structures are set up with a view to facilitating the management of the BRep. Thus, in addition to the previously described BRep data, we store the following lists: the tetrahedra, (Fig. 3), in each subbody; the facets in each boundary face; and the segments in each boundary edge. All the nodal points in the mesh are stored in a point-region octree, a data structure which reduces the complexity of search and retrieval operations [61]. The tetrahedra lists...
are themselves organized as structures, in which every tetrahedron points to the four tetrahedra incident to its four facets.

Two additional data structures, named wingedfacet and wingedsegment, are particularly useful in fragmentation applications. The wingedfacet structure (Fig. 4(a)) consists of a list of 2-simplices, or facets, each of which is incident to two adjacent tetrahedra, or, alternatively, lies on an external boundary in which case it is adjacent to one single tetrahedron. The facets are incident to six nodes, three of which are corner nodes and the remaining three are midside nodes, all of which are shared with the adjacent tetrahedra. The nodes in each facet are numbered sequentially. This local numbering uniquely defines the orientation of the facet. Each facet in the structure points to its two adjacent tetrahedra, designated in accordance with the orientation of the facet. We adopt the following convention: the top (bottom) or positive (negative) tetrahedron is that from which the nodes of the facet appear to be traversed counterclockwise (clockwise). For facets which are on the surface, the bottom tetrahedron is null. Each sub-body in the model defines the root of a wingedfacet structure.

Likewise, the wingedsegment structure (Fig. 4(b)), consists of a list of segments, each of which is incident to two adjacent surface facets. The segments are incident to three surface nodes, two of which are ending nodes and the remaining one is a midside node, all of which are shared with the adjacent facets. The nodes in each segment are numbered sequentially. This local numbering uniquely defines the orientation of the segment. Each segment in the structure points to its two adjacent facets, designated in accordance with the orientation of the segment. We adopt the following convention: the left (right) facet is that from which the nodes of the segment appear to be traversed from right to left (left to right). Each face in the model defines the root of a wingedsegment structure.

4. Fragmentation Concepts

The three-dimensional tracking of dynamic cracks in solids undergoing fragmentation and, possibly, large-scale plasticity remains a challenging goal at present (see [62] for a recent contribution). A fundamental difficulty concerns the applicability of the traditional K and J-based crack-growth initiation and propagation criteria to situations involving large-scale yielding. Even in cases which conform to the assumptions of small-scale fracture mechanics, considerable uncertainties remain as regards the proper choice of crack-growth initiation and propagation criteria that account for loading rate, material interfaces, small crack sizes and other complicating circumstances.

Dugdale [63], Barrenblatt [64], and others pioneered an alternative viewpoint which regards fracture as a gradual phenomenon in which separation takes place across an extended crack 'up', or cohesive zone, and is resisted by cohesive tractions. This theory of fracture permits the incorporation into the analysis of such fracture parameters as the spall strength – the peak cohesive traction – and the fracture energy – the area under the cohesive law – of the material. An appealing feature of this approach is that it does not presuppose a particular type of constitutive response in the bulk of the material, the extent of crack growth, or the size of the plastic zone.

Fig. 5. Search for an external segment. The tetrahedra incident to the segment are traversed from the top tetrahedron: (a) If the bottom tetrahedron is reached then the segment is interior; (b) otherwise, the segment is on the boundary and is incident to two boundary facets.

Fig. 6. Schematic of the segment data acquisition. (a) The segment lies on a face; (b) the segment lies on an edge.
Cohesive laws may conveniently be embedded into cohesive finite elements \([42, 48-50, 58, 66-68]\). These elements are surface-like and are compatible with general bulk finite element discretizations of the solid, including those which account for plasticity and large deformations. Ortiz and Pandolfi \([58]\) have developed a class of three-dimensional cohesive elements consisting of two six-node triangular facets. The opening displacements are described by quadratic interpolation within the element. The element is fully compatible with – and may be used to bridge – pairs of ten-node tetrahedral elements. The elements are endowed with full finite-deformation kinematics and, in particular, are exactly invariant with respect to superposed rigid body translations and rotations.

The particular field of application which we contemplate here is dynamic fragmentation, although static applications may be treated similarly. The analysis proceeds incrementally in time, e.g. by explicit dynamics. Following Camacho and Ortiz \([42]\), cohesive elements are introduced adaptively at element interfaces as required by a fracture – or spall – criterion. For instance, fracture may be supposed to initiate at a previously coherent element interface when a suitably defined effective traction attains a critical value \([42, 50, 58]\). When the fracture criterion is met at an element interface, a cohesive element is inserted, leading to the creation of new surface. In this manner, the shape and location of successive crack fronts is itself an outcome of the calculations.

From the standpoint of the boundary representation of the solid, the insertion of cohesive surfaces results in geometrical and topological changes in the \(BRep\). An example of the former is a simple extension of an existing crack which demands the corresponding extension of one existing shell. Topological changes which may be occasioned by fracture include the creation of a new facet, the addition of a new shell, the joining of two shells, the creation of a new body, and others. To be able to remesh the model, the \(BRep\) must be continually updated to reflect this geometrical and topological evolution.

A distinct possibility would be to simply update the mesh and then to reconstruct the \(BRep\) from the new mesh. However, the reconstruction of a complex \(BRep\) from a mesh may be computationally costly, and here we opt for a direct update of the \(BRep\) structures. The steps that may be followed to effect this update are enumerated in the sequel. A more formal description of the \(BRep\) update may be based on boolean operators, but this avenue is not pursued here.

5. The \(BRep\) Update

Once an interelement facet has been targeted for the insertion of a cohesive element, the attendant \(BRep\) update may conveniently be divided into three steps:

1. Acquire the complete topology of the facet.
2. Classify the opening case and decide the operations to be performed.
3. Modify the \(BRep\) and the mesh.

We proceed to discuss each of these steps in turn.
5.1. Data Acquisition

As discussed in Section 3, each subbody is the root of a wingedfacet structure. When fracture is detected in a facet, the body and the subbody to which it belongs are therefore known automatically. The remaining relevant data pertaining to the facet comprises its six nodes—three corner nodes and three midside nodes—and its two adjacent tetrahedra. As remarked earlier, the orientation of the facet determines the top and bottom tetrahedra. The target facet contains three segments, which are identified by their nodes.

As will become apparent subsequently, the precise actions to be taken as regards the update of the BRep depend critically on whether the segments of the fractured facet are or are not on the boundary of the body. To determine if a segment of the fractured facet is on the surface, we start from its top tetrahedron and, using the tetrahedron data structure described in Section 3, we sequentially traverse the tetrahedra incident to the segment: if the bottom tetrahedron is reached, the segment is interior (Fig. 5(a)); otherwise, the segment is on the surface. In this latter case, the tetrahedron traversal ends on a boundary facet which is identified as the left facet (FL) incident to the segment (Fig. 5(b)). A second tetrahedron traversal is then effected starting from the bottom tetrahedron of the target facet, leading to the right boundary facet (FR) incident to the segment. From the wingedfacet and wingedsegment structures it is then possible to determine the faces and shell to which FL and FR belong and which need to be updated (Fig. 6).

Finally, the ending nodes of each boundary segment of the fractured facet are examined. In particular, we determine whether one or both ending nodes lie on an edge. An enumeration of the different cases of interest is shown in Fig. 7. The cases are:

1. The segment does not touch any loops (a).
2. One ending node of the segment rests on a loop (b, c):
   (a) b is incident to one edge.
   (b) c is incident to a vertex between two edges.
3. Both ending nodes of the segment rest on loops (d, e, f):
   (a) d has its two ending nodes on the same loop but on different edges.
   (b) e has its two ending nodes on two distinct loops and, therefore, on different edges.
   (c) f has its two ending nodes on two different

loops belonging to two distinct facets sharing a common edge.

For each ending node lying on a loop, the preceding (PEE) and the succeeding (SEE) edge-segments

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure9}
\caption{Examples of nonmanifold cases which may arise during fragmentation. A fractured facet has two (a) or three (b) segments on the surface, but some of the corner nodes remain unopened.}
\end{figure}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure10}
\caption{Topological changes induced by the fracturing of a facet, manifold situations. The primed numbers indicate the new inserted nodes: (a) Case 1: no segments on the boundary. A new shell is added to the subbody; (b) Case 2: one segment on the boundary. The midside node is duplicated and the shell is modified; (c) Case 3: two segments on the boundary. The corner node is duplicated, the midside nodes are duplicated and the shell is modified; (d) Case 4: three segments on the boundary. All six nodes are duplicated, the external shell is modified and a new body is added.}
\end{figure}
Open 6-noded winged facet and insert free surface between the adjacent tetrahedra. BRep is automatically updated.

void BRepUpdate()
{
    int splitbody = 1; /* flag to create new body */
    int side1 = 0;    /* side counter variables */
    int side2 = 1;
    int side3 = 2;
    int AlreadyChecked[elements]; /* flag to check tetrahedron connectivity */

    /* Step 1 - Add new 2-manifold elements. */
    Add2Manifold();

    /* Step 2 - Add necessary 1-manifold elements for new facets facetB and facetT. */
    Add1Manifold();

    /* Step 3 - Check whether corner node needs to be duplicated. */
    DuplicateCorner();

    /* Step 4 - Duplicate midside nodes and update loops. */
    DuplicateMidside();

    /* Step 5 - Shells update. */
    ShellUpdate();

    /* Step 6 - Create new body and transfer corresponding data structures. */
    if (splitbody) CreateNewBody();

    return;
}

Fig. 11. Main for the BRep update. The subroutines are reported in Figs 12-17.

void Add2Manifold()
{
    /* Add the 2-manifold components and update the existing adjacencies. */
    faceB = AddFace(); /* Add two new faces (faceB, faceT) */
    faceT = AddFace();
    facetB = AddFacet(faceB); /* Add new facets (facetB, facetT) */
    facetT = AddFacet(faceT);
    newwingedfacet = AddWingedFacet(SB); /* Add new winged facet */
    newwingedfacet->Tetrahedron[Top] = TT; /* Define Top adjacency */
    wingedfacet->Tetrahedron[Top] = NULL; /* Erase Top adjacency of winged facet */
    TT->Adjacency[ib] = NULL; /* Erase TT & TB adjacency */
    TB->Adjacency[it] = NULL;

    return;
}

Fig. 12. Update of the 2-manifold elements.
```c
void Add1Manifold
{
    int side;
    /*
    Add necessary 1-manifold elements for new facets facetB and facetT.
    */
    for (side = 0; side < 3; side++)
    {
        NE[side][Top] = AddNewEdge ()); /* Add edge */
        NES[side][Top] = AddNewEdgeSegment (NE[side][Top]); /* Add edge-segment */
        NWS[side][Top] = AddNewWingedSegment (facetT); /* Add wingedge-segment */
        NWS[side][Bottom] = AddNewWingedSegment (facetB); /* Add wingedge-segment */
    }
    return;
}
```

Fig. 13. Update of the 1-manifold elements.

```c
void DuplicateCorner
{
    int corner;
    /*
    Check whether corner node needs to be duplicated.
    */
    for (corner = 0; corner < 3; corner++)
    {
        if (segment[corner]) splitbody = 0; /* segment inside, no new body */
        if (segment[corner] ) continue; /* left segment not on the boundary */
        if (segment[corner + 1]) continue; /* right segment not on the boundary */
        OC[corner] = wingedfacet->Node[corner]; /* save the original corner node */
        NC[corner] = NULL; /* initialize a new corner node */
        SetCheckToZero (AlreadyChecked);
        /* Check the possibility to duplicate the corner node */
        if (CheckCornerToOpen (TT, TB, OC[corner]))
        {
            NC[corner] = InsertNewNode (); /* insert new corner node */
            TT->Node[corner] = NC[corner]; /* update Tetrahedron elements corner node */
            facetT->Node[corner] = NC[corner];
            wingedfacet->Node[corner] = NC[corner];
            /* verify corner node for all tetrahedra and wingedfacets connected
                to original one; verify corner node for all facets and segments
                connected to updated tetrahedra */
            VerifyTetrahedra (TT, OC[corner], NC[corner]);
        }
        else {
            splitbody = 0; /* corner not open, no new body */
        }
    }
    return;
}
```

Fig. 14. This subroutine checks whether the corner nodes need to be duplicated, and performs the necessary changes to add a new corner node to the elements on the top side of the fractured facet.
Fig. 15. This subroutine duplicates the midside nodes by performing the necessary changes to the elements on the top side of the fractured facet.

along the loop are recorded. The segment f in Fig. 7(b) coincides with an edge-segment. In this case, two pairs of edge-segments, one for each loop, are stored.

In summary, the data which is collected each time a facet fractures is.

- wingedfacet structure (Fig. 4(a)):
  - Six nodes (three corner nodes and three midside nodes).
  - Two adjacent tetrahedra TT and TB.

- wingedsegment structure (Fig. 4(b)):
  - One wingedsegment if the fractured facet is incident to a face (Fig. 6(a)); two winged segments WL and WR, one per face, if the fractured facet is incident to an edge (Fig. 6(b)).
  - Two adjacent facets FL and FR.
  - One face if the fractured facet is incident to a face (Fig. 6(a)); two faces if the fractured facet is incident to an edge (Fig. 6(b)).
  - The shell to which the segment belongs.

- For each ending node of a segment lying on a loop (Fig. 7):
  - One (Fig. 7(a), case b) or two (Fig. 7(a), case c) edges containing the node.
  - One (Fig. 7(a)) or two (Fig. 7(b)) loops containing the node.
  - The edge-segments preceding and succeeding the node.
void ShellUpdate()
{
    /*
    Shells update. Topological changes are depicted in Fig. 10.
    */
    switch (no_segments) {
        case 0: /* no segments on boundaries, Fig. 10a */
            S = AddNewShell(SB); /* add a new shell to subbody */
            break;
        case 1: /* two sides lie on boundaries, Fig. 10c */
            if (Sh[sidex1] != Sh[sidex2]) { /* non-manifold situation: merge two shells */
                S = MergeShells(Sh[sidex1], Sh[sidex2]);
            }
            break;
        case 2: /* three sides lie on boundaries, Fig. 10d */
            if (Sh[sidex1] != Sh[sidex2]) { /* non-manifold situation: merge two shells */
                S = MergeShells(Sh[sidex1], Sh[sidex2]);
                splitbody = 0;
            }
            if (Sh[sidex1] != Sh[sidex3]) { /* non-manifold situation: merge two shells */
                S = MergeShells(Sh[sidex1], Sh[sidex3]);
                splitbody = 0;
            }
            break;
    }
    S = AddFaceToShell(S, faceT); /* add two new faces to shells */
    S = AddFaceToShell(S, faceB);
    return;
}

/*---------------------------------------------*/
void CreateNewBody()
{ /*---------------------------------------------*/
    void CreateNewBody()
    {
        int side;
        /*
        Create new body and transfer to it the corresponding data structures.
        */
        if (TestToCreateNewBody) { /* no connection between TT and TB */
            NB = AddBody(B); /* add new body */
            NSB = AddSubBody(NB); /* add new subbody */
            TransferTetrahedrons(NSB, TT); /* move tetrahedra and winged facets to NSB */
            TransferTetrahedrons(SB, TB); /* check tetrahedra and winged facets in SB */
            TransferFace(NSB); /* transfer faces to new subbody */
            TransferShells(NSB, 0); /* transfer shells to new subbody */
        }
        return;
    }

    Fig. 16. Shells update. Refer to Fig. 10 for a visualization of the topological changes.

    Fig. 17. Sequence of operations to add a new body to the BRep.
5.2. Classification of Cases

Four main cases may be identified depending on whether the fractured facet has zero, one, two or three segments resting on the boundary (Fig. 8). In all cases, the fractured facet is duplicated, which results in the addition of one new facet. The remaining operations to be performed in each case are:

1. No segments are on the boundary: a new shell composed by two new faces is created inside the subbody (Fig. 10(a)).
typedef struct loop_type
{
  loope_{Type} *Edges;      /* first edge on the loop */
  struct loop_type *link;   /* linked loop on the global face */
} loop_{Type};

typedef struct facet_type
{
  node_{Type} *Node[6];     /* 6 nodes of the facet */
  wingedsegment_{Type} *Segments[3]; /* adjacent wingedsegments */
  facet_{Type} *Adjacent[3]; /* adjacent facets */
  struct facet_type *Link;  /* preceding facet on global face */
  struct facet_type *Rlink; /* succeeding facet on global face */
} facet_{Type};

typedef struct globalface_type
{
  loop_{Type} *Loops;      /* first loop on global face */
  facet_{Type} *Facet;     /* first facet on global face */
  wingedsegments_{Type} *Side; /* first winged-segment on global face */
} globalface_{Type};

typedef struct face_type
{
  int Orientation;        /* global faces orientation */
  int GlobalFace;         /* pointer to global faces array */
  struct face_{Type} *Link;  /* linked face */
} face_{Type};

typedef struct shell_type
{
  face_{Type} *Faces;      /* first face of the shell */
  struct shell_type *Link; /* linked shell */
} shell_{Type};

typedef struct subbody_type
{
  tetrahedron_{Type} *Tetrahedron; /* first tetrahedron of subbody */
  wingedfacet_{Type} *WingedFacet; /* first wingedfacet of subbody */
  shell_{Type} *Shell;             /* first shell of subbody */
  struct subbody_{Type} *Link;    /* linked subbody */
} subbody_{Type};

typedef struct body_type
{
  subbody_{Type} *SB;         /* first subbody of body */
  struct body_{Type} *Link;   /* linked body in BRep */
} body_{Type};

Fig. 19. Description of the data structures used in the BRep update.
2. One segment is on the boundary: the segment is duplicated by doubling the midside node and the external shell is modified by the addition the two new faces (Fig. 10(b)).

3. Two segments are on the boundary: the segments are duplicated by doubling the midside nodes; the corner node is duplicated in the manifold case (Fig. 10(c)); the corner node is not duplicated in the nonmanifold case (Fig. 9(a)); in this latter case, if the corner node belongs to two different shells, then the shells are merged and then modified by the addition of two new faces.

4. Three segments are on the boundary: the segments are duplicated by doubling the midside nodes, a corner node is duplicated in the manifold case (Fig. 10(d)); if all three corner nodes are duplicated then a new body is added; a corner node is not duplicated in the nonmanifold case (Fig. 9(b)); in this latter case, if the corner node belongs to two different shells, then the shells are merged and then modified by the addition of two new faces.

Fig. 20. Description of the variables used by the BRep update code.
5.3. BRep Update

The commented C code reported in Fig. 11 gives a detailed account of the operations to be performed in order to update the BRep. Single steps referred in the code are described in Figs. 12-17. Figures 18, 19 and 20 collect the data structures and the variables used in the code.

The most significant subroutines used in the BRep update are reported in Figs. 22, 23, 24 and 26. Particularly, the subroutine reported in Fig. 24 performs loop updates on a face: the four topological situations that can arise and the corresponding changes are depicted in Fig. 25.

Note that, whenever a node has to be duplicated, the new node is added to the top elements of the fractured facet.

6. Examples

In this section, we demonstrate the scope and versatility of the above procedure with the aid of two
test cases. The first case is designed to test the BRep update without any coupling to mechanics. To this end, we simply fracture all the facets in the mesh one by one in a random order. The procedure is applied to the fragmentation of a cube. In this case the BRep contains a single body bounded by one shell. The shell is partitioned into six facets bounded by six square loops. The loops are decomposed into twelve straight edges. It should be noted that, whereas the initial topology of the model is very simple, the process of random fragmentation rapidly adds considerable complexity to the BRep. The cube is meshed automatically into 324 ten-node tetrahedra, 650 nodes, and 550 facets. A blow-up of the fully fractured cube is shown in Fig. 27. At the end of the test, an admissible BRep consisting of 324 bodies is obtained. By repeating the test several times, it was verified the final BRep is independent of the order of traversal of the facets.

A second test of the algorithm concerns the simulation of the dynamic fragmentation of a cube under the action of suddenly applied tractions on opposite ends of the cube. The simple criterion [42,50,58]:

$$\sigma = \sqrt{\sigma_n^2 + \sigma_t^2/\beta^2} = \sigma_f,$$  \hspace{1cm} (1)

determines the onset of fracture in a facet. Here $\sigma_n$ denotes the normal traction to the facet, $\sigma_t$ the tangential traction, $\sigma_f$ plays the role of an effective normal traction, $\beta$ is a material parameter, and $\sigma_f$ is a fracture or spall stress. Once fracture is detected, the splitting of the facet is assumed to occur instantaneously. Evidently, this model is oversimplified in that the creation of new surface takes place at no cost in energy, i.e. the attendant fracture energy is zero. The model suffices, however, as a test of the interplay between mechanics and the BRep update procedure. More complete cohesive models of fracture may be found elsewhere [42,50,58].

The lateral dimension of the cube is $L = 1$ cm, and the average applied traction is 375 MPa. The remaining parameters employed in the calculations are: $\sigma_f = 1$ GPa, $\beta = 0.707$; Young's modulus $E = 201$ GPa, Poisson's ratio $\nu = 0.3$; mass density $\rho = 7830$ kg/m$^3$. The equations of motion are integrated in time by recourse to Newmark's algorithm with parameters $\beta = 0$, $\gamma = 1/2$ [68,69]. The time step used in the calculations is $\Delta t = 0.01$ ms.

The initial mesh consists of 118 nodes, 43 tetrahedra and 62 facets (Fig. 28(a)). Evidently, this mesh is too coarse to afford an accurate simulation of the fragmentation process, and the example is intended solely as a test on the BRep update procedure. Several succeeding stages of the fragmentation process are shown in Figs 28(b–e). The final configuration (Fig. 28(e)), is reached in 1000 time steps. As may be observed from this figure, the cube eventually splits into 43 one-tetrahedron bodies.

---

**Fig. 22.** This subroutine checks if the corner of the opening facet is the last connection between the two connected tetrahedra.

**Fig. 23.**

7. $S$

We will consider only the following special case: such that $S$ that $S$.
```c
void VerifyTetrahedra

/* Recursive procedure searching for tetrahedra and wingedfacet
   connected to original node ON. Changes ON in NN. */
for (i = 0; i < 4; i++) {
    WF = TE->Wingedfacet[i];
    for (j = 0; j < 6; j++) {
        if (WF->Node[j] == ON) {
            // Same node */
            WF->Node[j] = NN;
        }
    }
    for (k = 0; k < 2; k++) {
        if (WF->Tetrahedron[k] == TE) {
            TA = WF->Tetrahedron[k];
            if (TA == NULL) {
                PA = WF->racef;
                for (m = 0; m < 6; m++) {
                    if (PA->Node[m] == ON) {
                        // Check if the node is the same/
                        PA->Node[m] = NN;
                        UpdatedWingedSegment (PA);
                        continue;
                    }
                }
            } else {
                // Check adjacent tetrahedron */
                if (PA->Node[m] == ON) {
                    // Check if the node is the same/
                    PA->Node[m] = NN;
                    UpdatedWingedSegment (PA);
                }
            }
        }
    }
    VerifyTetrahedra (TA, NN, ON); /* Recursive call */
}
return;
```

Fig. 23. Traversing the wingedfacet structure, this subroutine updates the connectivity of the tetrahedra, faces, segments and wingedsegments that remain on the top side of the fractured facet.

7. Summary and Conclusions

We have classified and enumerated the ways in which the topology and geometry of a three-dimensional finite-element model may evolve as a consequence of fracture and fragmentation, and described such actions as may be taken to update the boundary representation of the solid so as to faithfully reflect that evolution. The focus of our approach is on situations involving a completely arbitrary topological and geometrical evolution of a three-dimensional solid, not necessarily restricted to an evolution of its surface. Solids are described by their boundary representation (BRep) and a surface and volume triangulation. Fracture processes are modeled by the introduction of cohesive elements at element interfaces [42,50,58].

In our implementation of the approach, all element
interfaces are regarded as potential fracture surfaces. However, contrary to other finite-element implementations of cohesive theories of fracture [48,49], which insert cohesive elements at every facet in the initial mesh, we insert cohesive elements adaptively upon the attainment of a fracture criterion. The insertion of a cohesive element at a newly fractured facet induces topological and geometrical changes in the model which need to be faithfully recorded in the BRep. The continuous update of the BRep in turn makes it possible to remesh the model. The proposed procedure enables the simulation of very
Fig. 25. Topological changes on loops induced by the fracturing of a facet. The primed numbers indicate the new inserted nodes: (a) Insertion of a new loop in a face; (b) Update of an existing loop on a face; (c) Joining two existing loops on a face; (d) Definition of a new face and insertion of a new loop.

Fig. 26. This recursive subroutine checks the connections between the two tetrahedra TB and TT adjacent to the fractured facet. The original adjacency between TB and TT has been already erased. The search is performed starting from the tetrahedron TT and traversing the adjacent tetrahedra; the search is successful if the tetrahedron TB is not reached.

Fig. 27. A random opening criterion applied to the fragmentation of a simple cube.

complex fracture processes, including crack nucleation, crack propagation, branching, surface cracks and the formation of fragments, as demonstrated by the test cases reported in the preceding section. A particularly appealing feature of the
method is that it does not presuppose a particular type of constitutive behavior of the bulk of the material, nor does it place restrictions on the extent of crack growth, or the size of the plastic zone. Cohesive theories of fracture have been successfully applied by Camacho and Ortiz [42,43] to the axisymmetric simulation of impact damage and ballistic penetration of light armor. The present work establishes the feasibility of cohesive models of fracture, as bear on the direct simulation of complex fracture phenomena, possibly leading to fragmentation, in three dimensions.

Acknowledgements

This research has been supported by the Office of Naval Research through grant N00014-95-1-0453, and by the Army Research Office through grant DAAH04-96-1-0056.

References

20. Schroeder, W. J.; Shepard, M. S. (1988) Geometry-based fully automatic mesh generation and the Delaun-
adaptive remeshing and evolving geometry, Computers and Structures, 30: 319–325
cal, numerical and experimental aspects of three dimensional fracture processes, ASME-AEDM, 91; 321–327
64. Dugdale, D. S. (1960) Yielding of steel sheets containing slits, Journal of the Mechanics and Physics of Solids, 8; 100–104