2
Cohesive Fracture and Fragmentation Simulation

2.1
Numerical representation of quasi-brittle dynamic fracture

The application of interest in this work is fracture in quasi-brittle materials. These materials feature some ductility in a region ahead of the crack tip that cannot be approximated by linear elastic fracture mechanics [53]. We simulate such problems using the cohesive zone model approach in which the fracture process zone ahead of the crack tip is approximated by a nonlinear traction-separation relation. This approach is attractive in its simplicity: the degrading and softening mechanisms, where micro-cracks and voids initiate and coalesce ahead of the crack tip, are not explicitly modeled; rather they are approximated by the cohesive zone [54, 55]. The concept is illustrated by the simple schematics shown in Figures 2.1 and 2.2. The macro crack tip contains zero tractions and complete separation, then ahead of this point the traction increases and opening decreases.

The cohesive zone model approach can be incorporated into a number of numerical frameworks. In this work we limit our attention to the inter-element approach, in which the cohesive elements are only present at the bulk finite element (or volumetric element) boundaries. We employ standard quadratic triangular elements in 2 dimensions and linear tetrahedrons in 3 dimensions to represent the continuum behavior. The cohesive elements are quadratic edge elements encoded with the traction-separation relationship, given by the PPR (potential-based cohesive zone model). We will not address the model in detail here, instead the reader is referred to [56] for the derivation of the model and to [57] for a comparison of the model to others available in the literature. The extrinsic model is utilized, which allows for arbitrary crack growth (so long as

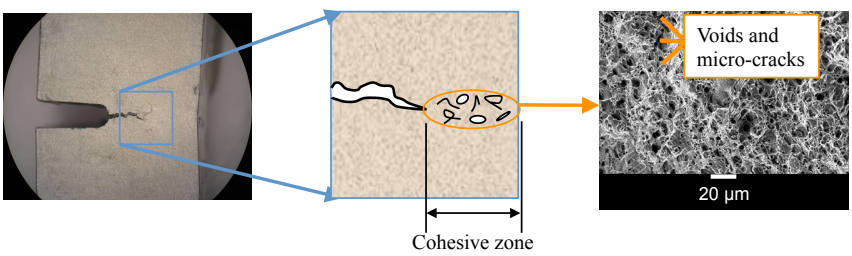

Figure 2.1 – Schematic of the cohesive zone model approach. The cohesive zone ahead of the macro crack tip consists of voids and micro-cracks,

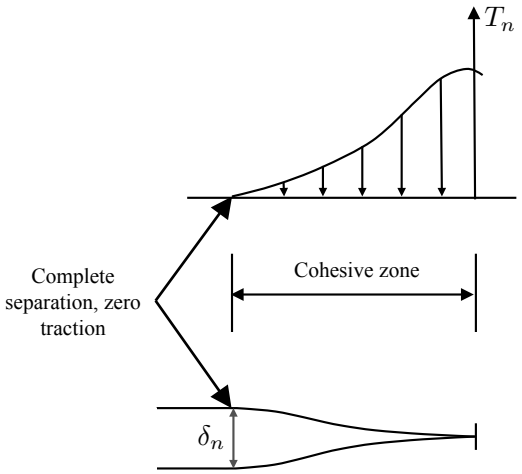

Figure 2.2 – The surfaces of the macro crack tip are completely separated and are traction free. Then separation decreases and traction increases into the cohesive zone.

it is along element boundaries) without restriction to predefined or preexisting fracture planes [58].

We consider temporal effects when modeling crack propagation through the explicit central difference time integration scheme with a lumped mass matrix [59]. This eliminates the need to solve the linear system, which makes the continuum problem without adaptivity readily parallelizable. However, in this work, the mesh evolves in time via insertion of cohesive elements, mesh refinement, and mesh coarsening; thus parallelization is not at all trivial. The details of the CPU simulation is described in the next section.

2.2 Simulation steps

The fracture phenomenum occurs in the finite element model when a load is applied to the mesh. The load can be a nodal displacement, or an imposed force or velocity. This leads to an increase in nodal stress, which leads to fractured facets. Fractured facets are checked for cohesive element insertion, which leads to fracture propagation in the model. We use the inter-element technique, where cohesive elements are inserted between bulk finite elements. They impose cohesive forces and are inserted in the mesh at simulation time.

We propose a simple but effective finite element mesh representation for fracture, microbranching, and fragmentation simulations. For a 2D simulation, we provide support for unstructured meshes of either linear (T3) or quadratic triangular elements (T6). For 3D simulation, we provide support unstructured tetrahedron (Tet4) meshes. Our data structure was designed for unstructured meshes. Figure 2.3 shows a typical representation for a T6 finite element mesh. The finite element types include nodes, facets, bulk elements, and cohesive

elements, which are explicitly represented as independent elements [60]. Our data structure implicitly represents facets corresponding to interfaces between two adjacent bulk elements. The intrinsic cohesive model assumes that all cohesive elements are embedded in the mesh before the simulation begins [61]. This leads to an unchanged mesh connectivity during the whole simulation process, but introduces an artificial reduction of stiffness. We adopt an extrinsic cohesive model, which assumes that separation between bulk elements only occurs when the interfacial traction reaches a finite strength [8, 61]. Challenges emerge when using an extrinsic model, since it requires an adaptive insertion of cohesive elements and topological changes of finite element mesh during the simulation process.

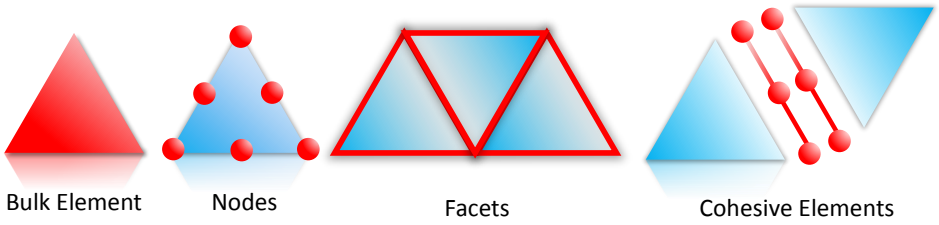

Figure 2.3 – T6 mesh attributes belonging to the simulation.

During simulation, internal, external, and cohesive forces at the nodes generate stresses along element interfaces, which may lead to fracture and fragmentation evolution. New nodes and cohesive elements are created whenever a facet fractures. Node attributes such as displacement, position, velocity, and acceleration are also updated from the internal, external, and cohesive forces. In order to obtain a precise and stable simulation, one must properly adjust parameters such as material properties and adopt small time steps, suitable for the explicit time integration scheme, together with a highly discretized model. Table 2.1 shows the algorithm for the fragmentation simulation. For simplification purposes, we will refer to the steps of a two-dimensional simulation, but the extension to 3D is analogue.

```
Compute Stiffness Matrix
Update Nodal Mass
current step ← 0
while current step <= maximum step do
    Update Displacements
    if current step == check step then
        Compute Stresses
        if stresses > stress threshold then
            Insert Cohesive Elements
            Update Nodal Masses
        end if
    end if
    Compute Internal Forces
    Compute Cohesive Forces
    Update Velocities and Accelerations
    Update Boundary Conditions
    current step = current step + 1
end while
```

Table 2.1 – Fragmentation algorithm

2.2.1 Pre-processing and updating

Given an initial decomposition of the domain, during the pre-processing phase, the stiffness matrix is calculated for each bulk element. We consider the stiffness matrix to remain constant during the whole simulation. Each lumped mass matrix is initialized before the simulation. The lumped mass matrix contains mass values relative to each bulk element node. Therefore, nodal masses are updated from the lumped mass matrix by going through the incident elements to each node. The lumped mass matrix has to be updated every time the mesh changes. This occurs when cohesive element insertion results from a fractured facet between two bulk elements.

Accelerations are computed from the cohesive and internal forces and nodal masses, which are then used to update the nodal velocities according to the following equations:

$$\begin{aligned} \mathbf{a}_{i+1} &= \frac{\mathbf{R}_{coh_i} - \mathbf{R}_{int_i}}{m_i} \\ \mathbf{v}_{i+1} &= \mathbf{v}_i + \frac{1}{2}(\mathbf{a}_i + \mathbf{a}_{i+1})\Delta t \end{aligned} \tag{2.1}$$

$$\mathbf{u}_{i+1} = \mathbf{u}_i + \mathbf{v}_i \Delta t + \frac{1}{2}\mathbf{a}_i \Delta t^2 \tag{2.2}$$

2.2.2 Stresses

The stresses computation is the most costly step of the simulation. After a certain number of steps (1 or 10 in this work), we compute the stress and strain at each bulk element node from their Gauss point evaluations using an extrapolation method. This whole procedure almost dominates the simulation time with excessive arithmetic operations and could be considered the bottleneck of the simulation loop if executed for all steps. To compute the stresses and strains at Gauss points of each bulk element (for each of the three Gauss points considered in this work) in 2D, we first obtain the shape functions and its derivatives, compute the Jacobian matrix and its inverse, and compute the strains and displacements relation matrix. Using the material properties of the element, the constitutive matrix is computed, followed by the stresses and strains at the Gauss points. In a 2-dimensional T6 mesh case, this is a 3x4 matrix, as shows below.

$$\boldsymbol{\sigma}_{G_{element}} = \begin{pmatrix} \sigma_{1,1} & \sigma_{1,2} & \sigma_{1,3} & \sigma_{1,4} \\ \sigma_{2,1} & \sigma_{2,2} & \sigma_{2,3} & \sigma_{2,4} \\ \sigma_{3,1} & \sigma_{3,2} & \sigma_{3,3} & \sigma_{3,4} \end{pmatrix}_{3\text{x}4} \tag{2.3}$$

By means of standard extrapolation, the stress and strain matrices are obtained using the previously computed stress and strain matrices at the Gauss points and the element shape functions ($\mathbf{N}$). Thus

$$\begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{yx} & \sigma_{yy} \end{pmatrix}_{\text{node i}} = \begin{pmatrix} N_{1,1} \\ N_{1,2} \\ N_{1,3} \end{pmatrix}^T_{\text{node i}} \begin{pmatrix} \sigma_{1,1} & \sigma_{1,2} & \sigma_{1,3} & \sigma_{1,4} \\ \sigma_{2,1} & \sigma_{2,2} & \sigma_{2,3} & \sigma_{2,4} \\ \sigma_{3,1} & \sigma_{3,2} & \sigma_{3,3} & \sigma_{3,4} \end{pmatrix}_{3\text{x}4} \tag{2.4}$$

The principal stresses and their directions are calculated with respect

to each nodal location. We determine if the principal stresses at each facet between two bulk elements exceed a limit for each of the nodes composing the element. Average stresses are computed to check for cohesive element insertion. We indicate that a facet is fractured if the stress exceeds a given threshold [8].

2.2.3 Insertion of cohesive elements

Insertion of cohesive elements imposes topological changes in the mesh [61]. After inserting the new cohesive element, each facet node is checked for duplication. Figure 2.4 illustrates a CPU algorithm for duplicating nodes on a triangular mesh. In 2D, the facet mid-side node must be duplicated in a T6 mesh. However, this assertion does not apply to the corner nodes, which must be checked by going through incident elements to which they belong. For each fractured facet, we verify if each of its corner nodes needs duplication. From a node, we traverse all its incident elements starting with one of the two adjacent elements the facet belongs to. If we reach the other adjacent element to the facet, then the node is not duplicated. However, if not reached, the node must be duplicated. The global node counter is incremented and the new node index is retrieved from it. Once again we must traverse the adjacent elements to update incidence with the new node index. Finally, the facet mid-side node is updated with the node index also retrieved from the node incremented global counter.

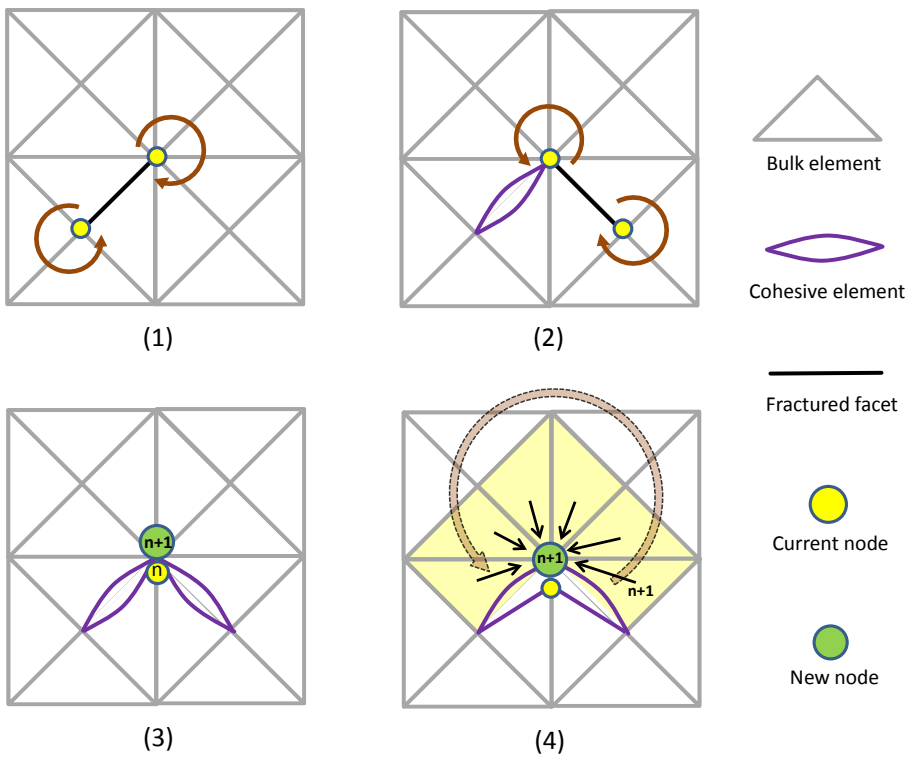

Figure 2.4 – Cohesive element insertion algorithm on a T3 mesh. (1) Mesh with initial facets that need to be fractured. Elements belonging to each node are traversed and cohesive element is inserted but no node is duplicated. (2, 3) The other fractured facet is checked for node duplication, the cohesive element is inserted and the node is marked as needing duplication. (4) Node is duplicated by traversing through the elements and updating the node index of the node belonging to them.

If there were new cohesive elements added to the mesh, the topological changes indicate that some nodal masses also changed since bulk elements loose adjacency relationship. Therefore, the nodal mass must be updated again like on the pre-processing phase. We then initialize the nodal internal, external, and cohesive forces for future computations.

2.2.4 Internal and cohesive forces

The nodal internal force computation is also computationally expensive, since it must be done every step and requires a large number of arithmetic operations. The internal force vector results from a product of the stiffness matrix and the element displacement vector containing displacements for its six nodes in a T6 mesh, as shown in Equation 2.5. This means we are multiplying a 12x12 matrix with a 12x1 vector, and it greatly reduces the performance of our parallel implementation due to its numerous global memory accesses.

$$R_{int_{12,12}} = \begin{pmatrix} k_{1,1} & k_{1,2} & \cdots & k_{1,12} \\ k_{2,1} & k_{2,2} & \cdots & k_{2,12} \\ \vdots & \vdots & \ddots & \vdots \\ k_{12,1} & k_{12,2} & \cdots & k_{12,12} \end{pmatrix} \begin{pmatrix} u_{1x} \\ u_{1y} \\ \vdots \\ u_{6y} \end{pmatrix} \tag{2.5}$$

The cohesive forces are then calculated by traversing through all cohesive elements and calculating their contributions to each node attached to them. The element vector of the deformed configuration is obtained, followed by the cohesive separations in the local coordinate system. Then, the separations and tractions at each Gauss point are calculated, together with the cohesive shape functions. Finally, the nodal cohesive force vector is obtained from cohesive tractions and shape functions. Together with the internal force and stresses, calculating the cohesive forces is one of the most costly computation steps within the simulation loop.