SoftwareX 7 (2018) 162–170

Contents lists available at ScienceDirect

SoftwareX

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

Original software publication

DEIP, discontinuous element insertion Program — Mesh generation for interfacial finite element modeling

Timothy J. Truster

Department of Civil and Environmental Engineering, University of Tennessee, Knoxville, 318 John D. Tickle Engineering Building, Knoxville, TN 37996, United States

ARTICLE INFO

Article history:
Received 3 November 2017
Received in revised form 10 May 2018
Accepted 14 May 2018

Keywords:
Interface elements
Cohesive zone models
Discontinuous Galerkin method
Mesh topology

ABSTRACT

The Discontinuous Element Insertion Program is a MATLAB/Octave toolbox for inserting zero-thickness interface elements into two and three dimensional finite element meshes. These interface elements, termed herein as "couplers", are used for intrinsic cohesive zone modeling and for the Discontinuous Galerkin method. The underlying algorithm is topology based and is suitable for complex, unstructured meshes of mixed-type linear and quadratic elements. Insertion is specified according to regions or subdomains within the overall analysis domain, a geometrically intuitive means to designate the coupler locations.

Code metadata

Current code version	DEIP v1.3
Permanent link to code/repository used of this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-17-00085
Legal Code License	University of Illinois/NCSA Open Source License (NCSA)
Code versioning system used	mercurial
Software code languages, tools, and services used	MATLAB
Compilation requirements, operating environments & dependencies	MATLAB 2014a –Present, Octave 3.8.1 –Present
If available Link to developer documentation/manual	http://clmi.utk.edu/software/
Support email for questions	ttruster@utk.edu

1. Motivation and Significance

The popularity of discontinuous formulations for computational solid mechanics has steadily increased in recent years. Physical applications relevant to fracture mechanics include fragmentation of brittle materials [1,2], delamination in composite materials [3–5], hydraulic fracturing [6,7], and grain boundary cracks in polycrystalline materials [8]. Similarly, the Discontinuous Galerkin (DG) method [9–12] has been applied in the solid mechanics field to efficiently model sharp gradient features and to enable mesh adaptivity. A common approach for numerically realizing these methods is using the so-called zero-thickness interface finite elements [13], whereby an element is created through node duplication such that the two sides of the interface initially coincide but subsequently may separate apart. Unfortunately, at the present time, standard commercial finite element codes do not contain mesh generation features for zero-thickness elements. Thus, the burden is placed on the user to create the modified mesh connectivity, which can be highly non-trivial for complex meshes in three dimensions.

Various researchers have proposed algorithms for generating zero-thickness elements for specialized applications. Some of the earliest examples were developed for adaptively inserting the elements during fragmentation simulations [1,14]. These algorithms support the extrinsic cohesive zone (CZ) modeling approach [15–17] "on-the-fly" based on element-local fracture criteria to track the propagation of cracks within a brittle material under impact loading. These algorithms have not yet been implemented in open-source software. Other methods, such as intrinsic cohesive zone models and Discontinuous Galerkin approaches, instead require all interface elements to be present from the start of a simulation at predetermined locations. These approaches could benefit from concurrent insertion algorithms where the list of interface facets are determined based on intuitive geometrical features of the physical domain. More recently, an algorithm and open-source code [18] have been developed for inserting zero-thickness

E-mail address: ttruster@utk.edu.

https://doi.org/10.1016/j.softx.2018.05.002

elements throughout the mesh as a pre-processing step, which has been applied to model grain boundary cracking, composite delamination, and matrix cracking [19] using a hybrid DG-CZ formulation [2]. However, that approach can be applied only when interface elements are required along all inter-region boundaries rather than a selection of them. Therefore, a method that provides the user with greater flexibility for choosing the location and type of interface elements is desirable.

In the current work, an open-source software package is developed from a general-purpose algorithm [20] for inserting zero-thickness interface elements, termed herein as "couplers", into specified regions of two and three dimensional conforming meshes. The term "coupler" is introduced since the algorithm is appropriate for the two common types of interface elements that have different connectivity templates: intrinsic cohesive zone elements and Discontinuous Galerkin elements. The distinguishing feature is that the user implicitly designates the locations for coupler insertion by (1) providing lists of elements within regions of the overall analysis domain and (2) naming the pairs of regions between which couplers should be inserted. By this geometrically intuitive means, couplers can be selectively inserted within specific regions or along specific interfaces. Also, different types of couplers as well as different material properties may be directly assigned to these particular sets, providing an intuitive means to complete the description of the interfacial-modified mesh for analysis purposes. The generation of these interfacial meshes is a key enabler of mechanical performance modeling of interface for debonding in fibrous composites [5,21] and grain boundary sliding and cavitation in polycrystalline metals [22,23].

2. Software description

The DEIP package contains a source directory with coupler insertion and finite element analysis scripts and an examples directory to demonstrate the modules. The files are compatible both with MATLAB and with Octave. An installation script loads the source directory into the user's search path, and a user-manual documents the usage of the package. The following sections describe the underlying algorithm and the concept of regions for designating the coupler locations as well as listing the modules and a typical workflow for using the package.

2.1. Topological definitions

The concept of *regions* that drives the insertion of *couplers* by DEIP is derived from the topology of a *domain* consisting of a conforming mesh of finite *elements* in two (2D) or three (3D) dimensional space [24]. Throughout the following discussions, topological entities are identified by *italic* typeset. Each *element* in the mesh is defined by a set of *nodes* which are points within the *domain* associated with their particular coordinates; see the 2D example in Fig. 1. In 2D, meshes containing a mixture of triangular and quadrilateral *elements* are considered, and in 3D, meshes containing tetrahedral, pyramid, wedge, or hexahedral *elements* are permitted. In Fig. 1, *nodes* are designated by numbers and *elements* are denoted by lower-case letters. For example, *element a* is composed of the three *nodes* 1, 4, and 5. The term *facet* refers either to an *edge* in 2D or a *face* in 3D on the boundary of an *element*.

In addition to the above standard features associated with finite element discretization, we define a *region* as a contiguous set of *elements* within the *domain*, which in general may form nonconvex subdomains and consist of spatially disjoint sets of *elements*. Each *element* in the *domain* is a member of exactly one *region*. In Fig. 1(a), the *regions* are denoted by capital letters, and the *elements* belonging to each *region* share the same color. Examples of *regions* in the context of finite element modeling include the fibers and the surrounding matrix in composites, where each *region* is considered to have different material properties. However, herein a *region* is a purely geometrical construct to enable completely general applications.

The *facets* of all the *elements* in the *domain* can be separated into three disjoint sets. The first set are those *facets* which lie on the *domain* boundary, which are adjacent to exactly one *element*. The second set are those which lie between *elements* of two different *regions*, which are said to belong to *interfaces*. This set is further divided according to pairs of *regions*, such that *interface(A,B)* is the set of all *facets* between *region A* and *region B*. The third set are those which lie between *elements* of the same *region*. Such *facets* belonging to *region C* will be denoted as *intraface(C)*, and so forth. *Interfaces* are shown as thick line segments in Fig. 1 while *intrafaces* are shown as thin line segments.

The last relevant topological concept defined herein is the *sector*. Two *elements* are defined to belong to a *sector* if the shared *facet* between them is not a *facet* designated to be cut by a *coupler* (either *interface* or *intraface*). A *sector* is then the largest set of *elements* satisfying this definition. For example, *elements b*, *c*, and *d* form a *sector* around *node* 5 in Fig. 1.

2.2. Coupler insertion algorithm

A topological-based algorithm is presented for inserting *couplers* along the *interfaces* or *intrafaces* in the *domain*. In the finite element literature, such computational entities are typically referred to as "zero-thickness elements" or "interface elements". Herein, we apply the term *coupler* to distinguish from the other topological definitions made in Section 2.1 and to provide for broader types of computational entities. Thus, a *coupler* is defined as a topological unit consisting of *nodes* from exactly two *elements* which are adjacent across either an *interface* or *intraface*. The *coupler* is generated by duplicating the *nodes* lying on the *facet* shared by the two *elements* to effectively split the *mesh* along that *facet*.

These *couplers* commonly appear as numerical realizations of discontinuous formulations for modeling PDEs. For example, to model the progressive debonding in composites, intrinsic cohesive zone models can be introduced as *interface couplers* between the *elements* of the fiber and matrix constituents. The *couplers* are present in the analysis from the initial stage in order to capture the initiation and progression of fracture at the interface through a traction-separation relation. Regarding the intrinsic cohesive zone models, the reader may consult [1,2,5,25] for mathematical aspects and [18,26] for notes on implementation.

Briefly, let Ω denote an open bounded *domain* with Lipschitz boundary and containing (possibly multiple and/or disjoint) *interfaces* Γ_I that partition Ω into a collection of *regions* Ω' such that the closure of Ω' equals to the closure of Ω. Furthermore, we denote the displacement field as $\boldsymbol{u}$, the linear elastic stress derived from the displacement as : $\boldsymbol{\sigma} = \mathbf{C} : \boldsymbol{\varepsilon}(\boldsymbol{u})$, the material moduli as $\mathbf{C}$, and the symmetric gradient operator as : $\boldsymbol{\varepsilon}$. Then, the weak form of intrinsic cohesive zone models is posed as finding $\boldsymbol{u} \in H_0^1(\Omega')$ such that for all $\boldsymbol{w} \in H_0^1(\Omega')$:

$$\int_{\Omega'} \boldsymbol{\varepsilon}(\boldsymbol{w}) : \mathbf{C} : \boldsymbol{\varepsilon}(\boldsymbol{u}) \, \mathrm{d}\Omega + \int_{\Gamma_I} [\![\boldsymbol{w}]\!] \cdot \boldsymbol{t}([\![\boldsymbol{u}]\!]) \, \mathrm{d}\Gamma = \int_{\Omega'} \boldsymbol{w} \cdot \boldsymbol{b} \, \mathrm{d}\Omega \tag{1}$$

where $\boldsymbol{b}$ is the body force, $\boldsymbol{t}([\![\boldsymbol{u}]\!])$ is the traction field defined from the displacement jump $[\![\boldsymbol{u}]\!]$ across the interface Γ_I within the domain, and $H_0^1(\Omega')$ is the standard Hilbertian-Sobolev space of functions satisfying homogeneous Dirichlet boundary conditions on the boundary of Ω.

Similarly, the formulation [9,11] and implementation [10,12] of the Discontinuous Galerkin method can be found in the indicated