Engineering Fracture Mechanics 126 (2014) 190–216

Contents lists available at ScienceDirect

Engineering Fracture Mechanics

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

A growing library of three-dimensional cohesive elements for use in ABAQUS

Daniel W. Spring, Glaucio H. Paulino [*]

Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, 205 North Mathews Ave., Urbana, IL 61801, United States

ARTICLE INFO

Article history:
Received 25 April 2013
Received in revised form 26 March 2014
Accepted 6 April 2014
Available online 18 April 2014

Keywords:
3D cohesive elements
ABAQUS
UEL
PPR Cohesive model
Finite deformation

ABSTRACT

In this paper, we present the implementation of a small library of three-dimensional cohesive elements. The elements are formatted as user-defined elements, for compatibility with the commercial finite element software ABAQUS. The PPR, potential-based traction–separation relation is chosen to describe the element's constitutive model. The intrinsic cohesive formulation is outlined due to its compatibility with the standard, implicit finite element framework present in ABAQUS. The implementation of the cohesive elements is described, along with instructions on how to incorporate the elements into a finite element mesh. Specific areas of the user-defined elements, in which the user may wish to modify the code to meet specific research needs, are highlighted. Numerical examples are provided which display the capabilities of the elements in both small deformation and finite deformation regimes. A sample element source code is provided in an appendix, and the source codes of the elements are supplied through the website of the research group of the authors.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

The use of cohesive elements within the framework of the finite element method has proven to be a powerful tool to model the fracture and fragmentation of materials. The concept of the cohesive zone model was presented over half a century ago by Dugdale [1] and Barenblatt [2]. They proposed modeling the inelastic zone in front of a macrocrack with a traction–separation relationship. Thus, as the crack separates, a softening traction is applied to the surrounding bulk material. There have been a variety of traction–separation relations proposed, including linear, bilinear, trilinear, trapezoidal, polynomial, and exponential softening models. A review of some of the significant traction–separation relations can be found in the work by Park and Paulino [3].

When using cohesive elements, the material model chosen for the bulk elements is independent of that chosen for the cohesive elements. For example, the bulk elements may be linear elastic, viscoelastic, hyperelastic, etc. In this publication we will present examples which use both linear elastic and hyperelastic material models. For the hyperelastic material, we use the Neo-Hookean material model; with corresponding stored-energy function, W [4]:

$$W(\mathbf{F}) = \frac{\mu}{2}[I_1 - 3]. \tag{1}$$

* Corresponding author. Tel.: +1 2177218422.
E-mail addresses: spring2@illinois.edu (D.W. Spring), paulino@illinois.edu (G.H. Paulino).

http://dx.doi.org/10.1016/j.engfracmech.2014.04.004
0013-7944/© 2014 Elsevier Ltd. All rights reserved.

Nomenclature

$\mathbf{B}$	global displacement–separation relation matrix
$\mathbf{D}_{local}$	local tangent stiffness matrix of the cohesive zone model
$[\mathbf{F}]_{el}$	internal force vector of a cohesive surface element
$[\mathbf{K}]_{el}$	tangent matrix of a cohesive surface element
$m,\ n$	non-dimensional exponents in the PPR model
N_i	cohesive element shape functions
$\mathbf{R}$	rotational matrix of nodal displacements
$\mathbf{T}_c$	cohesive traction vector
$\mathbf{T}_{ext}$	external traction vector
$T_n,\ T_t$	normal and tangential cohesive tractions
$T_n^v,\ T_t^v$	normal and tangential cohesive tractions for the unloading/reloading relation
$\delta\mathbf{u}$	virtual displacements
$\alpha,\ \beta$	shape parameters in the PPR model
$\alpha_v,\ \beta_v$	shape parameters in the unloading/reloading relation
Γ	boundary of external traction
Γ_c	boundary of cohesive fracture surface
$\Gamma_n,\ \Gamma_t$	energy constants in the PPR model
$\delta_n,\ \delta_t$	normal and tangential final crack opening widths
$\bar{\delta}_n,\ \bar{\delta}_t$	conjugate normal and tangential final crack opening widths
$\Delta_n,\ \Delta_t$	normal and tangential separations along the fracture surface
$\Delta_{n_{max}},\ \Delta_{t_{max}}$	maximum normal and tangential separations along the fracture surface
$\lambda_n,\ \lambda_t$	initial slope indicators in the PPR model
$\mathbf{E}$	Green strain
$\mathbf{S}$	2nd Piola–Kirchhoff stress
$\sigma_{max},\ \tau_{max}$	normal and tangential cohesive strengths
$\phi_n,\ \phi_t$	normal and tangential fracture energies
$\boldsymbol{\Psi}$	potential function for cohesive strength
Ω	domain
$\langle\cdot\rangle$	Macauley bracket
$\mathbf{t}_{local}$	local force vector of the cohesive model
$\mathbf{I}$	identity matrix
$\bar{\mathbf{n}}$	vector normal to the cohesive element midplane
$\bar{\mathbf{t}}_1,\ \bar{\mathbf{t}}_2$	vectors tangent to the element midplane
$\boldsymbol{D}_{nn},\ \boldsymbol{D}_{nt},\ \boldsymbol{D}_{tn},\ \boldsymbol{D}_{tt}$	components of the local tangent stiffness matrix
$\Delta_1,\ \Delta_2,\ \Delta_3$	opening components of cohesive element
$\boldsymbol{D}_{nn}^v,\ \boldsymbol{D}_{nt}^v,\ \boldsymbol{D}_{tn}^v,\ \boldsymbol{D}_{tt}^v$	components of the local tangent stiffness matrix for unloading/reloading
F	deformation gradient
μ	material shear modulus
$\mathbf{I}_1$	first invariant of the deformation gradient $(\mathbf{F}\cdot\mathbf{F})$
$\delta\boldsymbol{\Delta}$	virtual separation of cohesive element

where $\mathbf{F}$ is the deformation gradient, μ is the initial shear modulus of the rubber, and $I_1 = \mathbf{F}\cdot\mathbf{F}$ is the first principal invariant of the deformation gradient. Since the two constitutive models (bulk and cohesive) are independent, the modeling of functionally graded materials will require modifications to both the bulk and cohesive formulations. A means for modifying cohesive element formulations to allow for graded properties will be presented.

Currently, the built-in traction–separation relations for cohesive elements in the commercial software ABAQUS [5] have limitations. The available traction–separation relations typically consist of a linear hardening region, and either a linear or exponential softening region. The traction–separation relationship used in this paper is the consistent, potential-based model presented by Park et al. [6] in 2009. This model has been implemented in the Warp3D software [7], and in the Finite Element All-Wheel Drive (FEAWD) software [8–10], but it's use in ABAQUS has been limited to two-dimensional fracture problems. The intrinsic model will be presented here, the extended details of the model, and the extension of the model to an extrinsic formulation may be found in the principal publication. This work is an extension of a previous publication; in which the authors present the two-dimensional user defined element (UEL) for use in ABAQUS [11]. The requests, from users of that UEL, for the corresponding extension to three-dimensions, has been the motivation behind the present publication.

This paper aims to provide an educational look at the implementation of cohesive elements into a finite element mesh, in accordance with their use in ABAQUS or a similar commercial software with support for user-supplied subroutines. In the following section, we present related work and discuss potential applications of the cohesive model. In Section 3 the formulation of the three-dimensional UEL is outlined, and its use in ABAQUS explained. In Section 4, the intrinsic formulation of the Park–Paulino–Roesler (PPR) cohesive model is presented. It is understood that research, by its very nature, is diverse and ever changing. In order to aid in the removal of the black-box nature of many codes, and to make this work more adaptable to its user, Section 5 describes areas of the code which may be of interest to the readers, to modify and incorporate into their own work. Section 6 presents some relevant examples which make use of the 3D elements. Finally, a discussion of the work is presented, and a representative implementation (interface cohesive elements between linear tetrahedral bulk elements) is included in an Appendix.

2. Related work

Since its conception, the cohesive zone model has been used to describe the fracture and fragmentation of a variety of complex problems. Cohesive elements have been used extensively to model the fracture of concrete and asphalt [12]. In 1976, Hillerborg et al. [13] used cohesive elements with a linear softening relationship to study the fracture of plain concrete beams. Following the work of Hillerborg, Petersson [14] implemented a bilinear softening model to simulate the fracture of concrete and other quasi-brittle materials. More recently, Song et al. [15–17] used intrinsic cohesive elements to simulate crack propagation in asphalt concrete beams. They tailored both bilinear softening and exponential softening relations to asphalt to capture the behavior of the beams. Spring [18] used a graded bilinear softening relationship to investigate the failure of functionally graded concrete slabs. Park et al. [19] developed a trilinear softening relationship to model the fracture of functionally graded, fiber reinforced concrete beams. By accounting for the added fracture energy due to the inclusion of fibers, they displayed excellent agreement between numerical and experimental results.

There has also been a substantial amount of work done that uses cohesive elements in the simulation of impact damage and fragmentation. For instance, Camacho and Ortiz [20] employed a linear softening relation to model the pervasive fracture behavior which occurs during impact damage in brittle materials. They study the problem of a steel pellet impacting a ceramic plate (an important problem in armor and turbine blade design), and capture results consistent with experiments. In 1998, Espinosa et al. [21] combined cohesive elements with a continuum damage model to capture the fragmentation of ceramic rods under dynamic impact. Pandolfi et al. [22] used cohesive elements to model the fragmentation of radially loaded, expanding rings. More recently, Mota et al. [23] studied the fragmentation of a human cranium due to a firearm injury. They used cohesive elements with a linear softening relation, combined with three-dimensional finite element simulations, to capture behavior observed experimentally. In 2010, Caballero and Molinari used the same model as Camacho and Ortiz to study the fracture and fragmentation of kidney stones under impact. Their findings lead to the design of a new surgical tool to optimize the efficiency of kidney stone dissipation [24].

In general, the numerical simulation of dynamic crack propagation has been a popular application for cohesive elements. In 1999, Ortiz and Pandolfi [25] used effective-displacement three-dimensional cohesive elements to model dynamic fracture. They considered quadratic cohesive elements with an irreversible cohesive law, and were able to capture the correct crack trajectory in a drop-weight dynamic fracture test. Ruiz et al. [26] looked at the failure of a three-point-bend concrete beam under dynamic loading. They simulated the fracture using three-dimensional cohesive elements with a linear softening relation. More recently, Zhang and Paulino [27,28] were able to capture the microbranching phenomenon in dynamic fracture simulations using a linear softening relationship, they studied materials with both homogeneous and graded properties. In fracture simulations, particularly dynamic fracture simulations, it is important to take into consideration the effect of finite deformations. When cohesive elements are inserted into a model they initially have zero thickness, but once fracture progresses they separate and rotate. By not taking into consideration the rotation of these fracture surfaces, the physics of the problem may not be accurately captured.

3. UEL formulation

Three different cohesive element formulations are developed. As illustrated in Fig. 1, the three elements allow for compatibility with both linear brick and linear tetrahedral bulk elements, as well as with quadratic tetrahedral bulk elements.[1] As discussed previously, the material model used for the bulk elements is independent of that used for cohesive elements.

Intrinsic cohesive elements are inserted *a priori* into a finite element mesh. Initially, the intrinsic cohesive elements have zero thickness and, upon separation, impart a traction to the adjacent bulk elements. There are two common approaches to inserting cohesive elements into a mesh. One may either insert the cohesive elements along a pre-selected fracture path, restricting the crack to propagate where the user has specified, or one may insert cohesive elements between all bulk elements in a region of a mesh, allowing fracture to propagate freely within that region. When cohesive elements are inserted throughout a region of a mesh, the elements may introduce an artificial softening into the overall model, effectively distorting the bulk material properties [29,30]. In addition, if cohesive elements are inserted between all bulk elements in a region

[1] It is important to note that, at the time of publication, ABAQUS does not have a built-in cohesive element that is compatible with quadratic elements.

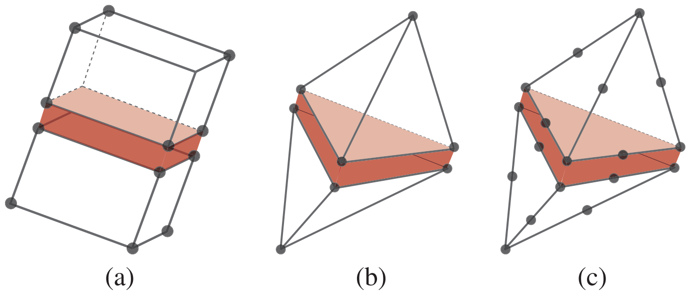


Fig. 1. Cohesive elements compatible with (a) linear brick, (b) linear tetrahedral and (c) quadratic tetrahedral elements. Cohesive elements, highlighted in red, are of initially zero thickness. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

of the mesh, the solution becomes highly non-convergent, and additional numerical measures need to be included in the solution scheme to account for these instabilities [31,32]. An alternative method for allowing fracture to propagate freely would be to use extrinsic cohesive elements [25,33].

Prior to insertion of cohesive elements, the standard finite element formulation may be described mathematically, using the principle of virtual work [34]:

$$W_{int} = \int_{\Omega} \delta\mathbf{E} : \mathbf{S}\, dV = \int_{\Gamma} \delta\mathbf{u} \cdot \mathbf{T}_{ext}\, d\Gamma = W_{ext} \tag{2}$$

where $\mathbf{E}$ is the Green strain tensor in the domain Ω and $\delta\mathbf{u}$ is the virtual displacement on the boundary Γ. Moreover, $\mathbf{T}_{ext}$ denotes the external traction, and $\mathbf{S}$ is the 2nd Piola–Kirchhoff stress tensor. When cohesive elements are included in the formulation, they contribute an additional term to the internal work:

$$\int_{\Omega} \delta\mathbf{E} : \mathbf{S}\, dV + \int_{\Gamma_c} \delta\boldsymbol{\Delta} \cdot \mathbf{T}_c\, d\Gamma_c = \int_{\Gamma} \delta\mathbf{u} \cdot \mathbf{T}_{ext}\, d\Gamma \tag{3}$$

where $\mathbf{T}_c$ is the cohesive traction, along the fracture surface Γ_c, corresponding to the virtual separation $\delta\boldsymbol{\Delta}$. Using the Galerkin method [34], we discretize the contribution from the cohesive elements:

$$[\mathbf{K}]_{el} = \int_0^1 \int_0^1 \mathbf{B}^T\mathbf{R}^T\mathbf{D}_{local}\mathbf{R}\mathbf{B}J\, d\xi\, d\eta \tag{4}$$

$$[\mathbf{F}]_{el} = \int_0^1 \int_0^1 \mathbf{B}^T\mathbf{R}^T\mathbf{t}_{local}\, J\, d\xi\, d\eta \tag{5}$$

where (ξ, η) denote intrinsic coordinates and J the Jacobian. In order to accurately incorporate a user defined cohesive element into the existing ABAQUS framework, we must provide the element stiffness matrix ($[\mathbf{K}]_{el}$) and force vector ($[\mathbf{F}]_{el}$) within the user-defined element (UEL). The global displacement–separation matrix, $\mathbf{B}$, is a $3 \times N$ matrix (where N is the number of shape functions) that computes the relative opening of the crack at any point in the cohesive element. The rotational matrix, $\mathbf{R}$, transforms from global to local coordinates. The local constitutive matrix, $\mathbf{D}_{local}$, and the local force vector, $\mathbf{t}_{local}$, are functions of the particular choice of cohesive model, and will be outlined below for the PPR cohesive model.

The $\mathbf{B}$ matrix is computed from the shape functions N_i $(i = 1, 2, \ldots N)$, and the identity matrix $\mathbf{I}$:

$$\mathbf{B} = [N_1\mathbf{I}_{3\times3} | N_2\mathbf{I}_{3\times3} | \cdots | N_N\mathbf{I}_{3\times3}][\mathbf{I}_{3N\times3N} | -\mathbf{I}_{3N\times3N}]. \tag{6}$$

The rotational matrix, $\mathbf{R}$, is a function of the normal vector ($\bar{\mathbf{n}}$) and two perpendicular tangential vectors ($\bar{\mathbf{t}}_2, \bar{\mathbf{t}}_3$) which form the basis of the midplane of the cohesive element, as illustrated in Fig. 2.

$$\mathbf{R} = \begin{bmatrix} \bar{\mathbf{n}}^T \\ \bar{\mathbf{t}}_2^T \\ \bar{\mathbf{t}}_3^T \end{bmatrix}. \tag{7}$$

The local constitutive matrix, specific to the PPR model, is defined as:

$$\mathbf{D}_{local} = \begin{bmatrix} D_{nn} & D_{nt}\Delta_2/\Delta_t & D_{nt}\Delta_3/\Delta_t \\ D_{tn}\Delta_3/\Delta_t & D_{tt}\Delta_2^2/\Delta_t^2 + T_t\Delta_3^2\Delta_t^3 & D_{tt}\Delta_2\Delta_3/\Delta_t^2 - T_t\Delta_2\Delta_3/\Delta_t^3 \\ D_{tn}\Delta_3/\Delta_t & D_{tt}\Delta_2\Delta_3/\Delta_t^2 - T_t\Delta_2\Delta_3/\Delta_t^3 & D_{tt}\Delta_3^2/\Delta_t^2 + T_t\Delta_2^2/\Delta_t^3 \end{bmatrix} \tag{8}$$

where D_{nn}, D_{nt}, D_{tn}, and D_{tt} are derived from the PPR model and are included in Appendix A. The variables Δ_2 and Δ_3 correspond to the crack opening widths in the plane perpendicular to the normal direction (Δ_1), as illustrated in Fig. 3. The local force vector, $\mathbf{t}_{local}$, for the PPR model is:

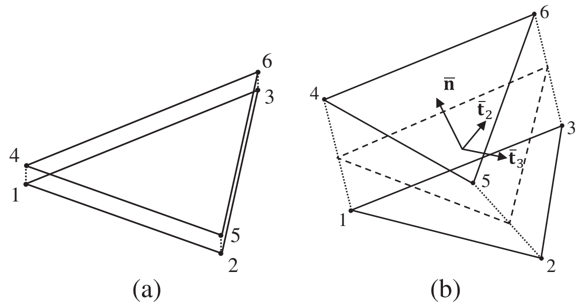


Fig. 2. A cohesive element in (a) its initial configuration with zero thickness and (b) its deformed configuration, depicting the normal and tangential vectors on the midplane of the element.

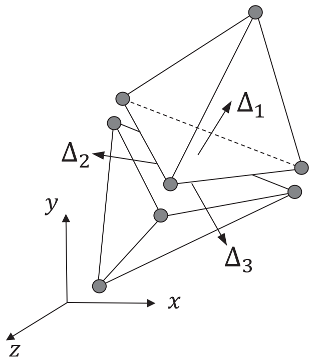


Fig. 3. Definition of opening directions in a typical element configuration.

$$\mathbf{t}_{local} = \begin{Bmatrix} T_n \\ T_t\Delta_2/\Delta_t \\ T_t\Delta_3/\Delta_t \end{Bmatrix} \tag{9}$$

where we have considered the tangential opening directions to be coupled through the relationship $\Delta_t = \sqrt{\Delta_2^2 + \Delta_3^2}$, on the fracture plane. The traction components (T_n and T_t) are derived from the cohesive potential and will be defined for the PPR model in Section 4.1.

4. Cohesive model

The cohesive model, chosen independent of the bulk model, determines the cohesive behavior in the inelastic region in front of the crack tip. There have been many models proposed (see [3] for a recent review of some of the prominent models), most of which suffer from deficiencies when trying to satisfy boundary conditions or which require non-physical input parameters. The PPR model is an adaptable cohesive model that was designed with the goal of overcoming the deficiencies suffered by previous models. The PPR model is able to capture both intrinsic and extrinsic fracture behavior [6], but the present implementation in ABAQUS focuses on the use of the intrinsic model. The extrinsic model is outlined in the principal publication [6]. Below we outline the formulation of the intrinsic softening model, as well as the choice of unloading/reloading and contact relations used in this work.

4.1. Intrinsic PPR model

The PPR model is potential-based. Such models have an advantage over non-potential-based models, in that their traction–separation relations are determined by taking the derivative of the potential with respect to the normal and tangential opening displacements. Similarly, the second derivative of the potential provides the consistent (material) tangent matrix. The potential for the PPR model is described as:

$$\boldsymbol{\Psi}(\Delta_n, \Delta_t) = min(\phi_n, \phi_t) + \left[\Gamma_n\left(1 - \frac{\Delta_n}{\delta_n}\right)^{\alpha}\left(\frac{m}{\alpha} + \frac{\Delta_n}{\delta_n}\right)^{m} + \langle\phi_n - \phi_t\rangle\right]\left[\Gamma_t\left(1 - \frac{|\Delta_t|}{\delta_t}\right)^{\beta}\left(\frac{n}{\beta} + \frac{|\Delta_t|}{\delta_t}\right)^{n} + \langle\phi_t - \phi_n\rangle\right] \tag{10}$$

where Δ_n and Δ_t are the normal and tangential opening tractions, respectively. There are eight user inputs to the model, a feature that allows for a very adaptable model that can capture a variety of responses. The eight inputs are: normal and tangential fracture energies (ϕ_n, ϕ_t), normal and tangential cohesive strengths $(\sigma_{max}, \tau_{max})$, normal and tangential shape parameters (α, β), and normal and tangential initial slope indicators (λ_n, λ_t). All parameters have physical interpretation. The shape parameters control the softening slope of the traction–separation relations. If α or β is set equal to 2, the relation is almost linear, whereas if they are less than or greater than 2, the relation is concave or convex, respectively. The initial slope indicators correspond to the ratio of the peak crack opening width to the final crack opening width. Thus, the smaller the initial slope indicator, the greater the initial slope in the traction–separation relation. The potential is defined over a domain of dependence; which is bounded by the normal final crack opening (δ_n) and the tangential final crack opening (δ_t):

$$\delta_n = \frac{\phi_n}{\sigma_{max}}\alpha\lambda_n(1-\lambda_n)^{\alpha-1}\left(\frac{\alpha}{m}+1\right)\left(\frac{\alpha}{m}\lambda_n+1\right)^{m-1}, \quad \delta_t = \frac{\phi_t}{\tau_{max}}\beta\lambda_t(1-\lambda_t)^{\beta-1}\left(\frac{\beta}{n}+1\right)\left(\frac{\beta}{n}\lambda_t+1\right)^{n-1}. \tag{11}$$

The corresponding traction–separation relations, T_n and T_t, determined by taking the derivative of the above potential with respect to Δ_n and Δ_t respectively, are described as:

$$\begin{aligned} T_n(\Delta_n,\Delta_t) &= \frac{\partial \boldsymbol{\Psi}}{\partial \Delta_n} \\ &= \frac{\Gamma_n}{\delta_n}\left[m\left(1-\frac{\Delta_n}{\delta_n}\right)^{\alpha}\left(\frac{m}{\alpha}+\frac{\Delta_n}{\delta_n}\right)^{m-1} - \alpha\left(1-\frac{\Delta_n}{\delta_n}\right)^{\alpha-1}\left(\frac{m}{\alpha}+\frac{\Delta_n}{\delta_n}\right)^{m}\right] \\ &\quad \times \left[\Gamma_t\left(1-\frac{|\Delta_t|}{\delta_t}\right)^{\beta}\left(\frac{n}{\beta}+\frac{|\Delta_t|}{\delta_t}\right)^{n} + \langle \phi_t - \phi_n\rangle\right], \end{aligned} \tag{12}$$

$$\begin{aligned} T_t(\Delta_n,\Delta_t) &= \frac{\partial \boldsymbol{\Psi}}{\partial \Delta_t} \\ &= \frac{\Gamma_t}{\delta_t}\left[n\left(1-\frac{|\Delta_t|}{\delta_t}\right)^{\beta}\left(\frac{n}{\beta}+\frac{|\Delta_t|}{\delta_t}\right)^{n-1} - \beta\left(1-\frac{|\Delta_t|}{\delta_t}\right)^{\beta-1}\left(\frac{n}{\beta}+\frac{|\Delta_t|}{\delta_t}\right)^{n}\right] \\ &\quad \times \left[\Gamma_n\left(1-\frac{\Delta_n}{\delta_n}\right)^{\alpha}\left(\frac{m}{\alpha}+\frac{\Delta_n}{\delta_n}\right)^{m} + \langle \phi_n - \phi_t\rangle\right]\frac{\Delta_t}{|\Delta_t|}. \end{aligned} \tag{13}$$

These traction components are used to determine the local force vector, as per Expression (9). Representative traction–separation relations are plotted in Fig. 4.

The energy constants Γ_n and Γ_t are related to the normal and tangential fracture energies. When the normal and tangential fracture energies are different $(\phi_n \neq \phi_t)$, the energy constants become:

$$\Gamma_n = (-\phi_n)^{\langle \phi_n-\phi_t\rangle/(\phi_n-\phi_t)}\left(\frac{\alpha}{m}\right)^{m}, \quad \Gamma_t = (-\phi_t)^{\langle \phi_t-\phi_n\rangle/(\phi_t-\phi_n)}\left(\frac{\beta}{n}\right)^{n}. \tag{14}$$

When the normal and tangential fracture energies are equal $(\phi_n = \phi_t)$, the energy constants become:

$$\Gamma_n = -\phi\left(\frac{\alpha}{m}\right)^{m}, \quad \Gamma_t = \left(\frac{\beta}{n}\right)^{n} \tag{15}$$

where the non-dimensional exponents, m and n, are evaluated from the shape parameters (α, β) and the initial slope indicators (λ_n, λ_t):

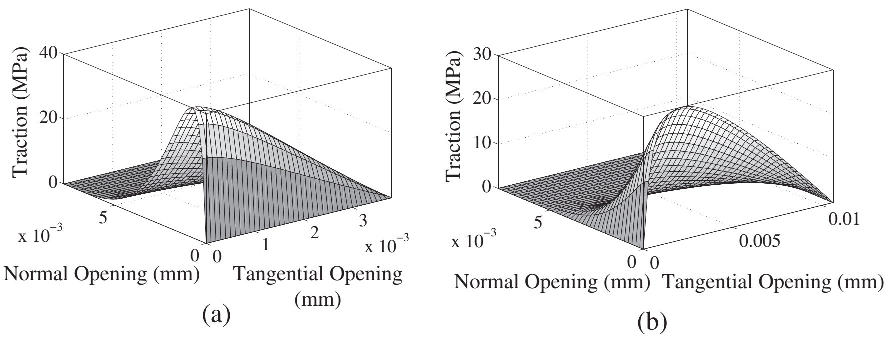


Fig. 4. Traction separation relations for (a) normal opening ($\phi_n = 100$ N/m, $\sigma_{max} = 40$ MPa, $\alpha = 5.0$, and $\lambda_n = 0.1$), and (b) tangential opening ($\phi_t = 200$ N/m, $\sigma_{max} = 30$ MPa, $\beta = 2.0$, and $\lambda_t = 0.2$).

$$m = \frac{\alpha(\alpha-1)\lambda_n^2}{(1-\alpha\lambda_n^2)}, \quad n = \frac{\beta(\beta-1)\lambda_t^2}{(1-\beta\lambda_t^2)}. \tag{16}$$

The traction–separation relations are valid in a domain of dependence. The domain for the normal traction is bounded by the final crack opening width in the normal direction (δ_n) and the conjugate final crack opening width in the tangential direction ($\pm\bar{\delta}_t$). Similarly, the domain for the tangential traction is bounded by the final crack opening width in the tangential direction ($\pm\delta_t$) and the conjugate final crack opening width in the normal direction ($\bar{\delta}_n$). The conjugate final crack opening width in the tangential direction ($\Delta_t = \bar{\delta}_t$) is the solution of the following nonlinear equation:

$$f_t(\Delta_t) = \Gamma_t\left(1-\frac{\Delta_t}{\delta_t}\right)^{\beta}\left(\frac{n}{\beta}+\frac{\Delta_t}{\delta_t}\right)^{n} + \langle\phi_t - \phi_n\rangle = 0; \tag{17}$$

which is unique between 0 and δ_t. Likewise, the conjugate final crack opening width in the normal direction ($\Delta_n = \bar{\delta}_n$) is the solution of the following nonlinear equation:

$$f_n(\Delta_n) = \Gamma_n\left(1-\frac{\Delta_n}{\delta_n}\right)^{\alpha}\left(\frac{m}{\alpha}+\frac{\Delta_n}{\delta_n}\right)^{m} + \langle\phi_n - \phi_t\rangle = 0; \tag{18}$$

which is unique between 0 and δ_n. The resulting domains of dependence are illustrated in Fig. 5.

In order to implement these elements into ABAQUS, we also need to consider the consistent tangent matrix, $\mathbf{D}_{local}$. This is calculated by either taking the second derivatives of the potential function (10), or the first derivatives of the traction relations (12) and (13), with respect to the normal and tangential separations (Δ_n and Δ_t). The components of the consistent tangent matrix are combined to form the local constitutive matrix through Expression (8), and are given in Appendix A.

4.2. Choice of unloading/reloading relation

The unloading/reloading relationship used in this model is uncoupled, in the sense that the unloading in the normal direction is viewed as independent of that in the tangential direction. The unloading relationship is activated when the normal or tangential separation is past the peak cohesive strength of the element, and effects both the traction vector and the tangent matrix. The current unloading/reloading relationship in the model is linear to the origin, as illustrated in Fig. 6. This is a very common approach to modeling unloading/reloading in cohesive elements [35,36], and will likely satisfy the expectations of the user. If an alternative unloading/reloading relationship is desired, we outline a possible replacement in Section 5.2 and specify the portions of the UEL which will need to be updated.

4.3. Choice of contact formulation

The contact formulation chosen for this work is based on the penalty stiffness approach. As the normal separation becomes negative, the resisting force increases linearly in accordance with a corresponding stiffness. The modulus of this stiffness is chosen to correspond to the slope of the hardening curve as it approaches zero opening displacement. Alternatively, other contact formulations could be used, such as the ones found in references [37,38]. We will discuss the modifications necessary to alter the contact formulation in Section 5.4.

5. Implementation and user modifications

In order to make this work as useful as possible, in this section we first describe how to implement the cohesive elements into a model, then we outline the sections in the code that the users may chose to modify, to suit their specific needs. This section provides the basis for other modifications (not anticipated in this paper).

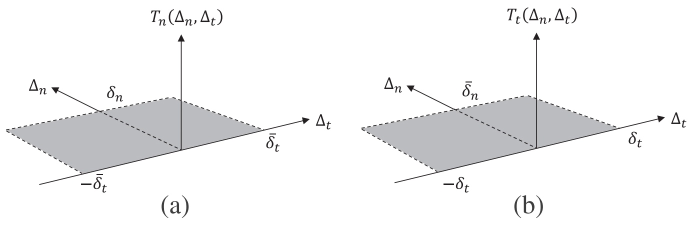
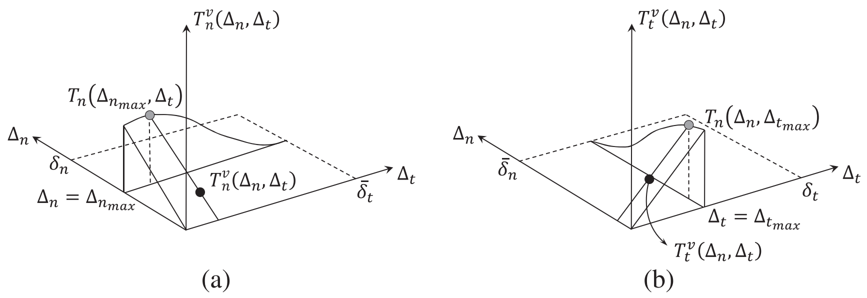


Fig. 5. Domain of dependence for each cohesive interaction (T_n, T_t), bounded by the final crack opening widths (δ_n, δ_t) and the conjugate final crack opening widths ($\bar{\delta}_n, \bar{\delta}_t$).



$T_n^v(\Delta_n, \Delta_t)$
$T_n(\Delta_{n_{max}}, \Delta_t)$
Δ_n
δ_n
$\Delta_n = \Delta_{n_{max}}$
$T_n^v(\Delta_n, \Delta_t)$
$\bar{\delta}_t$
Δ_t
(a)
$T_t^v(\Delta_n, \Delta_t)$
$T_n(\Delta_n, \Delta_{t_{max}})$
Δ_n
$\bar{\delta}_n$
δ_t
Δ_t
$\Delta_t = \Delta_{t_{max}}$
$T_t^v(\Delta_n, \Delta_t)$
(b)

Fig. 6. Depiction of the linear unloading/reloading relations for the (a) normal traction and (b) tangential traction.

5.1. Implementing elements into an ABAQUS input file

In order to insert the cohesive zone element (CZE) into a finite element mesh, the input file should refer to the CZEs using the following commands. In the case of the CZE with 8 nodes:

```
*USER ELEMENT, TYPE = U1, NODE = 8, COORDINATES = 3, PROPERTIES = 8, VARIABLES = 14
1, 2, 3
*ELEMENT, TYPE = U1, ELSET = ELSET_NAME
```

where the `VARIABLES` parameter indicates the number of solution-dependent state variables which are stored in each element [5]. For our purposes, these variables track the maximum opening at each Gauss point in the element to determine if the element is opening or closing. The main difference, for the other elements, is the number of nodes and variables. For the CZE with 6 nodes; `NODE = 6` and `VARIABLES = 8`, and for the CZE with 12 nodes; `NODE = 12` and `VARIABLES = 8`.

Below these input lines, the user specifies the element number and the node numbers corresponding to these elements. Nodes are numbered in the order shown in Fig. 7. To assign properties to the elements, one needs to include the following lines to the input file:

```
*UEL PROPERTY, ELSET = ELSET_NAME
100, 200, 30, 40, 2, 5, 0.1, 0.2
```

where we have chosen, for example, $\phi_n = 100$ N/m, $\phi_t = 200$ N/m, $\sigma_{max} = 30$ MPa, $\tau_{max} = 40$ MPa, $\alpha = 2$, $\beta = 5$, $\lambda_n = 0.1$ and $\lambda_t = 0.2$, respectively. Note that the units are important, and the values here assume that the dimensions of the mesh are given in meters, and any force boundary conditions are given in Newtons. Finally, when running this problem on a linux machine, the following command line prompt will run the problem:

```
abaqus job = job_name user = 3DpprBrick.f
```

where '`job_name`' is the name of the input file.

5.2. Modifying the unloading/reloading relation

The current unloading/reloading relationship in the model is linear to the origin, as illustrated in Fig. 6. Some users may wish to modify this; which may be done by altering only a few lines in the UEL. Unloading effects the traction vector and the tangent matrix, both of which will need updating. For purposes of illustration, we outline the steps necessary to implement a

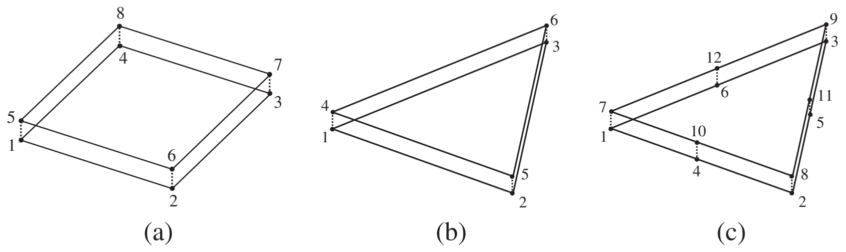
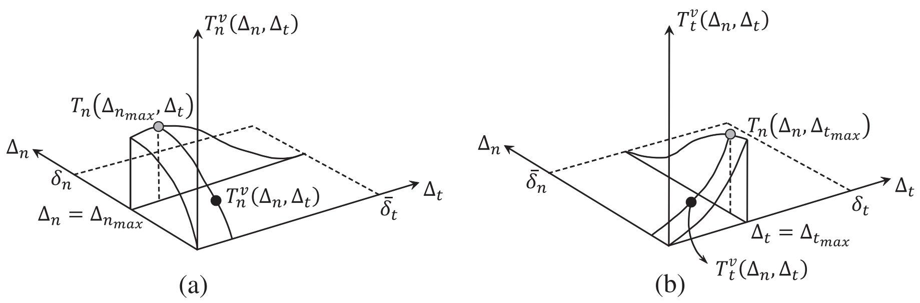


Fig. 7. Node numbering for (a) a CZE with 8 nodes, (b) a CZE with 6 nodes and (c) a CZE with 12 nodes.



$T_n^v(\Delta_n,\Delta_t)$
$T_n(\Delta_{n_{max}},\Delta_t)$
Δ_n
δ_n
$\Delta_n=\Delta_{n_{max}}$
$T_n^v(\Delta_n,\Delta_t)$
$\bar{\delta}_t$
Δ_t
(a)
$T_t^v(\Delta_n,\Delta_t)$
$T_n(\Delta_n,\Delta_{t_{max}})$
Δ_n
$\bar{\delta}_n$
Δ_t
δ_t
$\Delta_t=\Delta_{t_{max}}$
$T_t^v(\Delta_n,\Delta_t)$
(b)

Fig. 8. Depiction of the power-law unloading/reloading relations for the (a) normal traction and (b) tangential traction. When α^v or $\beta^v > 1$, we get a convex unloading/reloading relation, when α^v or $\beta^v < 1$ we get a non-convex unloading/reloading relation.

power-law unloading/reloading relation. Assuming a power-law relation, the corresponding tractions during the unloading phase are reformulated as:

$$T_n^v(\Delta_n,\Delta_t)=T_n(\Delta_{n_{max}},\Delta_t)\left(\frac{\Delta_n}{\Delta_{n_{max}}}\right)^{\alpha_v}\qquad T_t^v(\Delta_n,\Delta_t)=T_t(\Delta_n,\Delta_{t_{max}})\left(\frac{|\Delta_t|}{\Delta_{t_{max}}}\right)^{\beta_v}\frac{\Delta_t}{|\Delta_t|}\tag{19}$$

where α_v and β_v determine the shape of the unloading curve, as shown in Fig. 8. Similarly, the tangent matrix components become:

$$D_{nn}^v=T_n(\Delta_{n_{max}},\Delta_t)\frac{\alpha_v}{\Delta_{n_{max}}}\left(\frac{\Delta_n}{\Delta_{n_{max}}}\right)^{\alpha_v-1},\qquad D_{nt}^v=D_{nt}(\Delta_{n_{max}},\Delta_t)\left(\frac{\Delta_n}{\Delta_{n_{max}}}\right)^{\alpha_v}\tag{20}$$

$$D_{tt}^v=T_t(\Delta_n,\Delta_{t_{max}})\frac{\beta_v}{\Delta_{t_{max}}}\left(\frac{|\Delta_t|}{\Delta_{t_{max}}}\right)^{\beta_v-1},\qquad D_{tn}^v=D_{tn}(\Delta_n,\Delta_{t_{max}})\left(\frac{|\Delta_t|}{\Delta_{t_{max}}}\right)^{\beta_v}\tag{21}$$

To implement these changes into the UEL, we must first introduce additional input parameters, let's call them `alphaV` and `betaV`. This can be done in the input portion of the code (lines 35–42):

```
alphaV = PROPS (9)
betaV  = PROPS (10)
```

We then need to pass these variables into the subfunction `k_cohesive_law`. In order to update the normal unloading traction, we need to modify line 255 to the following:

```
*(popn/pmax)**alphaV
```

Similarly, to modify the tangential unloading traction, we need to change line 272 to:

```
*(popt/tmax)**betaV
```

The process of updating the consistent tangent matrix follows in the same manner as the tractions. Updating is required to the variables D_{nn}, D_{nt}, D_{tt} and D_{tn} on lines 307, 314, 348 and 355, respectively. In the order they appear, these lines need to be changed to:

```
307: *(alphaV/pmax)*(popn/pmax)**(alphaV-1)
314: *(popn/pmax)**alphaV
348: *(betaV/tmax)*(popt/tmax)**(betaV-1)
355: *(popt/tmax)**betaV
```

respectively. It is important to note that, since the unloading/reloading relation is not derived from a potential, the resulting system is not guaranteed to be symmetric.

5.3. Modifications for material gradation

There are multiple ways in which one may chose to grade their material properties [39,40]. This section will outline the modifications necessary to implement a graded normal fracture energy in the *z*-coordinate direction, based on the method proposed by Kim and Paulino [40]. Based on this simple exercise, we hope that the users will be more familiar with the UEL, and more adept at making their own modifications. In order to allocate the range and region of gradation, we need more user inputs; which may be included in the input section of the code (lines 35–42):

```
Gfn_min = PROPS (9)
Gfn_max = PROPS (10)
Zmin = PROPS (11)
Zmax = PROPS (12)
```

where `Gfn_min`, `Gfn_max`, `Zmin` and `Zmax` correspond to the minimum and maximum value of the fracture energy and to the minimum and maximum z-coordinates in the model, respectively. The remainder of the modifications need to occur prior to the first use of the fracture energy (on line 105). To compute the total change in the fracture energy, we include the line:

```
del_Gfn = Gfn_max-Gfn_min
```

Finally, assuming a linear variation of fracture energy, we include the lines:

```
Gfn1 = Gfn_min+((Zmax-co_de_m (3,1))/(Zmax-Zmin))*del_Gfn
Gfn2 = Gfn_min+((Zmax-co_de_m (3,2))/(Zmax-Zmin))*del_Gfn
Gfn3 = Gfn_min+((Zmax-co_de_m (3,3))/(Zmax-Zmin))*del_Gfn
Gfn4 = Gfn_min+((Zmax-co_de_m (3,4))/(Zmax-Zmin))*del_Gfn
Gfn  = sf (1)*Gfn1 + sf (2)*Gfn2 + sf (3)*Gfn3 + sf (4)*Gfn4
```

after we call the subfunction `k_shape_fun` on line 99, and before we determine the inputs to the cohesive model on line 103. The resulting cohesive model will have a varying final crack opening width, and corresponding softening curve, based on the variation of fracture energy. This method, although presented for variation of fracture energy in the z-direction, can be extended to other parameters (including multiple parameters) in any direction, such as that illustrated in Fig. 9 for the case of graded shape parameter, α.

5.4. Modifying the contact relation

The current UELs are implemented with the common penalty stiffness approach to contact. The slope of the stiffness is in accordance with that of the initial hardening slope at the initiation of the intrinsic formulation. There is much research into the appropriate method of accounting for contact in a cohesive element formulation; which could be a desired modification to the formulation [37,38]. In order to change the contact formulation, the user needs to modify the corresponding part of the code in which contact is handled. In the subfunction `k_cohesive_law` the user needs to modify the section of the code in which the normal opening displacement is negative (line 279):

```
if (delu (3).LT. 0.0) then
```

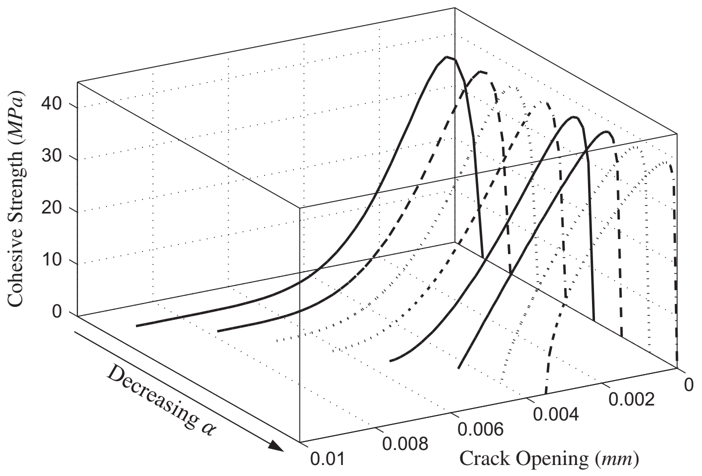


Fig. 9. Effect of varying α on the traction–separation relation in the normal direction. Notice the effect on both the shape of the softening curve and on the final crack opening width.

The users have complete freedom within this section, and may include any additional variables external to this statement as they wish.

In addition to the contact condition considered within the element itself, one may choose to use the surface contact conditions built into ABAQUS [5]. The surface contact conditions use the master–slave surface relationship to prevent element interpenetration. This condition is suitable for situations in which the fracture surfaces are predefined, but would be very difficult to implement if the intrinsic cohesive elements are inserted throughout a region. This method essentially requires one additional step in the analysis. One must define two surfaces, corresponding to the two fracture surfaces that make up the facets in which the cohesive elements are inserted. More information on how to implement this method may be found in the ABAQUS users manual [5].

6. Three-dimensional example problems

Some of the capabilities of the elements will be demonstrated here. The following problems are presented and discussed: (1) the mixed-mode bending (MMB) test problem, (2) the small deformation of a coated particle debonding from a matrix, and (3) the finite deformation debonding of a rigid particle reinforced elastomer.

6.1. Mixed-mode bending specimen

The mixed-mode bending problem is chosen as a simple example to compare the three cohesive elements to one another. The geometry of the problem is illustrated in Fig. 10. This is a popular test specimen to estimate fracture energies under mixed-mode loading conditions [41]. Its Cartesian geometry can be represented in a two-dimensional plane, and thus has received a lot of interest in fracture related publications [42–44,46]. Adding to its popularity is the presence of an analytical solution. The analytical solution consists of three separate components [45,6]. Initially, the response is linear, and can be compared with results from linear beam theory. However, once fracture initiates, the response follows that derived from linear elastic fracture mechanics. The elastic beam has a modulus of 122 GPa, and a Poisson's ratio of 0.3.

We consider two different scenarios for the cohesive properties. In both scenarios, the shape parameters (α, β) and initial slope indicators (λ_n, λ_t) are set equal to 3.0 and 0.02, respectively. The first scenario considers a fracture energy of 500 N/m in both the normal and tangential directions, and equivalent cohesive strengths in each direction. The influence of the magnitude of the cohesive strength is investigated, and the results are illustrated in Fig. 11(a). Similar behavior is observed for each element type (see Fig. 1), thus only the results using linear tetrahedral bulk elements are shown. As the cohesive strengths increase, the numerical solution approaches the analytical solution. The second scenario considers different fracture energies in the normal ($\phi_n = 500$ N/m) and tangential ($\phi_t = 500$ N/m) directions. The cohesive strength in the normal direction is set as 20 MPa and the strength in the tangential direction is varied. As the tangential cohesive strength increases, the results approach the analytical solution, as illustrated in Fig. 11(b). As expected, the present results for the three-dimensional problem are essentially the same as those obtained by Park and Paulino for an equivalent two-dimensional problem [11].

6.2. Small deformation coated particle debonding

The following example is that of a single coated particle imbedded in an elastic matrix. In composite materials, such as this, the micro-geometry and interface conditions have a significant influence on the macroscopic behavior [47–50]. The computational study is conducted on a single coated particle, embedded in a matrix. The model is simplified by only considering a single octant of the particle, with symmetric boundary conditions, as illustrated in Fig. 12. The particle has a radius of 1 mm, the coating has a thickness of 0.2 mm, and the particle volume fraction is 40%. Linear, eight-node brick (B8) elements are used to discretize the domain. Mesh refinement studies on the model indicate that meshes with approximately 150,000 bulk elements produce accurate results, and is the level of refinement used in this example, as illustrated in Fig. 12(b). With this level of refinement, 3230 cohesive elements are inserted between the coating and the bulk matrix to account for the debonding behavior. The composite structure is loaded hydrostatically through displacement boundary

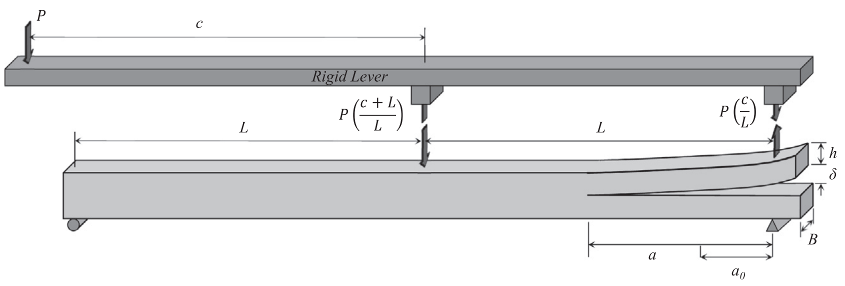


Fig. 10. Geometry and boundary conditions of the MMB beam. $L = 51$ mm, $c = 60$ mm, $B = 25.4$ mm, $h = 1.56$ mm, $a_0 = 33.7$ mm,$P = 850$ N.

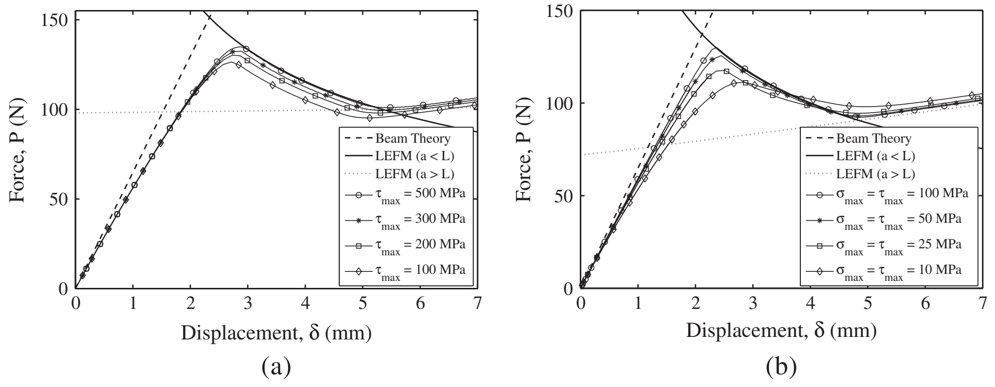


Fig. 11. Representative results for MMB beam using linear tetrahedral elements. The results with linear brick and quadratic tetrahedral elements are similar and thus are not repeated here.

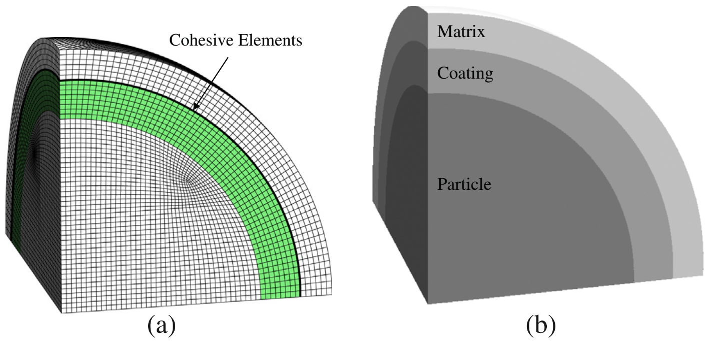


Fig. 12. Coated particle model. Debonding occurs between coating and matrix. (a) Geometry and (b) mesh.

conditions applied to the exterior surface of the model. The particle and the matrix have an elastic modulus of 100 MPa, with a Poisson's ratio of 0.25. The coating is assumed to be stiffer, and has an elastic modulus of 200 MPa, with a Poisson's ratio of 0.25. The cohesive fracture energy, ϕ_n, is varied, while the cohesive strength (σ_{max}), shape parameter (α) and initial slope indicator (λ_n) are set as 10 MPa, 3.0 and 0.005, respectively.

The results are illustrated in Fig. 13. The initial hardening slope is that of the combined, perfectly bonded structure. At large macroscopic strains, complete separation occurs and the load is carried entirely by the matrix shell. The transition between the initial and final hardening slopes is a function of the cohesive fracture energy. As the cohesive fracture energy increases, the transition becomes more gradual; which is consistent with the trends that Ngo et al. [51] observed for particles without coatings.

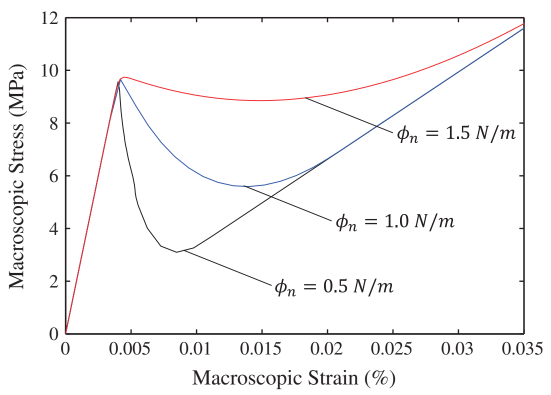


Fig. 13. Macroscopic stress vs. strain response of particle debonding under hydrostatic loading.

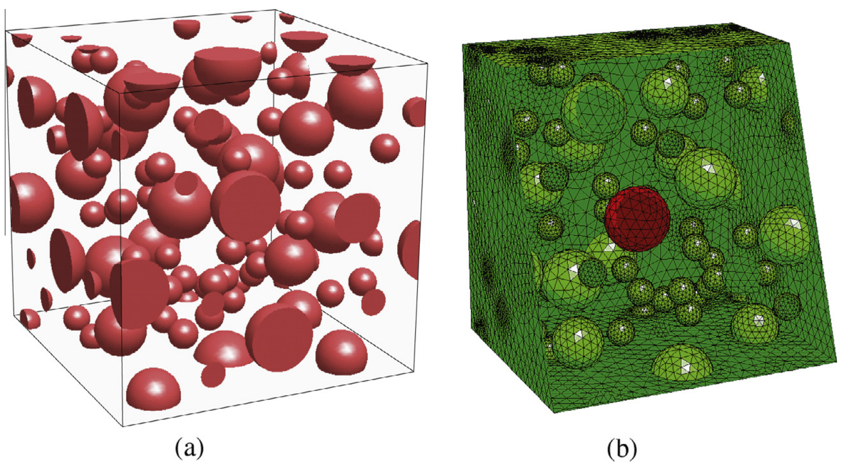


Fig. 14. Representative volume element with rigid particle inclusions. (a) Geometry and (b) mesh.

6.3. Finite deformation particle debonding

Recent studies suggest that the inclusion of particles in an elastomer acts to greatly increase the stiffness of these composite materials. The behavior at the interface of such composite materials is an important area of current research [47,52,53]. There are two commonly accepted phenomena which occur when particles are included in an elastomer. First, the particles influence the microstructure of the bulk material, and second, the particles may debond from the bulk material under finite deformations. This example proposes using cohesive elements to capture the particle debonding behavior under finite deformations. Under consideration is a cubic representative volume element (RVE) with 80 randomly placed polydisperse particles imbedded in the RVE; making up a total volume fraction of 10%, as illustrated in Fig. 14(a). The polydisperse particles have three different radii, as suggested by Lopez-Pamies et al. [54]. There are 10 particles with a 10 μm diameter, 10 particles with a 7.95 μm diameter, and 60 particles with a 4.4 μm diameter. The RVE is periodic, with periodic boundary conditions, meaning that the complete microstructure of the material can be obtained by translating the RVE in the three Cartesian directions [55,56]. The finite element mesh consists of approximately 100,000 quadratic tetrahedral elements, and is generated using the automatic mesh generator NETGEN [57]. A section of the mesh is shown in Fig. 14(b).

The matrix is modeled as an incompressible Neo-Hookean material with a shear modulus of $\mu = 1.0$ MPa. The stored-energy function for Neo-Hookean rubber is expressed in Eq. (1). The particles are also modeled using an incompressible Neo-Hookean material, with a shear modulus of $\mu = 10,000$ MPa. The large difference between the moduli for the matrix

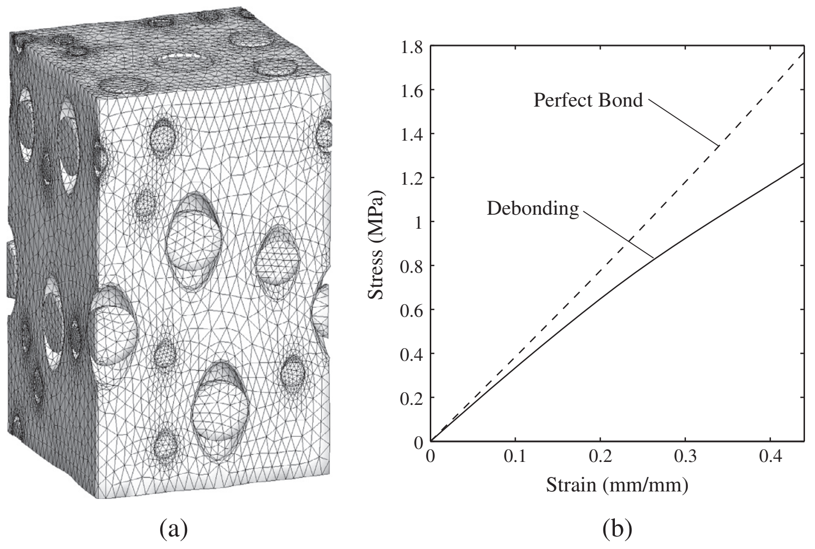


Fig. 15. Deformation behavior of the RVE under uniaxial stretch. (a) Deformed shape, illustrating the concurrent separation of the particles from the matrix and (b) stress–strain response of the RVE in comparison to the same RVE without the inclusion of particle debonding.

and the particles results in the particles acting rigidly. The fracture energies (ϕ_n, ϕ_t), cohesive strengths $(\sigma_{max}, \tau_{max})$, shape parameters (α, β), and initial slope indicators (λ_n, λ_t) are set as 1.0 N/m, 0.5 MPa, 3 and 0.2, respectively. The deformed shape, illustrated in Fig. 15(a) illustrates the concurrent debonding of all the particles from the matrix. The constitutive response of the composite is illustrated in Fig. 15(b). The RVE undergoes a much higher strain, under the same imposed stress, when debonding is considered; demonstrating the importance and significance of including particle debonding in such composite materials. Notice that Fig. 15(b) provides a global (aggregated) response of the RVE under uniaxial stretch. Locally, one can observe finite cohesive displacements on the interfaces and finite strains in the matrix (see Fig. 15(a)).

7. Concluding remarks

This paper outlines the implementation of a three-dimensional cohesive zone element into a user defined subroutine for use in ABAQUS. In total, three different cohesive elements are implemented; which are compatible with linear brick, linear tetrahedral and quadratic tetrahedral bulk elements (see Fig. 1). The constitutive model for the cohesive elements is independent of that for the bulk elements. The selected constitutive model for the cohesive elements is the intrinsic PPR model. A linear unloading/reloading relation and a penalty stiffness approach to contact are chosen. The modulus of the penalty stiffness is set equal to the initial hardening slope at the initiation of the intrinsic formulation.

In addition to the base formulation, a series of suggested modifications is presented. These modifications highlight the sections of the code which require changing if the user chooses to implement alternate unloading/reloading and contact conditions. As well, the modifications necessary to implement graded cohesive elements are presented. In order to present the varied possible applications, three numerical examples are presented: the mixed-mode bending beam, the small deformation debonding of a coated particle and the finite deformation debonding of multiple particles embedded in an elastomer. This paper is written from an educational perspective, and is aimed at promoting the use of cohesive elements to model the fracture and failure of materials using either commercial finite element software, such as ABAQUS; or research oriented software.

Acknowledgments

We acknowledge support from the Natural Sciences and Engineering Research Council of Canada and from the U.S. National Science Foundation (NSF) through grant CMMI #1321661. We also acknowledge support from the Donald B. and Elizabeth M. Willett endowment at the University of Illinois at Urbana-Champaign (UIUC). The information presented in this publication is the sole opinion of the authors and does not necessarily reflect the views of the sponsors or sponsoring agencies.

Appendix A. Tangent stiffness matrix

The local constitutive matrix, specific to the PPR model, is expressed in Eq. (8), where the components D_{nn}, D_{nt}, D_{tn}, and D_{nn} are calculated as:

$$D_{nn} = \frac{\partial^2 \Psi}{\partial \Delta_n^2} = \frac{\partial T_n}{\partial \Delta_n} = \frac{\Gamma_n}{\delta_n^2}\left[(m^2-m)\left(1-\frac{\Delta_n}{\delta_n}\right)^{\alpha}\left(\frac{m}{\alpha}+\frac{\Delta_n}{\delta_n}\right)^{m-2} + (\alpha^2-\alpha)\left(1-\frac{\Delta_n}{\delta_n}\right)^{\alpha-2}\left(\frac{m}{\alpha}+\frac{\Delta_n}{\delta_n}\right)^{m} - 2\alpha m\left(1-\frac{\Delta_n}{\delta_n}\right)^{\alpha-1}\left(\frac{m}{\alpha}+\frac{\Delta_n}{\delta_n}\right)^{m-1}\right]\left[\Gamma_t\left(1-\frac{|\Delta_t|}{\delta_t}\right)^{\beta}\left(\frac{n}{\beta}+\frac{|\Delta_t|}{\delta_t}\right)^{n} + \langle \phi_t-\phi_n\rangle\right] \tag{A.1}$$

$$D_{tt} = \frac{\partial^2 \Psi}{\partial \Delta_t^2} = \frac{\partial T_t}{\partial \Delta_t} = \frac{\Gamma_t}{\delta_t^2}\left[(n^2-n)\left(1-\frac{|\Delta_t|}{\delta_t}\right)^{\beta}\left(\frac{n}{\beta}+\frac{|\Delta_t|}{\delta_t}\right)^{n-2} + (\beta^2-\beta)\left(1-\frac{|\Delta_t|}{\delta_t}\right)^{\beta-2}\left(\frac{n}{\beta}+\frac{|\Delta_t|}{\delta_t}\right)^{n} - 2\beta n\left(1-\frac{|\Delta_t|}{\delta_t}\right)^{\beta-1}\left(\frac{n}{\beta}+\frac{|\Delta_t|}{\delta_t}\right)^{n-1}\right]\left[\Gamma_n\left(1-\frac{\Delta_n}{\delta_n}\right)^{\alpha}\left(\frac{m}{\alpha}+\frac{\Delta_n}{\delta_n}\right)^{m} + \langle \phi_n-\phi_t\rangle\right] \tag{A.2}$$

$$D_{nt} = \frac{\partial^2 \Psi}{\partial \Delta_n \partial \Delta_t} = \frac{\partial T_n}{\partial \Delta_t} = \frac{\Gamma_n \Gamma_t}{\delta_n \delta_t}\left[m\left(1-\frac{\Delta_n}{\delta_n}\right)^{\alpha}\left(\frac{m}{\alpha}+\frac{\Delta_n}{\delta_n}\right)^{m-1} - \alpha\left(1-\frac{\Delta_n}{\delta_n}\right)^{\alpha-1}\left(\frac{m}{\alpha}+\frac{\Delta_n}{\delta_n}\right)^{m}\right]\left[n\left(1-\frac{|\Delta_t|}{\delta_t}\right)^{\beta}\left(\frac{n}{\beta}+\frac{|\Delta_t|}{\delta_t}\right)^{n-1} - \beta\left(1-\frac{|\Delta_t|}{\delta_t}\right)^{\beta-1}\left(\frac{n}{\beta}+\frac{|\Delta_t|}{\delta_t}\right)^{n}\right]\frac{\Delta_t}{|\Delta_t|} \tag{A.3}$$

$$D_{tn} = D_{nt} \tag{A.4}$$

Appendix B. ABAQUS: User Defined Subroutine[2]

```
c 3D PPR UEL for ABAQUS Code (Compatible with 4 node Tet Elements)
c Reference: D Spring, GH Paulino, "A Growing Library of Cohesive
c Elements for Use in ABAQUS". Engineering Fracture Mechanics
c ================================================================
      SUBROUTINE UEL (RHS, AMATRX, SVARS, ENERGY, NDOFEL, NRHS,
     1 NSVARS, PROPS, NPROPS, COORDS, MCRD, NNODE, U, DU, V, A, JTYPE,
     2 TIME, DTIME, KSTEP, KINC, JELEM, PARAMS, NDLOAD, JDLTYP,
     3 ADLMAG, PREDEF, NPREDF, LFLAGS, MLVARX, DDLMAG, MDLOAD, PNEWDT,
     4 JPROPS, NJPROP, PERIOD)
c
      INCLUDE 'ABA_PARAM.INC'
c
      DIMENSION RHS(MLVARX,*),AMATRX(NDOFEL,NDOFEL),PROPS(*),
     1 SVARS(*),ENERGY(8),COORDS(MCRD,NNODE),U(NDOFEL),
     2 DU(MLVARX,*),V(NDOFEL),A(NDOFEL),TIME(2),PARAMS(*),
     3 JDLTYP(MDLOAD,*),ADLMAG(MDLOAD,*),DDLMAG(MDLOAD,*),
     4 PREDEF(2,NPREDF,NNODE),LFLAGS(*),JPROPS(*)
c
      DIMENSION ds1(3),ds2(3),dn(3),Trac(MCRD,NRHS),
     1 Trac_Jacob(MCRD,MCRD),R(MCRD,MCRD),coord_l(MCRD,NNODE),
     2 GP_coord(2),sf(3),B(MCRD,NDOFEL),co_de_m(3,3),
     3 B_t(NDOFEL,MCRD), Transformation_M(NDOFEL,NDOFEL),
     4 Transformation_M_T(NDOFEL,NDOFEL),temp1(MCRD,NDOFEL)
c
      DIMENSION stiff_l(NDOFEL,NDOFEL),temp2(NDOFEL,NDOFEL),
     1 stiff_g(NDOFEL,NDOFEL),residual_l(NDOFEL,NRHS),
     2 residual_g(NDOFEL,NRHS),aJacob_M(2,3),delu_loc_gp(mcrd),
     3 co_de(mcrd,nnode)
c
      DOUBLE PRECISION G_fn, G_ft, f_tn, f_tt, alpha, beta, rn, rt,
     1 p_m, p_n, deln, delt, tmax, pmax, opn, opt
c
c Define Inputs====================================================
c
      G_fn=PROPS(1)
      G_ft=PROPS(2)
      f_tn=PROPS(3)
      f_tt=PROPS(4)
      alpha=PROPS(5)
      beta=PROPS(6)
      rn=PROPS(7)
      rt=PROPS(8)
      GP_n=3d0
      GP_W = 1.0d0/3.0d0
c
```

[2] The source code provided in this appendix, compatible with the linear tetrahedral element, can be downloaded from the url http://ghpaulino.com. The source code for the other elements of Fig. 1 (linear brick and quadratic tetrahedral) can also be downloaded from this url.

```
c Initialize Matrices and Vectors=====================================
c
      call k_vector_zero(ds1,3)
      call k_vector_zero(ds2,3)
      call k_vector_zero(dn,3)
      call k_matrix_zero(Trac,mcrd,nrhs)
      call k_matrix_zero(Trac_Jacob,mcrd,mcrd)
      call k_matrix_zero(R,mcrd,mcrd)
      call k_matrix_zero(coord_l,mcrd,nnode)
      call k_vector_zero(GP_coord,2)
      call k_vector_zero(sf,3)
      call k_matrix_zero(Transformation_M,ndofel,ndofel)
      call k_matrix_zero(Transformation_M_T,ndofel,ndofel)
      call k_matrix_zero(B,mcrd,ndofel)
      call k_matrix_zero(B_t,ndofel,mcrd)
      call k_matrix_zero(temp1,mcrd,ndofel)
      call k_matrix_zero(stiff_l,ndofel,ndofel)
      call k_matrix_zero(temp2,ndofel,ndofel)
      call k_matrix_zero(stiff_g,ndofel,ndofel)
      call k_matrix_zero(residual_l,ndofel,nrhs)
      call k_matrix_zero(residual_g,ndofel,nrhs)
      call k_matrix_zero(aJacob_M,2,3)
      call k_matrix_zero(rhs,ndofel,nrhs)
      call k_matrix_zero(amatrx,ndofel,ndofel)
      call k_matrix_zero(co_de,mcrd,nnode)
      a_Jacob=0.d0
c
c Do local computations==============================================
c
      do i = 1, mcrd
         do j = 1, nnode
            co_de(i,j)=coords(i,j)+U(3.0*(j-1.0)+i)
         end do
      end do
c
      call k_local_coordinates(co_de,R,coord_l,Transformation_M,
     & Transformation_M_T,a_Jacob,aJacob_M,coords,u,ndofel,nnode,
     & mcrd)
c
c Compute shear and normal local opening displacments=================
c
      do i = 1, 3
         ds1(i)=coord_l(1,i+3)-coord_l(1,i)
         ds2(i)=coord_l(2,i+3)-coord_l(2,i)
         dn(i) =coord_l(3,i+3)-coord_l(3,i)
      end do
c
c Do Calculations at Gauss Points=====================================
c
      do i = 1, GP_n
c
c Determine the values of the shape function at each Gauss Point
```

```
c
         call k_shape_fun(i,sf)
c
c Determine Inputs to Cohesive Model===================================
c
         p_m=(alpha*(alpha-1.0)*rn**2.0)/(1.0-alpha*rn**2.0)
         p_n=(beta*(beta-1.0)*rt**2.0)/(1.0-beta*rt**2.0)
         deln=(G_fn/f_tn)*alpha*rn*(1.0-rn)**(alpha-1.0)
     & *((alpha/p_m)+1.0)*((alpha/p_m)*rn+1.0)**(p_m-1.0)
         delt=(G_ft/f_tt)*beta*rt*(1.0-rt)**(beta-1.0)
     & *((beta/p_n)+1.0)*((beta/p_n)*rt+1.0)**(p_n-1.0)
c
         call k_vector_zero(delu_loc_gp,mcrd)
c
c Determine shear and normal opening displamenets at Gauss points
c
         do j = 1, 3
            delu_loc_gp(1)=delu_loc_gp(1)+ds1(j)*sf(j)
            delu_loc_gp(2)=delu_loc_gp(2)+ds2(j)*sf(j)
            delu_loc_gp(3)=delu_loc_gp(3)+ dn(j)*sf(j)
         end do
c
         opn=delu_loc_gp(3)
         opt=sqrt(delu_loc_gp(1)**2.0+delu_loc_gp(2)**2.0)
c
         if ((Svars(GP_n*(i-1.0)+1.0) .LT. opt) .AND.
     & (opt .GT. rt*delt)) then
            Svars(GP_n*(i-1.0)+1.0)=opt
         end if
         if ((Svars(GP_n*(i-1.0)+2.0) .LT. opn) .AND.
     & (opn .GT. rn*deln)) then
            Svars(GP_n*(i-1.0)+2.0)=opn
         end if
         tmax=Svars(GP_n*(i-1.0)+1.0)
         pmax=Svars(GP_n*(i-1.0)+2.0)
c
c Determine Traction vector and tangent modulus matrix
c
         call k_cohesive_law(Trac,Trac_Jacob,G_fn,G_ft,deln,delt,
     & alpha,beta,p_m,p_n,pmax,tmax,delu_loc_gp,mcrd,nrhs)
c
c Determine B matrix and its transpose
c
         call k_Bmatrix(sf,B,mcrd,ndofel)
c
         call k_matrix_transpose(B,B_t,mcrd,ndofel)
c
c Compute the stiffness matrix
c Local Stiffness = B_t * Trac_Jacob * B
c
         call k_matrix_multiply(Trac_Jacob,B,temp1,mcrd,mcrd,
     & ndofel)
```

```
           call k_matrix_multiply(B_t,temp1,stiff_l,ndofel,
     & mcrd,ndofel)
c
c Compute Global stiffness matrix
c Global_K = Transpose(T) * K * T
c
           call k_matrix_multiply(Transformation_M_T,stiff_l,
     & temp2,ndofel,ndofel,ndofel)
           call k_matrix_multiply(temp2,Transformation_M,stiff_g,
     & ndofel,ndofel,ndofel)
c
c Multiply Jacobian with the Global stiffness and add contribution
c from each Gauss Point
c
           a_Mult = a_Jacob*GP_w
           call k_matrix_plus_scalar(amatrx,stiff_g,a_Mult,
     & ndofel,ndofel)
c
c Compute the global residual vector
c Local_residual = B_t * Trac
c Global_residual = Transpose(T) * Local_residual
c
           call k_matrix_multiply(B_t,Trac,residual_l,ndofel,
     & mcrd,nrhs)
           call k_matrix_multiply(Transformation_M_T,residual_l,
     & residual_g,ndofel,ndofel,nrhs)
c
c Multiply the Global residual by the Jacobian and add the
c contribution from each point
c
           call k_matrix_plus_scalar(rhs,residual_g,a_Mult,
     & ndofel,nrhs)
        end do
c
        return
        end
c===================================================================
c=============================SUBROUTINES===========================
c===================================================================
c
c Determine the global displacement-separation (B) matrix
c
        subroutine k_Bmatrix(sf,B,mcrd,ndofel)
        INCLUDE 'ABA_PARAM.INC'
        dimension sf(3),B(mcrd,ndofel)
        B(1,1) =  sf(1)
        B(1,4) =  sf(2)
        B(1,7) =  sf(3)
        B(1,10)= -sf(1)
        B(1,13)= -sf(2)
        B(1,16)= -sf(3)
        B(2,2) =  sf(1)
```

```
      B(2,5) =  sf(2)
      B(2,8) =  sf(3)
      B(2,11)= -sf(1)
      B(2,14)= -sf(2)
      B(2,17)= -sf(3)
      B(3,3) =  sf(1)
      B(3,6) =  sf(2)
      B(3,9) =  sf(3)
      B(3,12)= -sf(1)
      B(3,15)= -sf(2)
      B(3,18)= -sf(3)
c
      return
      end
c=====================================================================
      subroutine k_cohesive_law(T,T_d,G_fn,G_ft,deln,delt,
     & alpha,beta,p_m,p_n,pmax,tmax,delu,mcrd,nrhs)
      INCLUDE 'ABA_PARAM.INC'
      dimension T(mcrd,nrhs),T_d(mcrd,mcrd),delu(mcrd)
       DOUBLE PRECISION G_fn, G_ft, f_tn, f_tt, alpha, beta,
     & p_m, p_n, deln, delt, tmax, pmax, popn, popt, gam_n,
     & gam_t, Tn, Tt, Dnn, Dnt, Dtt, T_d, T, delu
c
      popn=delu(3)
      popt=sqrt(delu(1)**2.0+delu(2)**2.0)
c
      call k_Mac(pM1,G_fn,G_ft)
      call k_Mac(pM2,G_ft,G_fn)
c
      if (G_fn .NE. G_ft) then
         gam_n=(-G_fn)**(pM1/(G_fn-G_ft))*(alpha/p_m)**p_m
         gam_t=(-G_ft)**(pM2/(G_ft-G_fn))*(beta/p_n)**p_n
      elseif (G_fn .EQ. G_ft) then
         gam_n=-G_fn*(alpha/p_m)**p_m
         gam_t=(beta/p_n)**p_n
      end if
c
c Pre-calculation of the normal cohesive traction Tn
c
      if (popn .LT. 0.0) then
         popn=0.0
      elseif ((popn .GE. deln) .OR. (popt .GE. delt)) then
         Tn = 0.0
      elseif (popn .GE. pmax) then
         Tn=(gam_n/deln)*(p_m*(1.0-(popn/deln))**alpha*((p_m/alpha)
     & +(popn/deln))**(p_m-1.0)-alpha*((1.0-(popn/deln))**(alpha-1.0))
     & *((p_m/alpha)+(popn/deln))**p_m)*(gam_t*(1.0-(popt/delt))
     & **beta*((p_n/beta)+(popt/delt))**p_n+pM2)
      else
         Tn=(gam_n/deln)*(p_m*(1.0-(pmax/deln))**alpha*((p_m/alpha)
     & +(pmax/deln))**(p_m-1.0)-alpha*((1.0-(pmax/deln))**(alpha-1.0))
     & *((p_m/alpha)+(pmax/deln))**p_m)*(gam_t*(1.0-(popt/delt))
     & **beta*((p_n/beta)+(popt/delt))**p_n+pM2)
     & *popn/pmax
      end if
c
c Pre-calculation of the tangential cohesive traction, Tt
c
      if ((popn .GE. deln) .OR. (popt .GE. delt)) then
         Tt = 0.0
      elseif (popt .GE. tmax) then
         Tt=(gam_t/delt)*(p_n*(1.0-(popt/delt))**beta*((p_n/beta)
     & +(popt/delt))**(p_n-1.0)-beta*((1.0-(popt/delt))**(beta-1.0))
     & *((p_n/beta)+(popt/delt))**p_n)*(gam_n*(1.0-(popn/deln))
     & **alpha*((p_m/alpha)+(popn/deln))**p_m+pM1)
      else
         Tt=(gam_t/delt)*(p_n*(1.0-(tmax/delt))**beta*((p_n/beta)
     & +(tmax/delt))**(p_n-1.0)-beta*((1.0-(tmax/delt))**(beta-1.0))
     & *((p_n/beta)+(tmax/delt))**p_n)*(gam_n*(1.0-(popn/deln))
     & **alpha*((p_m/alpha)+(popn/deln))**p_m+pM1)
     & *popt/tmax
      end if
c
c Algortihm 1
c
c Normal Cohesive Interaction
c (1) Contact
      if (delu(3) .LT. 0.0) then
         Dnn = -(gam_n/(deln**2))*(p_m/alpha)**(p_m-1.0)*(alpha+p_m)*
     & (gam_t*(p_n/beta)**p_n + pM2)
         Dnt = 0.0
         Tn = Dnn * delu(3)
      else if ((popn .LT. deln) .AND. (popt .LT. delt)
     & .AND. (Tn .GE. -1.0E-5)) then
         Tn = Tn
c (2) Softening Condition
         if (popn .GE. pmax) then
         Dnn=(gam_n/(deln**2.0))*((p_m**2.0-p_m)*((1.0-(popn/deln))
     & **alpha)*((p_m/alpha)+(popn/deln))**(p_m-2.0)+(alpha**2.0
     & -alpha)*((1.0-(popn/deln))**(alpha-2.0))*((p_m/alpha)
     & +(popn/deln))**p_m -2.0*alpha*p_m*((1.0-(popn/deln))
     & **(alpha-1.0))*((p_m/alpha)+(popn/deln))**(p_m-1.0))*(gam_t*
     & (1.0-(popt/delt))**beta*((p_n/beta)+(popt/delt))**p_n+pM2)
         Dnt=(gam_n*gam_t/(deln*delt))*(p_m*((1.0-(popn/deln))**alpha)
     & *((p_m/alpha)+(popn/deln))**(p_m-1.0)-alpha*((1.0-(popn/deln))
     & **(alpha-1.0))*((p_m/alpha)+(popn/deln))**p_m)*(p_n
     & *((1.0-(popt/delt))**beta)*(((p_n/beta)+(popt/delt))
     & **(p_n-1.0))-beta*((1.0-(popt/delt))**(beta-1.0))*((p_n/beta)
     & +(popt/delt))**p_n)
c (3) Unloading/reloading condition
         else
         Dnn=(gam_n/deln)*(p_m*(1.0-(pmax/deln))**alpha*((p_m/alpha)
     & +(pmax/deln))**(p_m-1.0)-alpha*((1.0-(pmax/deln))**(alpha-1.0))
     & *((p_m/alpha)+(pmax/deln))**p_m)*(gam_t*(1.0-(popt/delt))
     & **beta*((p_n/beta)+(popt/delt))**p_n+pM2)
     & /pmax
         Dnt=(gam_n*gam_t/(deln*delt))*(p_m*((1.0-(pmax/deln))**alpha)
     & *((p_m/alpha)+(pmax/deln))**(p_m-1.0)-alpha*((1.0-(pmax/deln))
     & **(alpha-1.0))*((p_m/alpha)+(pmax/deln))**p_m)*(p_n
     & *((1.0-(popt/delt))**beta)*(((p_n/beta)+(popt/delt))
     & **(p_n-1.0))-beta*((1.0-(popt/delt))**(beta-1.0))*((p_n/beta)
     & +(popt/delt))**p_n)
     & *popn/pmax
         end if
c (4) Complete Failure
      else
         Tn = 0.0
         Dnn = 0.0
         Dnt = 0.0
      end if
c
c Tangential Cohesive Interaction
c
      if ((popt .LT. delt) .AND. (popn .LT. deln)
     & .AND. (Tt .GE. -1.0E-5)) then
        Tt = Tt
c (1) Softening Condition
        if (popt .GE. tmax) then
           Dtt=(gam_t/(delt**2.0))*((p_n**2.0-p_n)*((1.0-(popt/delt))
     & **beta)*((p_n/beta)+(popt/delt))**(p_n-2.0)+(beta**2.0-beta)
     & *((1.0-(popt/delt))**(beta-2.0))*((p_n/beta)+(popt/delt))**p_n
     & -2.0*beta*p_n*((1.0-(popt/delt))**(beta-1.0))*((p_n/beta)
     & +(popt/delt))**(p_n-1.0))*(gam_n*(1.0-(popn/deln))
     & **alpha*((p_m/alpha)+(popn/deln))**p_m+pM1)
           Dtn=(gam_n*gam_t/(deln*delt))*(p_m*((1.0-(popn/deln))
     & **alpha)*((p_m/alpha)+(popn/deln))**(p_m-1.0)-alpha*((1.0-
     & (popn/deln))**(alpha-1.0))*((p_m/alpha)+(popn/deln))**p_m)*(p_n
     & *((1.0-(popt/delt))**beta)*(((p_n/beta)+(popt/delt))
     & **(p_n-1.0))-beta*((1.0-(popt/delt))**(beta-1.0))*((p_n/beta)
     & +(popt/delt))**p_n)
c (2) Unloading/reloading condition
        else
           Dtt=(gam_t/delt)*(p_n*(1.0-(tmax/delt))**beta*((p_n/beta)
     & +(tmax/delt))**(p_n-1.0)-beta*((1.0-(tmax/delt))**(beta-1.0))
     & *((p_n/beta)+(tmax/delt))**p_n)*(gam_n*(1.0-(popn/deln))
     & **alpha*((p_m/alpha)+(popn/deln))**p_m+pM1)
     & /tmax
           Dtn=(gam_n*gam_t/(deln*delt))*(p_m*((1.0-(popn/deln))
     & **alpha)*((p_m/alpha)+(popn/deln))**(p_m-1.0)-alpha*((1.0-
     & (popn/deln))**(alpha-1.0))*((p_m/alpha)+(popn/deln))**p_m)*(p_n
     & *((1.0-(tmax/delt))**beta)*(((p_n/beta)+(tmax/delt))
     & **(p_n-1.0))-beta*((1.0-(tmax/delt))**(beta-1.0))*((p_n/beta)+
     & (tmax/delt))**p_n)
     & *popt/tmax
        end if
c (3) Complete failure condition
```

```
      else
        Tt = 0.0
        Dtt = 0.0
        Dtn = 0.0
      end if
      if (Dtn .NE. Dnt) then
        Dtn = 0.5*(Dtn + Dnt)
        Dnt = Dtn
      end if
c
      if (popt .EQ. 0.0) then
         T(1,1) = 0.0
         T(2,1) = 0.0
         T(3,1) = Tn
c
         T_d(1,1)=Dtt
         T_d(1,2)=Dtn
         T_d(1,3)=0.0
         T_d(2,1)=Dnt
         T_d(2,2)=Dtt
         T_d(2,3)=0.0
         T_d(3,1)=0.0
         T_d(3,2)=0.0
         T_d(3,3)=Dnn
      else
         T(1,1)=Tt*delu(1)/popt
         T(2,1)=Tt*delu(2)/popt
         T(3,1)=Tn
c
         T_d(1,1)=Dtt*(delu(1)/popt)**2.0+Tt*((delu(2)**2.0)
     &    /(popt**3.0))
         T_d(1,2)=Dtt*delu(1)*delu(2)/(popt**2.0)
     &    -Tt*delu(1)*delu(2)/(popt**3.0)
         T_d(1,3)=Dnt*delu(1)/popt
c
         T_d(2,1)=Dtt*delu(1)*delu(2)/(popt**2.0)
     &    -Tt*delu(1)*delu(2)/(popt**3.0)
         T_d(2,2)=Dtt*(delu(2)/popt)**2.0+Tt*((delu(1)**2.0)
     &    /(popt**3.0))
         T_d(2,3)=Dnt*delu(2)/popt
c
         T_d(3,1)=Dtn*delu(1)/popt
         T_d(3,2)=Dtn*delu(2)/popt
         T_d(3,3)=Dnn
      end if
c
      return
      end
c=============================================================
      subroutine k_local_coordinates(co_de,R,coord_l,Transformation_M,
     & Transformation_M_T,a_Jacob,aJacob_M,coords,u,ndofel,nnode,
     & mcrd)
```

```
      INCLUDE 'ABA_PARAM.INC'
      dimension R(mcrd,mcrd),coord_l(mcrd,nnode),aJacob_M(2,3),
     & Transformation_M(ndofel,ndofel),coords(mcrd,nnode),
     & Transformation_M_T(ndofel,ndofel),u(ndofel),
     & co_de(mcrd,nnode), co_de_m(3,3),SFD(2,4)
c
      call k_matrix_zero(co_de_m,3,4)
c
      do i = 1, 3
         co_de_m(i,1)=(co_de(i,1)+co_de(i,4))*0.5
         co_de_m(i,2)=(co_de(i,2)+co_de(i,5))*0.5
         co_de_m(i,3)=(co_de(i,3)+co_de(i,6))*0.5
      end do
c
      SFD(1,1) =-1
      SFD(1,2) = 1
      SFD(1,3) = 0
      SFD(2,1) =-1
      SFD(2,2) = 0
      SFD(2,3) = 1
c
      do i = 1,2
         do j = 1,3
            do k =1, 3
               aJacob_M(i,j) = aJacob_M(i,j) + SFD(i,k)*co_de_m(j,k)
            end do
         end do
      end do

      dum1 = aJacob_M(1,2)*aJacob_M(2,3) - aJacob_M(1,3)*aJacob_M(2,2)
      dum2 = aJacob_M(1,3)*aJacob_M(2,1) - aJacob_M(1,1)*aJacob_M(2,3)
      dum3 = aJacob_M(1,1)*aJacob_M(2,2) - aJacob_M(1,2)*aJacob_M(2,1)
c
      a_Jacob = sqrt(dum1**2 + dum2**2 + dum3**2)/2.0d0
      Rn1 = sqrt(dum1**2 + dum2**2 + dum3**2)
c
      R(3,1) = dum1/Rn1
      R(3,2) = dum2/Rn1
      R(3,3) = dum3/Rn1
c
      aLen=sqrt(aJacob_M(1,1)**2.0 + aJacob_M(1,2)**2.0
     & + aJacob_M(1,3)**2.0)
      R(1,1)=aJacob_M(1,1)/aLen
      R(1,2)=aJacob_M(1,2)/aLen
      R(1,3)=aJacob_M(1,3)/aLen
c
      R(2,1)=R(3,2)*R(1,3)-R(3,3)*R(1,2)
      R(2,2)=R(3,3)*R(1,1)-R(3,1)*R(1,3)
      R(2,3)=R(3,1)*R(1,2)-R(3,2)*R(1,1)
c
c=====================================================================
      num=nnode
```

```
c
      do i = 1, num
         dum=3.0*(i-1.0)
         Transformation_M(dum+1,dum+1)=R(1,1)
         Transformation_M(dum+1,dum+2)=R(1,2)
         Transformation_M(dum+1,dum+3)=R(1,3)
         Transformation_M(dum+2,dum+1)=R(2,1)
         Transformation_M(dum+2,dum+2)=R(2,2)
         Transformation_M(dum+2,dum+3)=R(2,3)
         Transformation_M(dum+3,dum+1)=R(3,1)
         Transformation_M(dum+3,dum+2)=R(3,2)
         Transformation_M(dum+3,dum+3)=R(3,3)
      end do
c
      call k_matrix_transpose(Transformation_M,Transformation_M_T,
     $ ndofel,ndofel)
c
      do i = 1, nnode
         coord_l(1,i)=(R(1,1)*co_de(1,i)+R(1,2)*co_de(2,i)
     & +R(1,3)*co_de(3,i))
         coord_l(2,i)=(R(2,1)*co_de(1,i)+R(2,2)*co_de(2,i)
     & +R(2,3)*co_de(3,i))
         coord_l(3,i)=(R(3,1)*co_de(1,i)+R(3,2)*co_de(2,i)
     & +R(3,3)*co_de(3,i))
      end do
c
      return
      end
c=====================================================================
      subroutine k_shape_fun(i,sf)
      INCLUDE 'ABA_PARAM.INC'
      dimension sf(3), GP_coord(2)
c
      if (i .eq. 1) then
         GP_coord(1)= 1.0d0/6.0d0
         GP_coord(2)= 2.0d0/3.0d0
      elseif (i .eq. 2) then
         GP_coord(1)= 2.0d0/3.0d0
         GP_coord(2)= 1.0d0/6.0d0
      elseif (i .eq. 3) then
         GP_coord(1)= 1.0d0/6.0d0
         GP_coord(2)= 1.0d0/6.0d0
      end if
c
      sf(1)= 1.0 - GP_coord(1) - GP_coord(2)
      sf(2)= GP_coord(1)
      sf(3)= GP_coord(2)
c
      return
      end
c=====================================================================
      subroutine k_matrix_multiply(A,B,C,l,n,m)
      INCLUDE 'ABA_PARAM.INC'
      dimension A(l,n),B(n,m),C(l,m)
c
      call k_matrix_zero(C,l,m)
c
      do i = 1, l
         do j = 1, m
            do k = 1, n
               C(i,j)=C(i,j)+A(i,k)*B(k,j)
            end do
         end do
      end do
c
      return
      end
c======================================================================
      subroutine k_matrix_plus_scalar(A,B,c,n,m)
      INCLUDE 'ABA_PARAM.INC'
      dimension A(n,m),B(n,m)
c
      do i = 1, n
         do j = 1, m
            A(i,j)=A(i,j)+c*B(i,j)
         end do
      end do
c
      return
      end
c======================================================================
      subroutine k_matrix_transpose(A,B,n,m)
      INCLUDE 'ABA_PARAM.INC'
      dimension A(n,m),B(m,n)
c
      do i = 1, n
         do j = 1, m
            B(j,i)=A(i,j)
         end do
      end do
c
      return
      end
c======================================================================
      subroutine k_matrix_zero(A,n,m)
      INCLUDE 'ABA_PARAM.INC'
      dimension A(n,m)
c
      do i = 1, n
         do j = 1, m
            A(i,j)=0.d0
         end do
      end do
c
```

```
      return
      end
c=====================================================================
      subroutine k_vector_zero(A,n)
      INCLUDE 'ABA_PARAM.INC'
      dimension A(n)
c
      do i = 1, n
         A(i)=0.d0
      end do
c
      return
      end
c=====================================================================
      subroutine k_Mac(pM,a,b)
      INCLUDE 'ABA_PARAM.INC'
c
      if ((a-b) .GE. 0.0) then
         pM=a-b
      elseif ((a-b) .LT. 0.0) then
         pM=0.d0
      end if
c
      return
      end
c=====================================================================
c=================================END==================================
c=====================================================================
```

References

[1] Dugdale DS. Yielding of steel sheets containing slits. J Mech Phys Solids 1960;8:100–4.
[2] Barenblatt GI. The formation of equilibrium cracks during brittle fracture: general ideas and hypotheses, axially symmetric cracks. J Appl Math Mech 1959;23:255–65.
[3] Park K, Paulino GH. Cohesive zone models: a critical review of traction–separation relationships across fracture surfaces. Appl Mech Rev 2012;64:060802.
[4] Ogden RW. Nonlinear elastic deformations. Courier Dover Publications; 1997.
[5] ABAQUS, Version 6.11 Documentation, 2011. Dassault Systemes Simulia Corp. Providence, RI, USA.
[6] Park K, Paulino GH, Roesler JR. A unified potential-based cohesive model for mixed-mode fracture. J Mech Phys Solids 2009;57:891–908.
[7] Healy B, Gullerud A, Koppenhoefer K, Roy A, RoyChowdhury S, Walters M, et al. WARP3D: 3-D dynamic nonlinear fracture analyses of solids using parallel computers. Tech. rep., Champaign: University of Illinois at Urbana; 2012.
[8] Cerrone A, Heber G, Dacek P, Wawrzynek P, Paulino GH, Ingraffea A. Grain boundary decohesion and particle-matrix debonding in aluminum alloy 7075-t651 using the PPR potential-based cohesive zone model. In: The 11th US national congress on computational mechanics, 2011.
[9] Cerrone A, Tucker J, S.C., Rollet A, Ingraffea A. Micromechanical modeling of rené88dt: From characterization to simulation. In: Joint conference of the engineering mechanics institute and the 11th ASCE joint specialty conference on probabilistic mechanics and structural reliability, 2012.
[10] Cerrone A, Wawrzynek P, Nonn A, Paulino GH, Ingraffea A. Implementation and verification of the Park–Paulino–Roesler cohesive zone model in 3D. Engng Fract Mech, http://dx.doi.org/10.1016/j.engfracmesh.2014.03.010.
[11] Park K, Paulino GH. Computational implementation of the PPR potential-based cohesive model in ABAQUS: educational perspective. Engng Fract Mech 2012;93:239–62.
[12] Bittencourt T, Ingraffea AR, Llorca J. Fracture mechanics of concrete structures. Elsevier Applied Science; 1992 [ch. Simulation of arbitrary, cohesive crack propagation].
[13] Hillerborg A, Modeer M, Petersson PE. Analysis of crack formation and crack growth in concrete by means of fracture mechanics and finite elements. Cem Concr Res 1976;6:773–81.
[14] Petersson PE. Crack growth and development of fracture zones in plain concrete and similar materials. Lund Institute of Technology; 1981.
[15] Song SH, Paulino GH, Buttlar WG. Cohesive zone simulation of mode I and mixed-mode crack propagation in asphalt concrete. Geotech Spec Publ 2005;130:189–98.
[16] Song SH, Paulino GH, Buttlar WG. Simulation of crack propagation in asphalt concrete using an intrinsic cohesive zone model. ASCE J Engng Mech 2006;132:1215–23.
[17] Song SH, Paulino GH, Buttlar WG. A bilinear cohesive zone model tailored for fracture of asphalt concrete considered viscoelastic bulk material. Engng Fract Mech 2006;73:2829–48.
[18] Spring D. Cohesive zone modeling of fracture of sustainable and functionally graded concrete. Master's thesis, Champaign: University of Illinois at Urbana; 2011.
[19] Park K, Paulino GH, Roesler J. Cohesive fracture model for functionally graded fiber reinforced concrete. Cem Concr Res 2010;40:956–65.
[20] Camacho GT, Ortiz M. Computational modelling of impact damage in brittle materials. Int J Solids Struct 1996;33:2899–938.
[21] Espinosa HD, Zavattieri PD, Dwivedi SK. A finite deformation continuum/discrete model for the description of fragmentation and damage in brittle materials. J Mech Phys Solids 1998;46:1909–42.

[22] Pandolfi A, Krysl P, Ortiz M. Finite element simulation of ring expansion and fragmentation: the capturing of length and time scales through cohesive models of fracture. Int J Fract 1999;95:279–97.
[23] Mota A, Klug WS, Ortiz M, Pandolfi A. Finite-element simulation of firearm injury to the human cranium. Comput Mech 2003;31:115–21.
[24] Caballero A, Molinari JF. Finite element simulations of kidney stones fragmentation by direct impact: tool geometry and multiple impacts. Int J Eng Sci 2010;48:253–64.
[25] Ortiz M, Pandolfi A. Finite-deformation irreversible cohesive elements for three-dimensional crack-propagation analysis. Int J Numer Methods Engng 1999;44:1267–82.
[26] Ruiz G, Pandolfi A, Ortiz M. Three-dimensional cohesive modeling of dynamic mixed-mode fracture. Int J Numer Methods Engng 2001;52:97–120.
[27] Paulino GH, Zhang Z. Dynamic fracture of functionally graded composites using an intrinsic cohesive zone model. Mater Sci Forum 2005;492–493:447–52.
[28] Zhang Z, Paulino GH. Cohesive zone modeling of dynamic failure in homogeneous and functionally graded materials. J Plast 2005;21:1195–254.
[29] Klein PA, Foulk JW, Chen EP, Wimmer SA, Gao H. Physics-based modeling of brittle fracture: cohesive formulations and the application of meshfree methods. Tech. rep., Sandia National Laboratories; 2000.
[30] Blal N, Daridon L, Manerie Y, Pagano S. Artificial compliance inherent to the intrinsic cohesive zone models: criteria and application to planar meshes. Int J Fract 2012;178:71–83.
[31] Matouš K, Geubelle PH. Multiscale modelling of particle debonding in reinforced elastomers subjected to finite deformations. Int J Numer Methods Engng 2006;65:190–223.
[32] Hamitouche L, Tarfaoui M, Vautrin A. An interface debonding law subject to viscous regularization for avoiding instability: application to the delamination problems. Engng Fract Mech 2008;75:3084–100.
[33] Zhang Z. Extrinsic cohesive modeling of dynamic fracture and microbranching instability using a topological data structure. Ph.D. Thesis, Champaign: University of Illinois at Urbana; 2007.
[34] Hughes TJR. The finite element method: linear static and dynamic element analysis. Dover Publications; 2000.
[35] Pandolfi A, Guduru PR, Ortiz M, Rosakis AJ. Three dimensional cohesive-element analysis and experiments of dynamic fracture in C300 steel. Int J Solids Struct 2000;37:3733–60.
[36] Tomar V, Zhai J, Zhou M. Bounds for element size in a variable stiffness cohesive finite element model. Int J Numer Methods Engng 2004;61:1894–920.
[37] Falk ML, Needleman A, Rice JR. A critical evaluation of cohesive zone models of dynamic fracture. J Phys IV 2001;11:43–50.
[38] Espinosa HD, Dwivedi S, Lu H-C. Modeling impact induced delamination of woven fiber reinforced composites with contact/cohesive laws. Comput Methods Appl Mech Engng 2000;183:259–90.
[39] Santare MH, Lambros J. Use of graded finite elements to model the behavior of nonhomogeneous materials. J Appl Mech 2000;67:819–22.
[40] Kim J-H, Paulino GH. Isoparametric graded finite elements for nonhomogeneous isotropic and orthotropic materials. ASME J Appl Mech 2002;69:502–14.
[41] Reeder JR, Crews Jr JH. Mixed-mode bending method for delamination testing. Am Inst Aeronaut Astronaut 1990;28:1270–6.
[42] Dávila CG, Camanho PP, de Moura MF. Mixed-mode decohesion elements for analyses of progressive delamination. In: 42nd AIAA/ASME/ASCE/AHS/ASC structures structural dynamics and materials conference; 2001.
[43] Camanho PP, Dávila CG, de Moura MF. Numerical simulation of mixed-mode progressive delamination in composite materials. J Compos Mater 2003;37:1415–24.
[44] Xie D, Waas AM. Discrete cohesive zone model for mixed-mode fracture using finite element analysis. Engng Fract Mech 2006;73:1783–96.
[45] Mi Y, Crisfield MA, Davies GAO. Progressive delamination using interface elements. J Compos Mater 1998;32:1246–72.
[46] Park K. Potential-based fracture mechanics using cohesive zone and virtual internal bond modeling. Ph.D. Thesis, Champaign: University of Illinois at Urbana; 2009.
[47] Leblanc JL. Rubber-filled interactions and rheological properties in filled compounds. Prog Polym Sci 2002;27:627–87.
[48] Ramier J, Chazeau L, Gauthier C. Influence of silica and its different surface treatments on the vulcanization process of silica filled SBR. Rubber Chem Technol 2007;80:183–93.
[49] Qu M, Meth JS, Blackman GS, Cohen GM, Sharp K, Van Vliet K. Tailoring and probing particle–polymer interactions in PMMA/silica nanocomposites. Soft Matter 2011;7:8401–8.
[50] Qu M, Deng F, Kalkhoran SM, Gouldstone A, Robisson A, Van Vliet KJ. Nanoscale visualization and multiscale mechanical implications of bound rubber interphases in rubber–carbon black nanocomposites. Soft Matter 2011;7:1066–77.
[51] Ngo D, Park K, Paulino GH, Huang Y. On the constitutive relation of materials with microstructure using a potential-based cohesive model for interface interaction. Engng Fract Mech 2010;77:1153–74.
[52] Pukánszky B. Interfaces and interphases in multicomponent materials: past, present and future. Eur Polym J 2005;41:645–62.
[53] Ramier J. Comportement mécanique d'élastomères chargés, influence de l'adhésion charge – polymère, influence de la morphologie. Ph.D. Thesis, L'Institut National des Sciences Appliquées de Lyon; 2004.
[54] Lopez-Pamies O, Goudarzi T, Danas K. The nonlinear elastic response of suspensions of rigid inclusions in rubber: II – A simple explicit approximation for finite concentration suspensions. J Mech Phys Solids 2013;61:19–37.
[55] Segurado J, Llorca J. A numerical approximation to the elastic properties of sphere-reinforced composites. J Mech Phys Solids 2002;50:2107–21.
[56] Moraleda J, Segurado J, Llorca J. Finite deformation of incompressible fiver-reinforced elastomers: a computational micromechanics approach. J Mech Phys Solids 2009;57:1596–613.
[57] Schröberl J. NETGEN – an advancing front 2D/3D-mesh generator based on abstract rules. Comput Visual Sci 1997;1:41–52.