Modeling of Debonding and Crack Propagation in Fibrous Composites Using a Discontinuous Galerkin Approach

A Thesis Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Wesley Michael Hicks
May 2015

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Timothy Truster who provided me with this wonderful opportunity to work with him on this exciting research and who has provided me with guidance and encouragement throughout this process. Without his support and generosity none of this would be possible.

I would also like to thank Dr. Z. John Ma and Dr. Dayakar Penumadu for accepting to become members of my committee and the feedback they provided.

Finally, I would like to thank my family who have pushed me all my life to reach my goals.

ABSTRACT

A recently proposed Discontinuous Galerkin (DG) method for modeling debonding and fracture within the context of the finite element method is investigated for problems relating to fiber reinforced composites, namely fiber-matrix interfacial debonding which progresses to matrix fracture. The results are then compared against an existing intrinsic cohesive zone method and a hybrid Discontinuous Galerkin method. The results show that the method can eliminate the problem of artificial compliance that is associated with intrinsic cohesive zone models. Eliminating this stiffness has the positive effect of removing user-defined numerical parameters that are not actually present in the physical system. A key contribution of this work is the demonstration that the DG elements can be inserted between all solid elements in the model, allowing general crack propagation without imposing a predefined surface and without compromising numerical stability.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF VARIABLES

E_{el} - Elastic energy of bulk domain

E_f - Modulus of elasticity of fiber

E_{int} - Cohesive energy of interface

E_m - Modulus of elasticity of matrix

G_c - Griffith critical energy release rate

u - Displacement

$\llbracket \boldsymbol{u} \rrbracket$ - Displacement jump

u^h - Discretized displacement field

W_{ext} - External work on system

α & β - Shape parameters in PPR model

β - Limit proportionality factor in DG method

δ_c - Final separation

δ_{nc} & δ_{tc} - Separation corresponding to $\sigma_{\max}$ & $\tau_{\max}$ in PPR model

ε - Strain

ϕ_n & ϕ_t - Normal and tangential fracture energies in PPR model

γ - Scalar weight used for numerical flux term

Γ_{int} - interface region

λ_n & λ_t - Normal and tangential initial slope indicators for PPR model

ν_f - Poisson's ratio of fiber

ν_m - Poisson's ratio of matrix

$\mathcal{V}_\omega$ & $\mathcal{V}_\gamma$ - Kinematically admissible space of weighting functions

σ - Stress

σ_c - Critical stress in DG method

$\sigma_{\max}$ & $\tau_{\max}$ - Normal and tangential critical stresses in PPR model

σ_x - Axial stress in x-direction

$\{\sigma n\}$ - Average traction, termed numerical flux

Ω - Bulk domain

1 INTRODUCTION AND GENERAL INFORMATION

1.1 Numerical modeling of fracture using cohesive laws

Polymer matrix composites are a material consisting of strands of fibrous material surrounded by an epoxy matrix material. Their unique forming process allows them to have a seemingly limitless number of shapes for a wide variety of applications, from laminates used as repair patches to structural members. The failure mechanisms of these materials is often nucleated from the interface between the fiber and matrix materials, but experimental data about these mechanisms is difficult to obtain since slicing the composite specimen open disturbs the mechanical state and microstructure of the material. Therefore, numerical techniques, such as the finite element method, provide alternatives for predicting these interior failure mechanisms.

Cohesive zone modeling techniques are often utilized to simulate the debonding that happens in composites. The cohesive zone encompasses an area ahead of a crack tip that begins separating on an atomic scale. The development of mechanistic-based models for this region are complicated by the uncertainties of the material behavior on this scale; therefore Barenblatt [1] idealized the mechanical response as a separation between the two surfaces of the crack tip coupled with a stress softening relation. The motivation for this so called cohesive zone model (CZM) comes from the phenomenological damage mechanisms in elastic brittle failure which are caused by the splitting and separation of the two surfaces on an atomic level. At the continuum scale, the separation of the crack tip is resisted by a traction force, hence the name, traction separation laws (TSL).

The cohesive zone modeling approach has also led to popular numerical techniques for simulating crack propagation or debonding in materials. The popularity of this method stems from its ease to implement in existing finite element codes by inserting interface elements along the fracture surface between two layers of solid finite elements. With these interface elements, the crack opening is represented as jumps or discontinuities in the displacement field between neighboring solid elements and is related through the TSL to the interfacial stress at the inter-element boundaries.

In the current state of the art, there are two main numerical realizations of the cohesive zone modeling technique [2]: intrinsic and extrinsic. They differ in the treatment of the material's response before fracture occurs. Intrinsic models assume the material has a reversible separation across the finite element interface that occurs as soon as any force is applied, up until the point of fracture initiation at a critical stress value. This reversible separation is shown in Figure 1.1a. Subsequently, softening of the traction proceeds with increasing separation up to a terminal value at which the traction vanishes and the surface point is fully cracked. In contrast, the extrinsic method assumes the material interface has a rigid response prior to fracture without separation prior to

reaching the critical stress value, as shown in Figure 1.1b. This approach is accomplished by inserting interface finite elements between solid elements only after the stress criteria is satisfied.

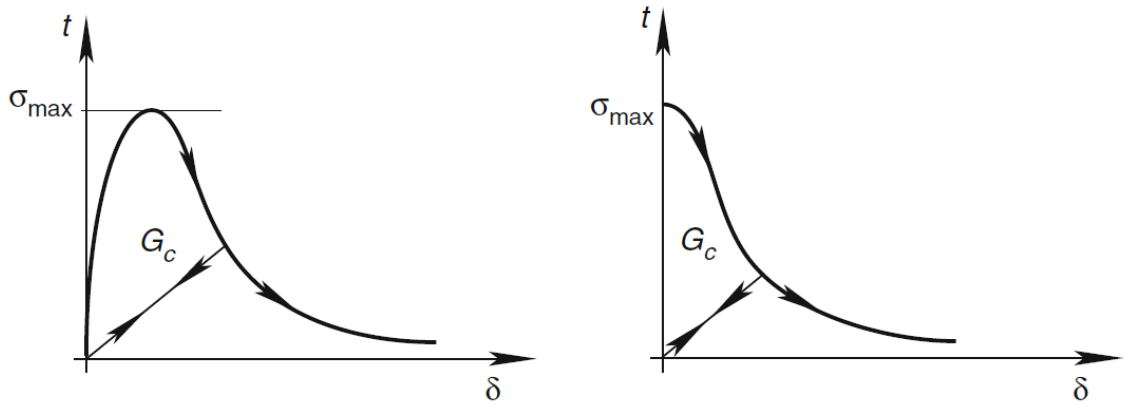


Figure 1.1 - Traction Separation Laws (Reproduction of figure from [2])

(a) Intrinsic TSL **(b) Extrinsic TSL**

Presently, intrinsic cohesive laws are the most common method used for cohesive elements. However, recent advances have provided alternatives to the intrinsic and extrinsic laws by implementing Discontinuous Galerkin (DG) elements [3]This new modeling approach aims to improve upon the current intrinsic and extrinsic approaches by eliminating some of the numerical issues with these laws, such the artificial initial stiffness. This thesis compares the numerical performance of the recently proposed DG method against intrinsic laws and another hybrid DG method within the context of debonding and crack propagation of fibrous composites.

1.2 Intrinsic Method

The functional form or shape of the traction separation law is an important feature in order to provide reasonable approximation of the underlying physical process. One of the earliest intrinsic cohesive laws is the polynomial potential law developed by Needleman [4]. This model employs a polynomial for a cohesive energy density, incorporating both normal and tangential separation at the material interface under the application of tensile and shearing stresses, respectively. Originally, only normal separation was formulated by Needleman [4]; the tangential relation was supplied by Tvergaard [5]. The tangential direction is related to the normal direction using a non-dimensional scalar effective separation. The next intrinsic traction separation law to come about was the exponential potential law. This law was an attempt by Needleman [6] to create a law that stuck better to physical principles and used the universal binding energy law for metallic interfaces and bulk metals.

The intrinsic cohesive zone models described above share a common feature: an initial elastic stiffness between the two sides of the interface. Because these elements consider the traction as a single-valued function of the separation, this requires there to always be a finite separation across the interface whenever a traction field is applied. This initial separation caused by the artificial stiffness is not present in the physical system, since the undamaged bulk material does not contain any surfaces of reduced stiffness but is rather a contiguous body. This artificial stiffness can have negative effects on the numerical simulation in a few ways. When performing dynamic simulations, the artificial stiffness can cause stress waves traveling through the material to propagate at different speeds or in different directions than they would in the actual physical bulk material. The effect of this numerical parameter on how the modeled system behaves must be carefully investigated in order to minimize these issues. In particular, the parameter must be calibrated such that the computed response closely matches to finite element results for meshes without cohesive elements and yet also provide robust results for crack opening simulations, because choosing too high a value for the parameter leads to numerical instabilities [2]. Interestingly, the calibrated parameter may be problem-specific, such that the value must be adjusted when different materials or interface geometry are considered. Finally, the intrinsic models, along with most other methods for modeling fracture in materials, possess the limitation that the crack path is confined to lie along element boundaries, possibly inducing mesh dependency onto the computed crack path. A possible remedy to lessen the impact of this problem is to refine the finite element mesh, although further investigation of this technique is necessary [7].

1.3 Extrinsic Method

The extrinsic method assumes an initially rigid response prior to debonding, meaning that there is no reversible artificial initial stiffness like the one used in the intrinsic approach. This form of TSL is referred to as a linear irreversible softening law and was initially proposed by Camacho and Ortiz [8]. .

Extrinsic laws can also be rate-dependent or rate-independent. Rate-independent laws tend to predict steady-state crack tip speeds much faster than experimental data shows, whereas rate-dependent laws tend to be much more accurate for predicting the crack tip speed. These experiments and simulations show that rate-dependent laws tend to better predict what is happening for dynamic simulations. [9]

The lack of the initial stiffness eliminates the problem of artificial compliance that can be encountered using the intrinsic approach. Another benefit from the initially rigid response before fracture is that interface elements do not have to be present throughout the entire simulation, but can be inserted into the mesh whenever the critical separation stress has been reached. This insertion of new elements, however, can have its drawbacks when performing large scale problems. As cracks form the new elements inserted cause a topological change in the mesh which brings about the need to restructure the data such as stiffness and connectivity matrices.[10] ABAQUS does not

provide the user with the ability to perform these global changes to the mesh which makes it impractical to implement element types that use this method into the ABAQUS code.

1.4 Discontinuous Galerkin Method

The proposed Discontinuous Galerkin method brings together the benefits of each of the methods and eliminates some of the larger problems associated with each of them. This method removes the artificial stiffness introduced in the intrinsic method by weakly enforcing the continuity across the interface before debonding, providing initially rigid interface response. This method also eliminates the need for dynamically inserting elements as they reach the fracture criteria by instead placing them in the initial mesh, which allows problems to be scaled much easier than the extrinsic approach. Of particular importance in this thesis is that the Discontinuous Galerkin elements can be inserted along all inter-element interfaces in the mesh, allowing the crack to propagate more naturally through the model without imposing a failure path a-priori. However, the limitation still remains that the crack path must conform to the element boundaries and cannot pass through elements.

The major sections of this thesis are as follows. The proposed Discontinuous Galerkin method is summarized in Chapter 2. A script for inserting the interface elements between the solid elements of a finite element mesh is described in Chapter 3. Numerical investigations of the proposed DG method follows in Chapter 4. First, a comparison is made against an intrinsic method performed on a composite unit cell with debonding along a prescribed interface in Section 4.1. Second, the method is compared against a hybrid Discontinuous Galerkin method for a simulation of a composite unit cell with debonding along the material interface and cracking throughout the matrix in Section 4.2. Conclusions follow in Chapter 5 with some proposed areas for future work.

2 SUMMARY OF DISCONTINUOUS GALERKIN METHOD

The Discontinuous Galerkin (DG) method proposed by Truster and Masud [11] avoids the artificial compliance problem of the intrinsic cohesive zone methods by adopting a continuum approach that eliminates the need for node-to-node coupling along the interface by using distributed coupling. There are two main variants of the continuum approach for weakly enforcing continuity, Lagrange Multiplier methods and Discontinuous Galerkin methods. The two methods differ according to the treatment of the traction at the interface. Lagrange multiplier methods, such as that of Lorentz [12] developed for modeling fracture in laminated composites, use an additional multiplier field defined at the interface. The DG method eliminates the need for the added multiplier and instead uses the average stress from the bulk domain on both sides of the interface to represent the traction field. Making the DG method a function of the average stress causes it to become a pure displacement method. A summary of the formulation is presented below.

Consider an elastic domain $\Omega \subset \mathbb{R}^3$, containing two materials on either side of an interface. The variational principle associated with the DG method is given below, equation (1). In this equation $\underline{\mathbf{u}}$ is displacement, $\boldsymbol{\delta}$ is the separation, $E_{\text{el}}(\boldsymbol{u})$ is the elastic energy of the bulk domain Ω as a function of $\boldsymbol{u}$, $E_{\text{int}}(\boldsymbol{\delta})$ is the cohesive energy of the interface as a function of $\boldsymbol{\delta}$, and $W_{\text{ext}}(\boldsymbol{u})$ is the external work done on the system as a function of $\boldsymbol{u}$. These energies make up the total potential energy for the system. The penalty parameter r ensures the stability of the formulation. $\{\boldsymbol{\sigma n}\}$ represents the average traction shown in equation (2), henceforth termed as the numerical flux at the interface. In this section, the + and - superscripts denote the restriction of the field to the material on either side of the interface. The average traction is where this DG method differs from the Lagrange multiplier methods. Instead of introducing an additional multiplier into the system the proposed DG method uses information already known within the system, namely the stresses and displacements on either side of the interface, to reduce the number of unknowns in the system. By reducing the unknowns traditional positive-definite solvers, which are used in most solid mechanics codes, are able to solve the discrete problem.

$$L_{DG}(\boldsymbol{u},\boldsymbol{\delta}) = E_{el}(\boldsymbol{u}) + E_{\text{int}}(\boldsymbol{\delta}) - W_{ext}(\boldsymbol{u}) + \int_{\Gamma_{\text{int}}} \{\boldsymbol{\sigma n}\} \cdot (\llbracket \boldsymbol{u} \rrbracket - \boldsymbol{\delta}) d\Gamma + \int_{\Gamma_{\text{int}}} \frac{r}{2} (\llbracket \boldsymbol{u} \rrbracket - \boldsymbol{\delta}) \cdot (\llbracket \boldsymbol{u} \rrbracket - \boldsymbol{\delta}) d\Gamma \tag{1}$$

$$\{\boldsymbol{\sigma n}\} = [\gamma^{+}\boldsymbol{\sigma}^{+}(\boldsymbol{u}^{+}) + \gamma^{-}\boldsymbol{\sigma}^{-}(\boldsymbol{u}^{-})]\boldsymbol{n} \tag{2}$$

In the original version of the DG method for debonding [11], the numerical flux, $\{\boldsymbol{\sigma n}\}$, is defined as the simple average of the stresses from each side of the interface. However, in this thesis, the definition of the numerical flux has been extended by incorporating scalar weights, γ . These weights are evaluated according to the procedure in [13], in which the numerical flux and stability parameters are derived in closed-form through a Variational Multiscale modeling approach. Namely, the fine scales in the finite element model local to the interface are modeled using edge bubble functions, yielding expressions for γ and *r*. The advantage of this approach is that the method is free from user-defined numerical tuning parameters, in contrast to the artificial stiffness in the intrinsic methods, and that the weights are found to be more suitable for coupling materials with highly disparate moduli, such as composite materials.

The weak form of the problem is shown in equations (3) and (4). These are obtained by applying equation (1) to the primary and auxiliary fields. $\mathcal{V}_{\omega}$ and $\mathcal{V}_{\gamma}$ are the kinematically admissible space of weighting functions that satisfy the regularity requirements for equations (3) and (4) respectively. Equation (3) weakly enforces the bulk equilibrium and the condition $\llbracket \boldsymbol{u} \rrbracket = \boldsymbol{\delta}$. Equation (4) enforces the traction separation law for the interface. Assuming that the fiber and matrix are each homogenous and isotropic, equation (5) shows the formula for the stress tensor in each domain, with α = +/- for each, in the same way it was denoted above. $\lambda^{(\alpha)}$ and $\mu^{(\alpha)}$ are Láme parameters for each side of the interface.

$$\begin{aligned} &\int_{\Omega/\Gamma_{\text{int}}} \boldsymbol{\sigma}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{w}) d\Omega + \int_{\Gamma_{\text{int}}} (\llbracket \boldsymbol{u} \rrbracket - \boldsymbol{\delta}) \cdot \{\boldsymbol{\sigma}(\boldsymbol{w})\boldsymbol{n}\} d\Gamma + \\ &\int_{\Gamma_{\text{int}}} [\{\boldsymbol{\sigma}(\boldsymbol{u})\boldsymbol{n}\} + r(\llbracket \boldsymbol{u} \rrbracket - \boldsymbol{\delta})] \cdot \llbracket \boldsymbol{w} \rrbracket d\Gamma = W_{ext}(\boldsymbol{w}) \quad \forall \boldsymbol{w} \in \mathcal{V}_{\omega} \end{aligned} \tag{3}$$

$$\int_{\Gamma_{\text{int}}} [\boldsymbol{t} - \{\boldsymbol{\sigma}(\boldsymbol{u})\boldsymbol{n}\} d\Gamma - r(\llbracket \boldsymbol{u} \rrbracket - \boldsymbol{\delta})] \cdot \boldsymbol{\gamma} d\Gamma = 0 \quad \forall \boldsymbol{\gamma} \in \mathcal{V}_{\gamma} \tag{4}$$

$$\boldsymbol{\sigma}^{(\alpha)}\left(\boldsymbol{u}^{(\alpha)}\right) = \lambda^{(\alpha)} \text{tr}\left[\boldsymbol{\varepsilon}^{(\alpha)}\left(\boldsymbol{u}^{(\alpha)}\right)\right] \boldsymbol{I} + 2\mu^{(\alpha)} \boldsymbol{\varepsilon}^{(\alpha)}\left(\boldsymbol{u}^{(\alpha)}\right) \tag{5}$$

This weak form, equation (3) is discretized using solid finite elements, and the constitutive law equation (4) is enforced at integration points along the interface Γ_{int}. This leads to the traction at each gauss point, subscript "g", along the interface equation (6), where Π is the cohesive potential of the system. With this traction, an implicit function is created for the separation between the interfaces in terms of the bulk domain quantities equation (7). This equation, (7), and the discretized displacement fields, $\boldsymbol{u}^{\text{h}}$ and $\boldsymbol{w}^{\text{h}}$, are then substituted into equation (3), resulting in a nonlinear system of equations for the unknown displacement, equation (8). Further discussion is contained in the reference [11].

$$\boldsymbol{t}_g = \{\boldsymbol{\sigma}(\boldsymbol{u}_g)\boldsymbol{n}\} + r(\llbracket \boldsymbol{u}_g \rrbracket - \boldsymbol{\delta}_g) \in \partial\Pi \tag{6}$$

$$\boldsymbol{\delta}_g = \tilde{\boldsymbol{\delta}}(\llbracket \boldsymbol{u}_g \rrbracket, \{\boldsymbol{\sigma}(\boldsymbol{u}_g)\boldsymbol{n}\}) \tag{7}$$

$$\begin{aligned} &\int_{\Omega/\Gamma_{\text{int}}} \boldsymbol{\sigma}(\boldsymbol{u}^h) : \boldsymbol{\varepsilon}(\boldsymbol{w}^h) d\Omega + \int_{\Gamma_{\text{int}}} (\llbracket \boldsymbol{u}^h \rrbracket - \tilde{\boldsymbol{\delta}}) \cdot \{\boldsymbol{\sigma}(\boldsymbol{w}^h)\boldsymbol{n}\} d\Gamma \\ &+ \int_{\Gamma_{\text{int}}} [\{\boldsymbol{\sigma}(\boldsymbol{u}^h)\boldsymbol{n}\} + r(\llbracket \boldsymbol{u}^h \rrbracket - \tilde{\boldsymbol{\delta}})] \cdot \llbracket \boldsymbol{w}^h \rrbracket d\Gamma = W_{ext}(\boldsymbol{w}^h) \quad \forall \boldsymbol{w}^h \in \mathcal{V}_w^h \end{aligned} \tag{8}$$

A distinguishing feature of the present DG method is that the displacement field $\boldsymbol{u}$ is weakly enforced to be equal to the inelastic gap $\boldsymbol{\delta}$ throughout the entirety of the numerical simulation. Having this unifying condition in both branches of the interface behavior (pre- and post-fracture) makes the method more robust, because the underlying mathematical equations do not sharply change. This property is unique in comparison to other cohesive zone models [14] and DG methods for fracture [2, 7].

The particular constitutive model used in the Discontinuous Galerkin method follows the Generalized Talon-Curnier model for interface behavior, [3, 12, 15] shown below in Figure 2.1 for normal interactions and Figure 2.2 for shear interactions. The model assumes that two sides of the interface remain intact up to a certain critical stress (σ_c). After σ_c is reached, the separation (δ) is allowed to increase between the two faces. Linear softening then takes effect as the separation between the two faces increases. This softening continues until a critical opening (δ_c) is reached and the two surfaces debond completely. If unloading takes place, the model assumes that the separation that has already occurred remains in the model. The model also allows for unequal critical values for normal and shear interactions through a proportionality factor (β) that scales the shear limits relative to the normal limits.

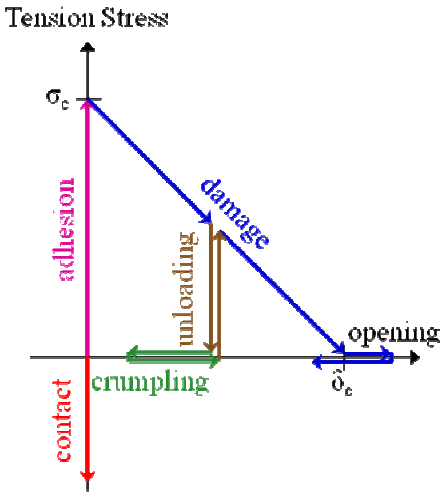


Figure 2.1 - Generalized Talon-Curnier model for interface behavior for normal interaction [11]

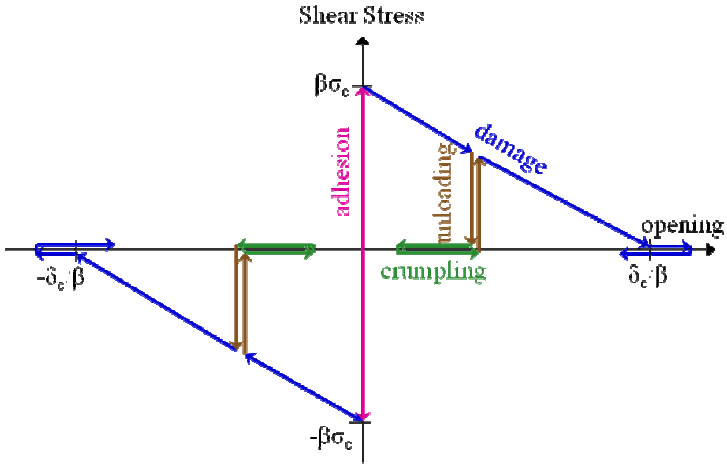


Figure 2.2 - Generalized Talon-Curnier model for interface behavior for shear interaction [11]

In previous numerical studies performed for composites, [11], DG elements had only been placed along the predefined interface between the fiber and matrix. This approximation was justified since the interface between the two materials tends to debond first, negating the composite action of the material. This assumed behavior allowed for straightforward placement of the cohesive elements along the fiber/matrix interface when creating the finite element model.

However, the assumption of debonding localized to the fiber-matrix interface places restrictions on the behavior of the material after significant interface failure. In particular, cracking can spread into the matrix surrounding the fiber as the applied external stress continues to increase [16]. A new contribution from this thesis is the approach whereby the DG elements are inserted between all inter-element boundaries. By

having the zero-thickness interface elements present everywhere across the matrix, the crack can propagate more freely without a predetermined surface. While this idea has been pursued for intrinsic and extrinsic CZM elements, the proposed DG method has not previously been investigated in this regard. Numerous benefits arise from allowing general crack propagation. In particular, the user does not have to inject insight into the engineering model by specifying the surfaces. Rather, the path is obtained as a direct outcome of the simulation, naturally accounting for boundary conditions, geometry, loading history, and other relevant features.

A focus of the numerical investigations in Chapter 4 is to determine if instabilities arise in the computed response when the interface elements are present along all inter-element boundaries. Recall that the DG elements begin to separate when σ_c is reached. An instability can be encountered if numerous element edges reach this value at the same load step, causing a widespread failure of the material and divergence of the nonlinear solution algorithm. However, this instability was not encountered during the simulations of Chapter 4, at least during the primary loading branches. Thus, the DG method appears to be robust for allowing general crack propagation and thus is a viable option for numerical modeling alongside the CZM approaches. Further remarks are given in Chapter 4.

3 INTERFACE ELEMENT INSERTER

The Discontinuous Galerkin (DG) elements presented in Chapter 2 are not currently implemented in the finite element program ABAQUS. Therefore, problems involving the DG elements needed to be executed in a research finite element code written in MATLAB. However, this MATLAB code does not possess a mesh generator, so the features of ABAQUS were used in this regard to create input files for the MATLAB code. The problem information, supplied in this manner, includes the geometry, mesh, element types, boundary conditions and applied loads. A short script was written to process this transfer. Also, ABAQUS does not provide a feature for inserting zero-thickness interface (namely, DG) elements into the mesh, either along preferred material interfaces or between all solid elements. Therefore, an interface element inserter script was developed to insert these elements into the mesh.

After the model files were created in ABAQUS, they needed to be imported into the MATLAB finite element code. This was done by making a small script that extracted the relevant information needed to run the simulation from the ABAQUS text input file. The script reads all of the input information and stores it in a cell array within MATLAB. From this cell array the values were transferred into the format expected by the MATLAB finite element code, such that the simulation could then be performed.

Another part of the script was developed to insert either the DG or the other cohesive elements into the mesh, which operates similarly to the algorithm developed by Nguyen [17]. This insertion only requires a topology change in the mesh, so the originally defined geometry is not affected. Also, the connectivity of the interface is used to determine its location rather than a geometrical description; therefore the procedure is quite general and is not prone to failures due to geometric tolerances. The new code required extra information to be supplied in the ABAQUS model. Sets need to be defined in ABAQUS that correspond to one side of the interface and the interface itself. The interface element inserter then finds the nodes along the defined interface and creates a duplicate of them. Then these duplicated nodes are assigned to the solid elements on the opposing side, and an interface element is created between them. This process is illustrated below in Figure 3.1, where the interface is separated for visual clarity. The new elements are zero-thickness elements because the duplicated nodes are placed directly on top of the existing nodes created in the original ABAQUS mesh. After these are created, the connectivity and node coordinate arrays are updated in the global data structure to include the new nodes and elements. After this step the variables could be exported into the MATLAB finite element code format or into an ABAQUS input text file.

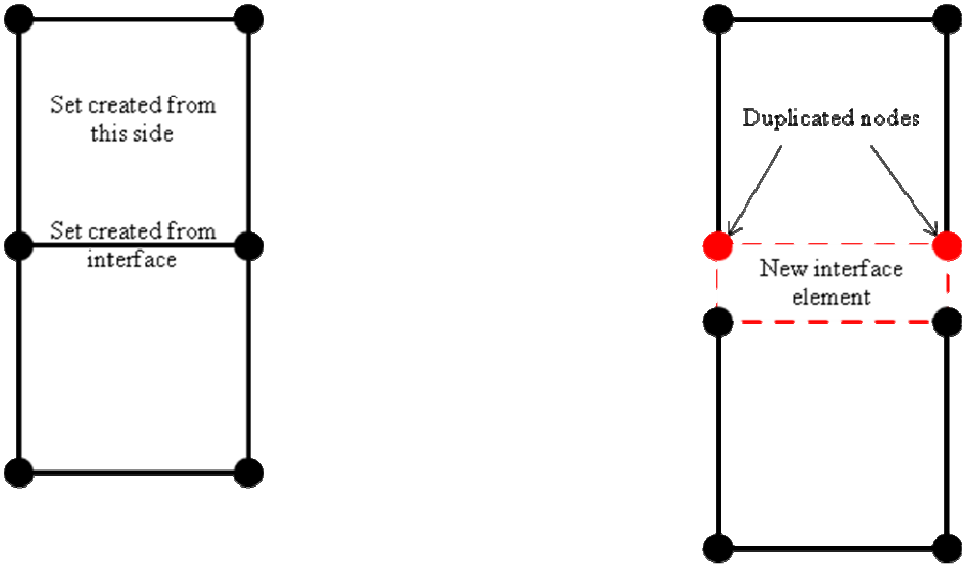


Figure 3.1 - Interface Element Inserter Diagram

The inserting script is capable of handling multiple types of interface elements. In this research, it was used to insert the DG elements described in Chapter 2 and the Park - Paulino - Roesler (PPR) elements described in Section 4.1 [18]. The PPR method required the data to then be output back into ABAQUS format because the MATLAB code did not have these elements implemented.

In previous studies [11] the DG interface elements had been limited to the specific interface between the fiber and matrix of composite materials. This research also extends the insertion, placing the elements along all inter-element boundaries in the matrix which allows for crack nucleation and propagation through this material in addition to debonding along the interface. The associated Matlab script for insertion is a straightforward generalization of the procedure mentioned above.

4 NUMERICAL RESULTS

In this chapter comparisons between the Discontinuous Galerkin method and other methods are performed using problems described in each section. There are two different variations of the cohesive zone method compared against the proposed DG method. The first is the Park - Paulino - Roesler (PPR) method [14]. The next is another version of a hybrid Discontinuous Galerkin method proposed by Nguyen [7].

The problems were all run using a similar process. The finite element models were created in ABAQUS. This is where the geometry, mesh, boundary conditions and applied loads were input. The elements used were linear plane stress or strain and either triangular or quadrilateral. Then with this model an ABAQUS input file was created. This input file is then run through the interface element inserter script to create the cohesive elements. Once these are inserted, the problem was then run in either ABAQUS or MATLAB depending on which kind of cohesive element was used.

4.1 Comparison with PPR model

The problem being analyzed, shown in Figure 4.1, was analyzed before in [11]. It is the quasi-static debonding of a unit cell of fiber-matrix composite material with a fiber radius of 5000 μm and a volumetric ratio of fiber to matrix of 0.2. This made the unit cell's width and height close to 20,000 μm. The model assumes a plane strain condition with a thickness of 1 μm. The material properties of the fiber are $E_f = 210$ GPa & $\nu_f = 0.3$. The matrix material properties are $E_m = 4.6$ GPa & $\nu_m = 0.4$. Symmetrical boundary conditions have been placed on the top, bottom, and left sides, and the right side has a prescribed displacement of 1/2 μm per time step. The simulation ideally runs for 360 time steps, so a total displacement of 180 μm if ABAQUS runs it completely. ABAQUS sometimes did not fully complete the simulation because it subdivides steps when necessary to improve solution convergence, and only lets a max number of steps be defined. This causes some of the simulations to terminate early, so the curves in the following plots do not cover the full range of values.

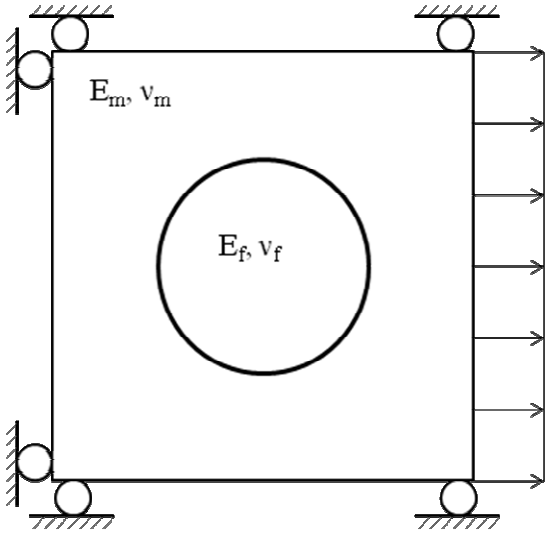


Figure 4.1 - Model setup for PPR method comparison

The mesh, shown in Figure 4.2, is made up of linear plane strain quadrilateral elements in the matrix and fiber regions and PPR or DG elements along the interface between the two materials which is defined as the circumference of the fiber. There are 290 elements and 315 nodes. This mesh is fairly coarse and shows the construction of the model well. A quadrant of each the matrix and the fiber were made along the 45° angles. These two parts were merged and the boundaries were kept. Then that part was copied and rotated around until all four quadrants completed the full unit cell. The mesh was then seeded along the outside edges, the interface between the materials, and along the quadrant boundaries.

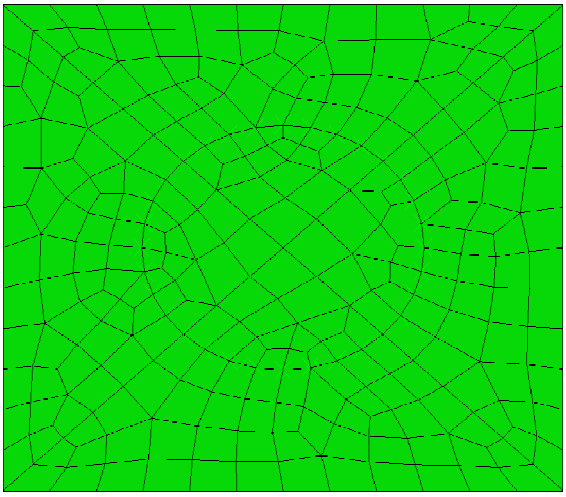


Figure 4.2 - Mesh for PPR method comparison

The basic construction described above was done in ABAQUS but the cohesive elements still needed to be created. This was done by passing the ABAQUS input file through the Matlab interface element inserter script that duplicated the nodes along the fiber interface then created new cohesive elements with the duplicated and original nodes. These new elements could be utilized for any kind of cohesive element. In this case they were made into the Discontinuous Galerkin Method elements and the PPR method elements. Then these two methods were compared.

The PPR method is used in this study as a basis for comparison of the Discontinuous Galerkin Method. The PPR method treats stress as a function of separation meaning that for any stress to occur the model must experience separation along the interface. In reality, the composite interface would be able to transmit stress up to a point without relative separation before starting to debond. The Discontinuous Galerkin Method emulates this behavior by eliminating the initial slope of the function and only inducing separation after the stress reaches the σ_{max} value, which initializes the debonding.

The PPR method has nine input parameters which are grouped into four parameters each for the normal and tangential directions and thickness. The normal and tangential parameters are fracture energy ($\square_n$, $\square_t$), max stress (σ_{max}, τ_{max}), shape factors (α, β), and initial slope factors (λ_n, λ_t). [18] Because the applied displacement load is perpendicular to the interface, the shear stresses projected along the fiber-matrix interface are fairly insignificant. Thus, this investigation focuses on the normal traction-separation behavior. However, the tangential parameters were set equal to the normal parameters for each respective simulation.

This study examines the sensitivity of the material response to the initial slope factors, λ, and the shape factors, α and β. λ is a ratio of the critical opening, δ_c to the final opening, δ ($\lambda = \delta_c / \delta$). The critical opening is the separation that corresponds to the max stress. The final opening is the separation that corresponds to the zero stress. α and β are shape factors that control the descending slope as shown in Figure 4.3 below. α is the shape factor for the normal direction, and β is the shape factor for the tangential direction. [18]

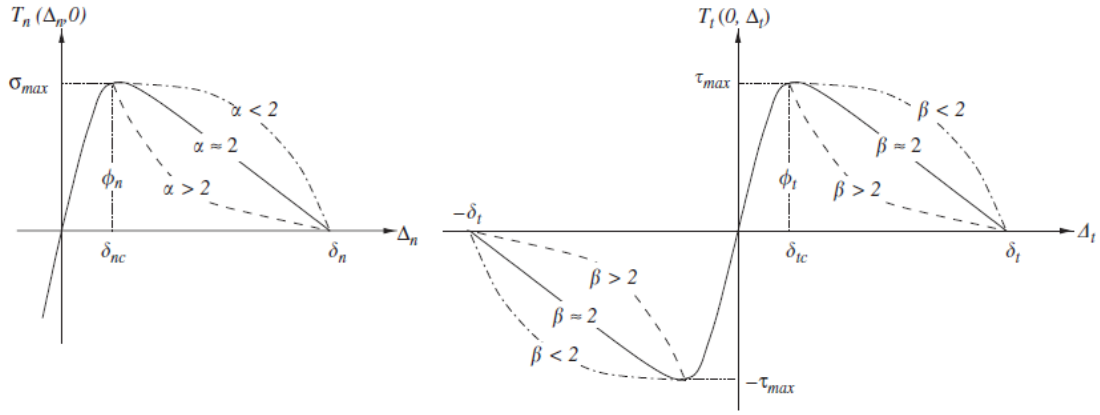


Figure 4.3 - Fracture boundary conditions for the unified mixed-mode potential [14]

Values for the fracture energies were taken to be □ = 0.4 GN/μm which was calculated by taking the area under the stress vs. separation curve, taken as a triangular area so that the integral matches that of the DG method, implying that the fracture energy is identical for both models. That area was bounded by σ_{max} as the height and δ_n = 40 μm for the base. The max stress were assumed as σ_{max} = 20 MPa. The values of λ vary from $0.0002 \leq \lambda \leq 0.2$, and values of α and β from $2 \leq \alpha, \beta \leq 100$. These material parameters ensure that the computed response from the PPR model remains close to that of the proposed DG method.

There are only three parameters needed for the Discontinuous Galerkin Method [11], σ_{max}, δ_c, and β. Between the two methods $\sigma_c = \sigma_{max}$, and $\delta_c = \delta_n$. β is not the same parameter for this method though. It is defined as a proportionality factor between mode I and mode II fracture, which is assumed to be $\beta = 1.0$. The form of the constitutive model is given in Figure 2.1. The key difference between the DG method and the PPR method is the absence of user defined stability parameters in the DG method. These parameters, namely the artificial stiffness, require calibration for each material due to their interactions with the material or constitutive parameters. The DG method only has three material parameters and does not possess the extra shape parameters utilized in the PPR method. This varying shape could have implications on the final opening stress value, which could be shifted depending on the shape of the softening portion of the curve along with the parameter λ. The DG formulation is general; other material models incorporating such shape parameters could be included. One purpose of this study is to determine the sensitivity of the bulk response to such parameters.

In the results presented below, the field response of importance is the relation between macro-scopic stress and strain. These values represent the total strain of the unit cell and the average resulting stress. These values can be computed by two means: (i)

calculating the volume average of the quantities or (ii) calculating the resultant force on the vertical faces and using the prescribed boundary displacement for the strain. The latter approach is used in this study.

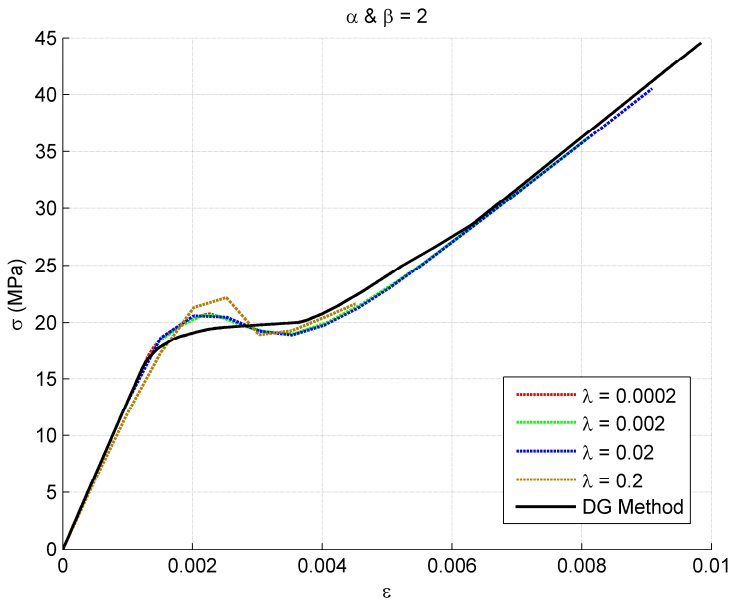


Figure 4.4 - Macro Stress v. Strain for different values of λ when α, β = 2

Figure 4.4 - Figure 4.10 hold a constant value of α, β for each plot, while different λ values are compared. As λ increases there is an increase in the peak stress in the debonding region of the graph. There is also a decrease in the initial slope as λ increases. This indicates that the lower λ, the stiffer the model becomes. The peak stress value is also reached sooner for smaller values of λ. Certain curves do not reach the final applied strain due to the aforementioned issue with ABAQUS requiring smaller steps than specified to make the solution converge.

Note that ABAQUS also has a built-in cohesive zone model which is a simple bilinear traction-separation law. As a preliminary investigation, this constitutive model was also employed. However, the simulations were not able to converge past the initiation of debonding for any chosen sets of material parameters. The initial slope of the bilinear model seems to have a significant effect on the robustness of the numerical method. These observations also shed light on the reason for the PPR model diverging for certain parameter choices. In fact, those which have difficulty converging are the ones which yield a shape for the traction-separation curve which is close to bi-linear. As a final remark, the proposed DG method for debonding does not to be susceptible to these issues, as no convergence issues were encountered when either smaller or larger displacement steps were applied.

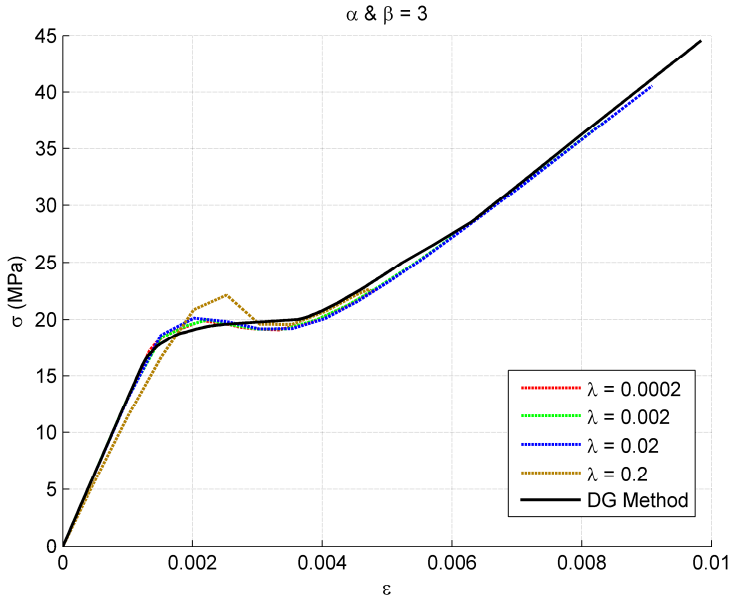


Figure 4.5 - Macro Stress v. Strain for different values of λ when α, β = 3

A key benefit from the DG method is the elimination of the artificial stiffness used in intrinsic methods. To better illustrate this point as well as to interpret the three main branches of the stress-strain curves in the above figures, a simplified model of the various sources of stiffness present in the numerical composite system for both the intrinsic and DG methods is shown in Figure 4.6. The intrinsic method includes the extra interface stiffness; in fact, a spring with infinite stiffness could be envisioned for the DG method in series with the fiber. When the displacement is applied to the side before debonding the stress will tend to gravitate towards the fiber since it is the stiffest element of the system. Then, as the fiber and matrix start to debond, the load will gradually transfer to the matrix. The problem with the intrinsic method is the interface stiffness acts in series with the fiber. Therefore the amount of load the fiber takes prior to debonding is dependent on the stiffness of the interface. The DG method eliminates the interfacial stiffness, thereby allowing direct force transfer between the fiber and matrix. Once debonding initiates, the interface spring stiffness decreases from infinity to a smaller number, and progressively lessens with progressive debonding along the circular interface.

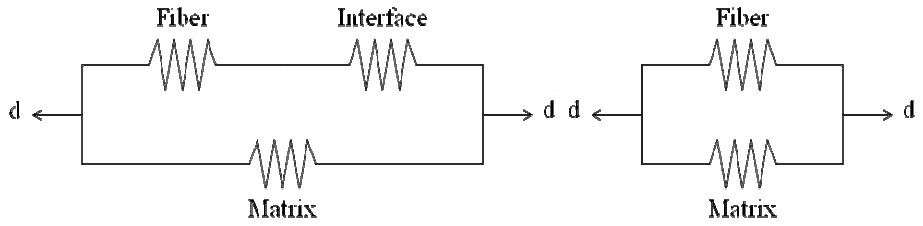


Figure 4.6 - Simplified Stiffness Models

(a) Intrinsic Model **(b) DG Model**

Figure 4.4 - Figure 4.10 also illustrate that the debonding region of the material response ($\varepsilon = 0.0018 : 0.0035$) can be altered greatly by changing the initial slope parameter. The initial slope region can be extended to a greater σ value than the DG method by increasing λ and cause σ to decrease back below the DG method after the debonding region. This variation is caused by additional shape parameter not present in the DG method which are the initial slope, included in all intrinsic methods, and the curvature α. As mentioned, the intent of this study is not to match the material response to experimental data. Therefore, the argument of which CZM fits the data more closely is a moot point. However, in general models with extra parameters also require multiple experiments to perform full calibration.

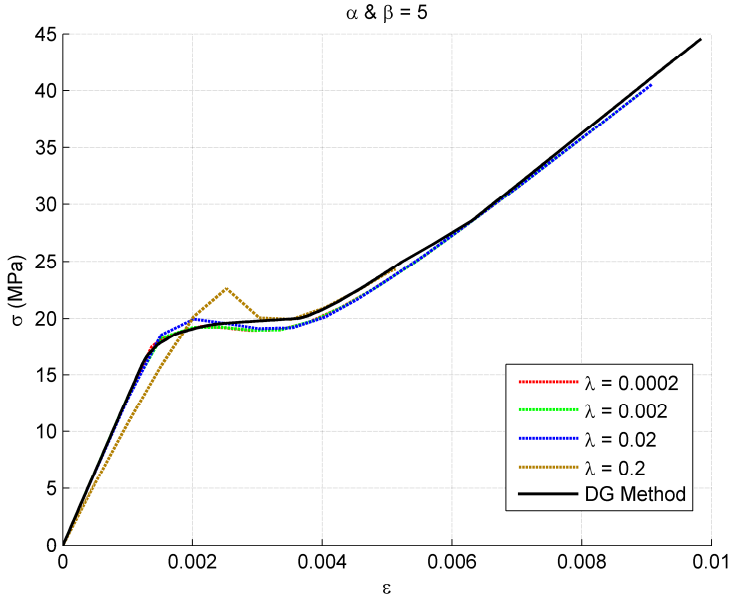


Figure 4.7 - Macro Stress v. Strain for different values of λ when α, $\beta = 5$

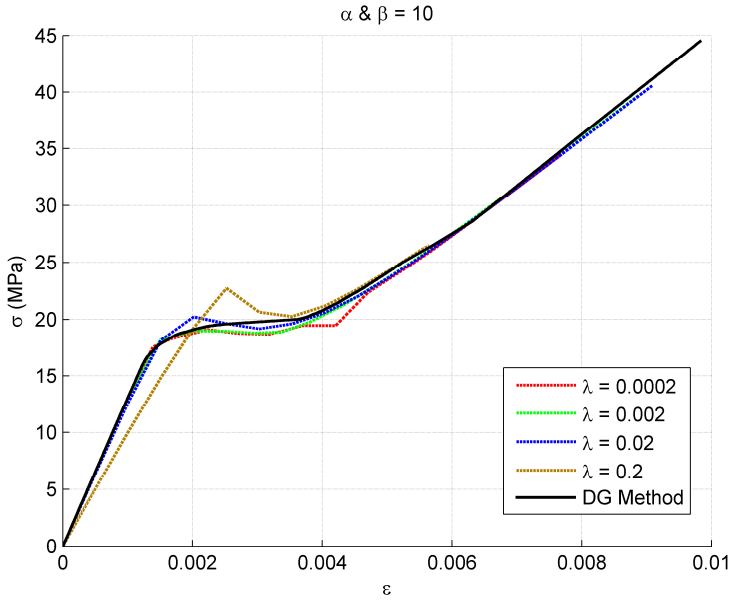


Figure 4.8 - Macro Stress v. Strain for different values of λ when α, β = 10

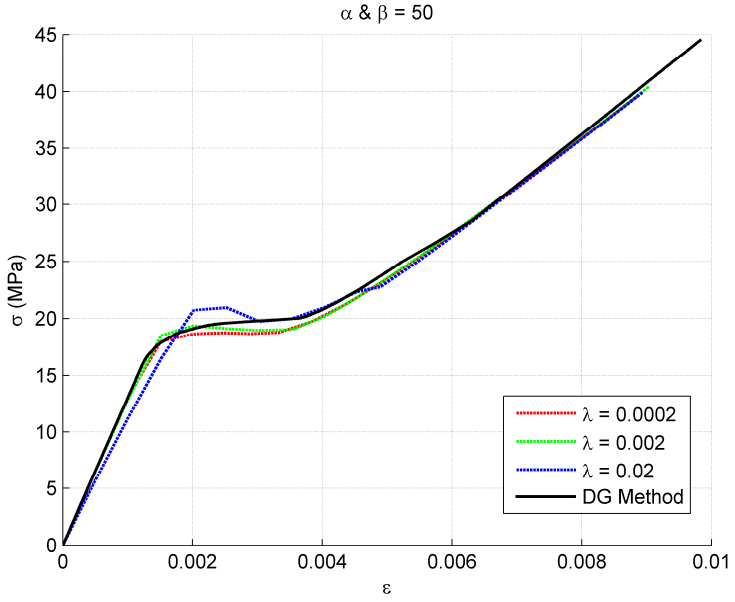


Figure 4.9 - Macro Stress v. Strain for different values of λ when α, β = 50

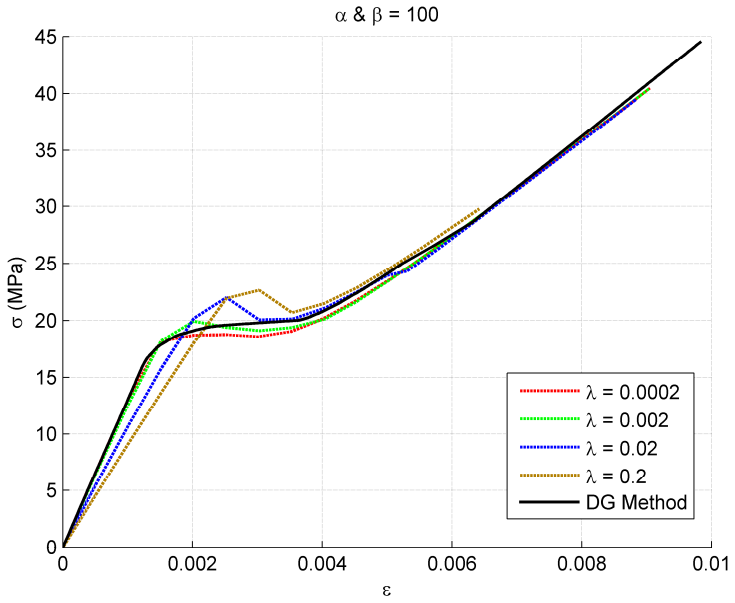


Figure 4.10 - Macro Stress v. Strain for different values of λ when α, β = 100

In Figure 4.4 - Figure 4.10 the peak macro stress of the unit cell shifts outward as the initial stiffness parameter is increased. This is a result of the shallower initial portion of the curve when the fiber and matrix are still bonded, shifting the rest of the curve outwards because of the decrease in slope. These figures also show that the peak macro stress itself increases as the initial stiffness parameter increases. This is because the macro stress corresponding to the point at which σ_c is first reached along the interface has decreased, causing the forces to redistribute into the matrix and less of the load to be taken by the fiber. This means there is a lower stress concentration at the interfacial point where σ_c is reached which means a higher macro stress is necessary to initiate debonding. This higher macro stress is the higher peak stress in these plots.

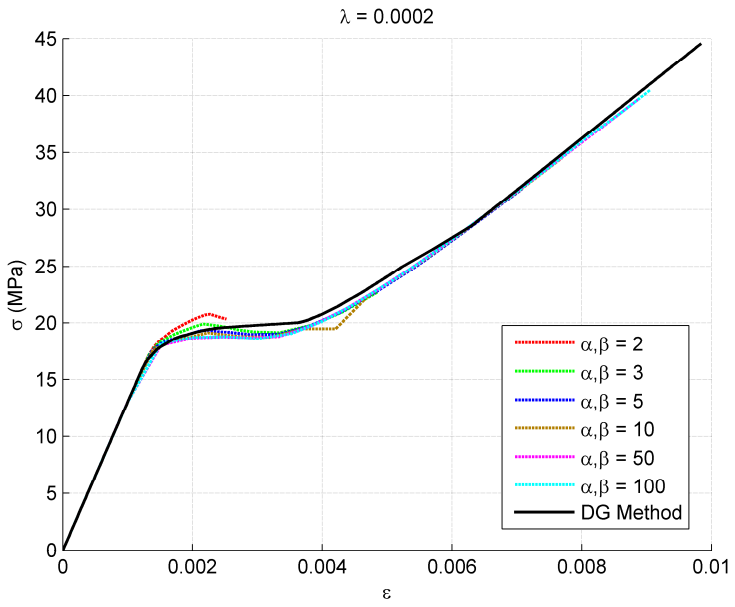


Figure 4.11 - Macro Stress v. Strain for different values of α, β when λ = 0.0002

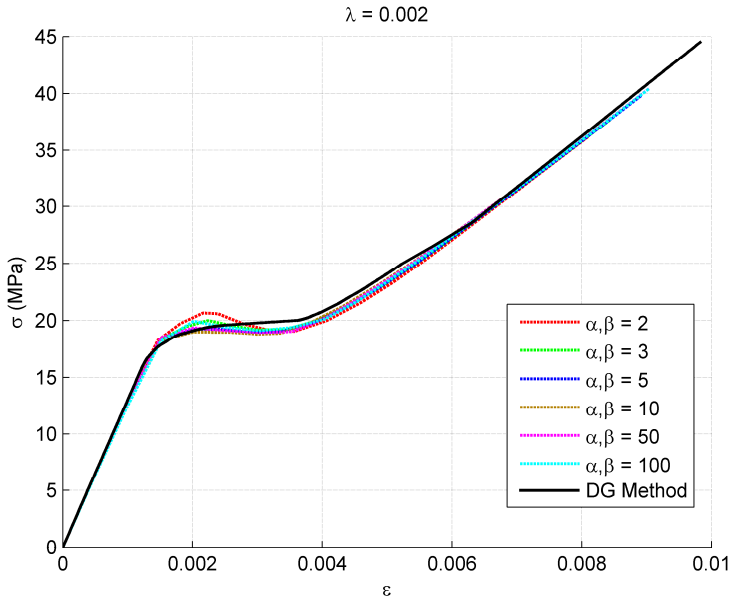


Figure 4.12 - Macro Stress v. Strain for different values of α, β when λ = 0.002

Figure 4.11 - Figure 4.14 hold the λ value constant and examine how different α, β values affect the plot. These figures show that when increasing the shape factors, α,β, the peak stress slightly increases and is reached at larger strains. The initial slopes also indicate that the stiffness increases as the shape factor decreases. This is more apparent with larger values of λ (λ = 0.02 & 0.2) as the lower ones (λ = 0.002 & 0.0002) tend to be consistent with the initial slope of the DG method. The larger λ value in Figure 4.14 emphasizes the decrease in initial stiffness caused by increasing α,β.

Figure 4.11 & Figure 4.12 also show that the choice of lower λ values leads to a material response matching more closely in the debonding region to the response of the DG method, as compared to the larger values in Figure 4.13 & Figure 4.14.

Overall, the DG method is shown to produce the flattest stress-strain curve in the debonding region. Also, the variation in macroscopic stress-strain response is larger for the PPR model with shallow values are employed for the artificial stiffness. These insights can assist the user in making proper calibration of the material parameters for these constitutive models.

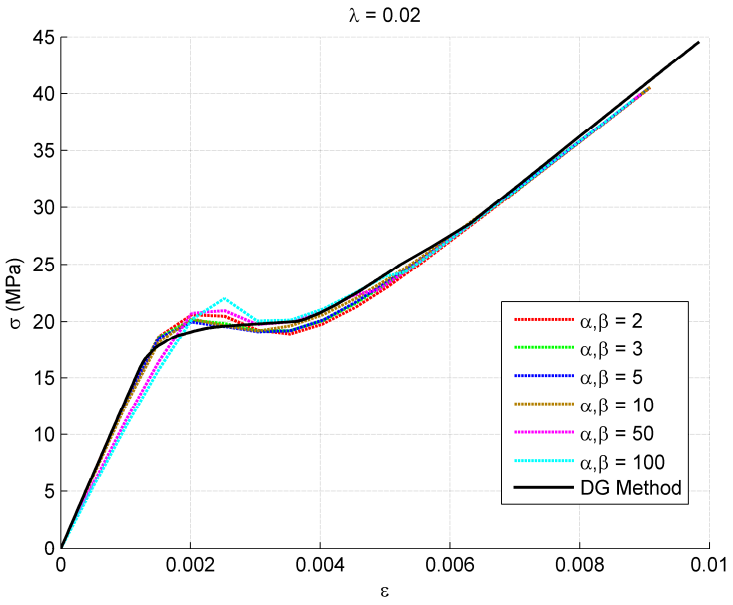


Figure 4.13 - Macro Stress v. Strain for different values of α, β when λ = 0.02

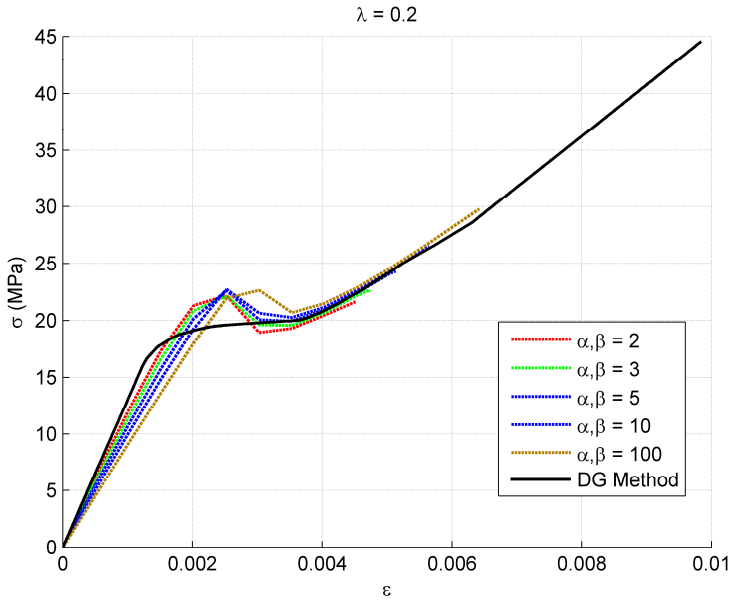


Figure 4.14 - Macro Stress v. Strain for different values of α, β when λ = 0.2

4.2 Unit Cell Matrix Cracking

The next numerical problem serves as a comparison against a hybrid DG method for fracture proposed by Nguyen [7]. This method's distinguishing feature is that it has a pre-fracture formulation and a post-fracture formulation. The pre-fracture form is similar to the DG method proposed by Truster and Masud [11], where displacement continuity is weakly enforced using numerical fluxes. The post-fracture form is a traditional bi-linear cohesive zone law. The two forms are combined into a single unified weak-form that has a binary switch parameter β which toggles between the two stages of pre and post fracture. The parameter is set to $\beta = 0$ before the fracture criterion is met at a point on the interface and subsequently is switched to $\beta = 1$ after the peak stress is exceeding, thereby activating the traction separation law. Further details are contained in [7] and references therein. While the method has been shown to perform well for a variety of crack propagation simulations, the sharp transition in the system of equations could lead to stability issues, such as in the case of implicit dynamics. Recall that for implicit methods, the nonlinear dynamics equations must be solved by the Newton-Raphson

method at each time step; non-smoothness in the system of equations can lead to non-convergence or divergence of this algorithm.

The problem set up for this comparison, shown in Figure 4.15 below, is another fiber reinforced composite unit cell. The unit cell is a 1mm x 1mm square with a fiber in the center that has a diameter of 0.5 mm. This makes the fiber volume fraction 19.6%, which is comparable to the 20% volume fraction for the problem in section 5.1. The unit cell then has a prescribed tensile displacement applied horizontally to both sides at a rate of 2.5 x 10^{-2} μm per time step, running for 120 steps. The bulk material properties are as follows: E_f = 40,000 MPa, ν_f = 0.33, E_m = 4,000 MPa, and ν_m = 0.4.

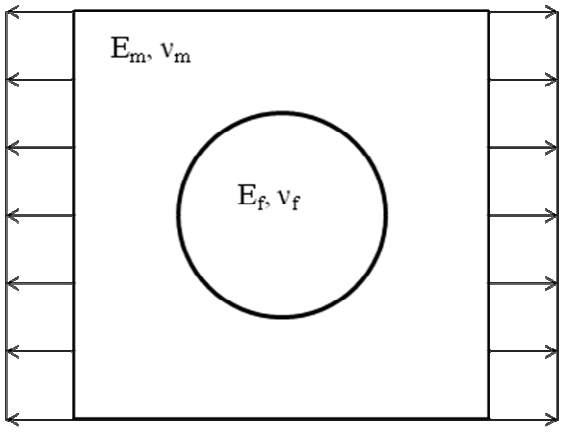


Figure 4.15 - Model setup for hybrid DG method comparison

The mesh, shown in Figure 4.16, was created in ABAQUS and is made up of linear triangular plane strain elements and two dimensional interface elements. It has 10610 nodes and 10490 elements. It was seeded with 25 elements along each edge of the matrix and 132 elements around the fiber edge. This unit cell was constructed differently from the PPR method unit cell though. It abandons the quadrant construction method and is just made of two main parts, the fiber and the matrix. These parts were then merged together in order to create the full composite. In particular, this process led to an unstructured mesh that may help to reduce the bias of matrix crack propagation. The white spaces shown in the figure are the zero-thickness DG elements which are expanded for visualization purposes.

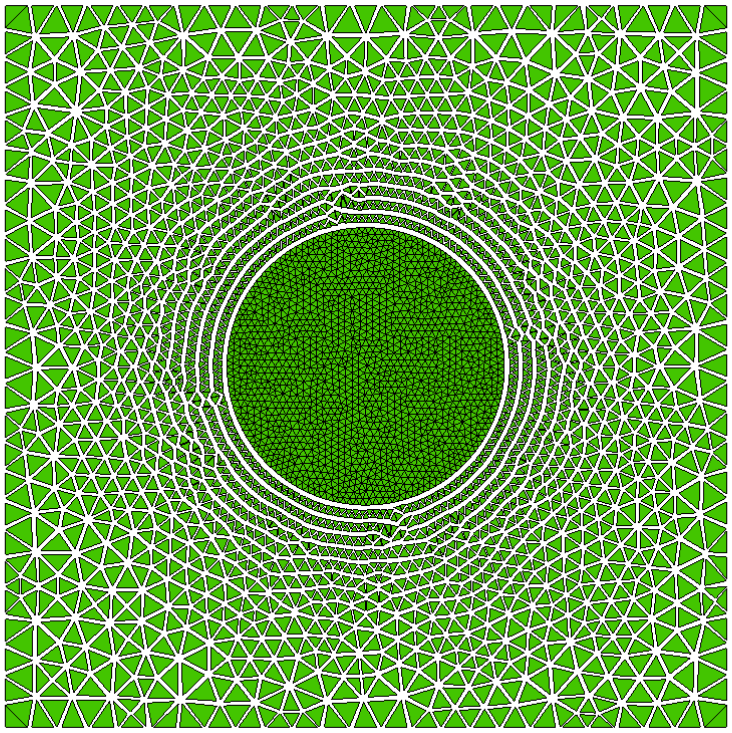


Figure 4.16 - Mesh for hybrid DG method comparison

The purpose of this problem is to investigate the behavior of the proposed DG method for resolving cracking in the surrounding matrix. Similar to the previous problem, DG elements are inserted along the interface between the fiber and matrix using the script in section 4. However, interface elements are also inserted between all elements in the matrix. So that debonding initiates along the composite interface, the critical stress $\sigma_c = 10$ MPa is used at that location, while a value $\sigma_c = 30$ MPa is employed within the matrix material. These values correspond to those in [7].

Figure 4.17 - Force vs Displacement DG Method Comparison below shows the reaction force in the x-direction plotted against the displacement in the x-direction. The total reaction force were found by summing all of the nodal reaction forces along the left edge at each load step. The data for the Hybrid DG method is also presented for comparison. This plot shows the different stages of the loading and subsequent debonding of the model during the simulation; representative stress plots for these stages are given subsequently. During the initial stage, from 0mm - 0.0015mm displacement, the fiber and the matrix are still fully bonded together. At around 0.0015mm the fiber and matrix start to separate, and the slope of the curve decreases as the stress in the model redistributes out of the stiffer fiber and into the less stiff matrix. This branch of the model response is analogous to the debonding simulated in the previous section and may be similarly explained by recalling the reducing interface stiffness in the series-parallel spring diagram in Figure 4.6. As the simulation travels up this portion of the curve, from

0.0015mm - 0.006mm displacement, the slope remains fairly constant but starts to decay as it approaches the peak of the curve. The additional softening is the result of cracking in the matrix, which initiates at about 0.005mm. Then the composite unit cell reaches the peak load at about 0.006mm displacement. The region after the peak is a softening region that occurs because the model starts to fail on a much larger scale, with significant cracking of the matrix material. Currently, the proposed DG method was unable to converge past this peak point at the prescribe load increment, although results are shown for the hybrid DG method past this peak point. Physically, the unit cell has run out of load paths to provide continued increase to the force, and instead many of the open cracks are within the downward sloping region of the traction-separation law from Figure 2.1. The speed at which the cracks begin to spread, between other matrix finite elements, likely requires a much smaller time step to resolve. Also, other nonlinear equation solving techniques could be employed, such as line search or arc-length methods.

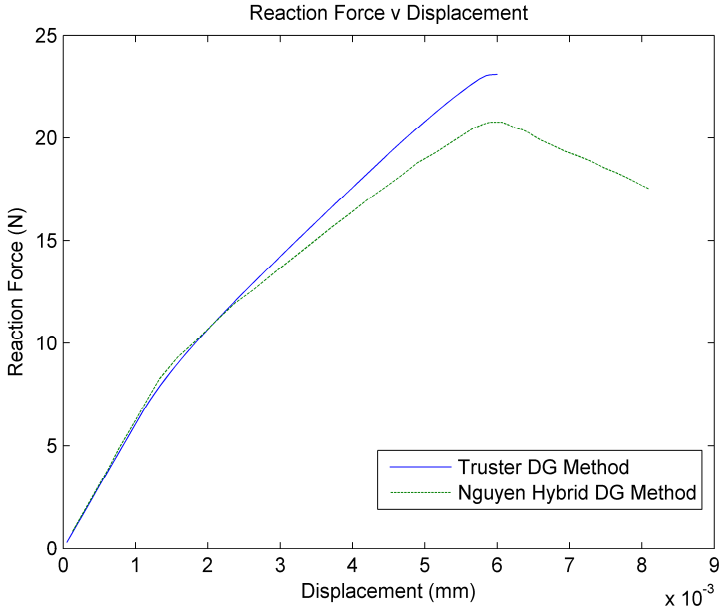


Figure 4.17 - Force vs Displacement DG Method Comparison

The slope of the reaction-displacement curve is steeper in the debonding region for the current method compared to the hybrid DG results. Possible causes for this discrepancy are that the CZM model in the hybrid method possesses a steeper descent compared to the parameters in the Talon-Curnier model. Namely, the initial separation inserted by the Nguyen Hybrid DG method at the instant of fracture initiation at an integration point may lead to a softer response. In contrast, such a jog in the traction-separation curve does not arise in the proposed DG method. Also, the crack paths taken by the two methods are dictated by the edges of the finite element mesh. Thus, further investigation of the mesh-dependence of the crack path is needed.

In Figure 4.18, different stages of the Truster & Masud DG method simulation are visualized through contour plots of σ_x (axial stress component) on the deformed figure of the unit cell. These figures also correspond to the previously described regions of Figure 4.17. The first region prior to debonding is illustrated in Figure 4.18, when the response is purely elastic. The stress is concentrated through the center of the unit cell with the fiber carrying the majority of the burden.

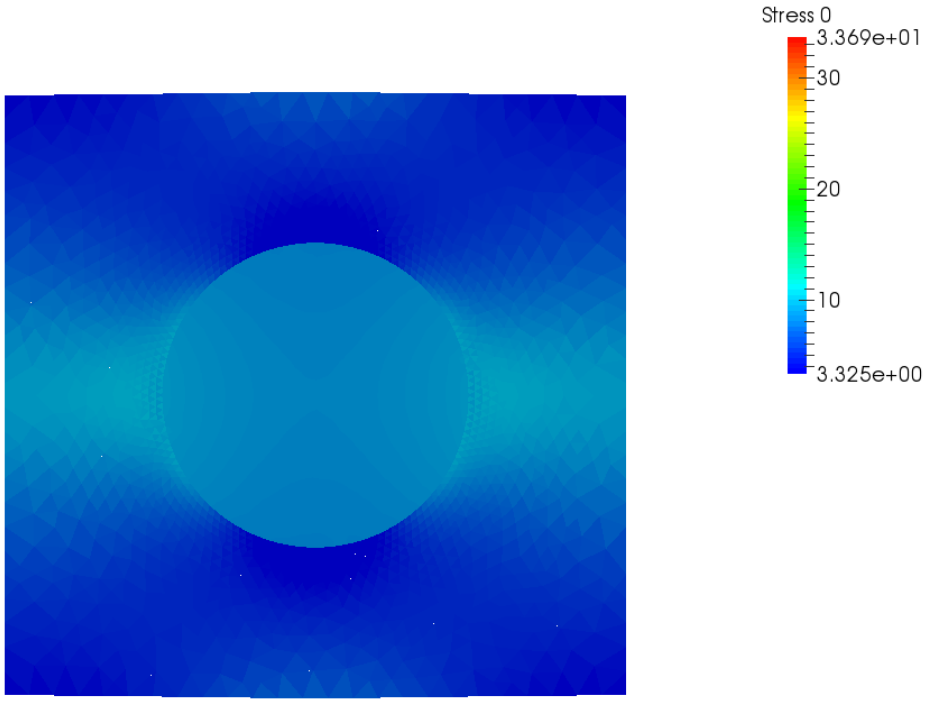


Figure 4.18 - σ_x when displacement = 0.001 mm

Figure 4.19 shows the fiber/matrix interfacial debonding. The debonding starts along the horizontal centerline of the unit cell, where the matrix is the narrowest in the x-direction. Then it starts progressing in both directions along the interface. As the amount debonded increases the stress in the model redistributes. Mild stress concentrations occur at the regions of the crack tip in the model, which has progressed significantly by the displacement level of 0.003mm. Within the fiber, the highest stresses are now seen in the upper and lower portions, near the portions of the interface that remain intact. The fiber is attempting to carry as much of the stress as possible since it is the stiffer element of the composite, but the load can only access the fiber in the region that is still bonded. Notice in particular that the axial stress in the fiber is below the critical stress of 10 MPa in the regions with debonding. The matrix stress is concentrated at and directly behind the tip because the tip is the first place the load can get into the fiber and the area behind this has a concentration, as mentioned above.

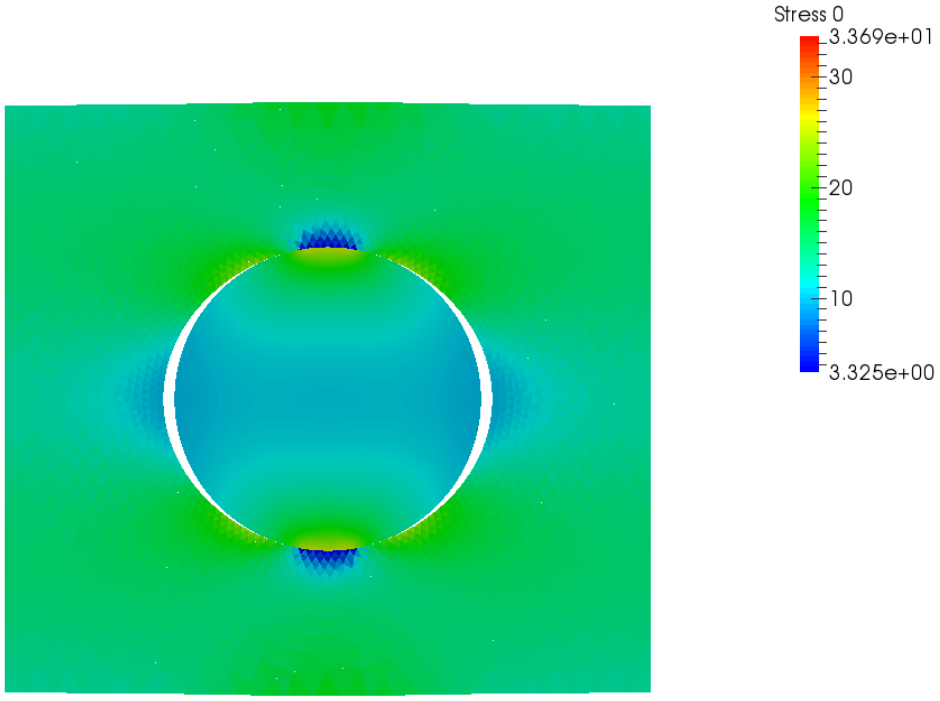


Figure 4.19 - σ_x when displacement = 0.003 mm

In Figure 4.20 the fiber and matrix debonding has progressed significantly, and now cracking has started to become visible in the matrix. This point is where the crack kinking phenomenon occurs, where the crack at the interface turns in toward the matrix ligaments [19]. The stress has reached a point along the interface where is has become difficult to transfer into the fiber so most of the load is carried by the matrix, and a little is able to be transferred into the top and bottom of the fiber, but this is relatively small now compared to the matrix stress. The area of the matrix on the side of the crack nearest the bonded region of the interface has the highest concentration of stress because it is trying to still distribute itself into the fiber; this is likely a numerical artifact. More importantly, concentrated stresses of about 30 MPa are present at the tips of the kinked cracks. Recall that this is the value of the matrix critical stress; hence the numerical model is correctly predicting the onset of the matrix cracks. Also, the crack path is purely a result of the nonlinear calculations and is not a-priori imposed onto the model in any way. In fact, the elements with vertical edges experience the largest normal stress, compared to those at an angle to the applied horizontal load. Thus, the interfaces between these elements are the first to reach the peak stress compared to edges that are inclined.

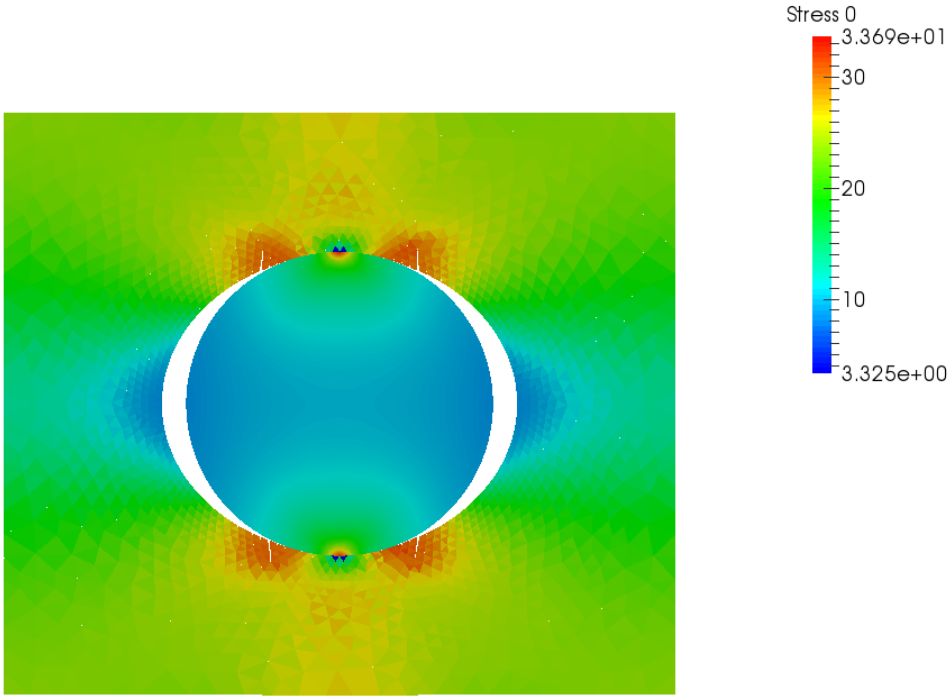


Figure 4.20 - σ_x when displacement = 0.005 mm

Figure 4.21 shows the unit cell when it has reached its peak loading. At this point the matrix is performing essentially as a plate with a hole with cracking in the ligaments. A small area of the interface remains bonded, but once the matrix starts cracking not much load can be transferred into the fiber. While the initial cracks branching outward from the fiber are still prominent, other single cracks have begun to propagate from the boundaries of the unit cell toward the fiber region. These paths are not symmetrical about the horizontal centerline as compared to the internal cracks. Thus, the edges of the finite element mesh, again, likely played a role in determining these initiation sites. The crack also propagates down from the top of the unit cell as well as progressing further away from the interface. Other small cracks are also appearing in neighboring layers of elements, also running in the vertical direction. These cracks forming start a cataclysmic failure where a great deal of elements reach σ_c at the same time causing the solution to diverge. Nonetheless, the general patterns of cracks match closely to the results and figures presented by Nguyen [7]. Therefore, the performance and predicted response of the two methods seems to be in good comparison for this model with identical geometry and material parameters.

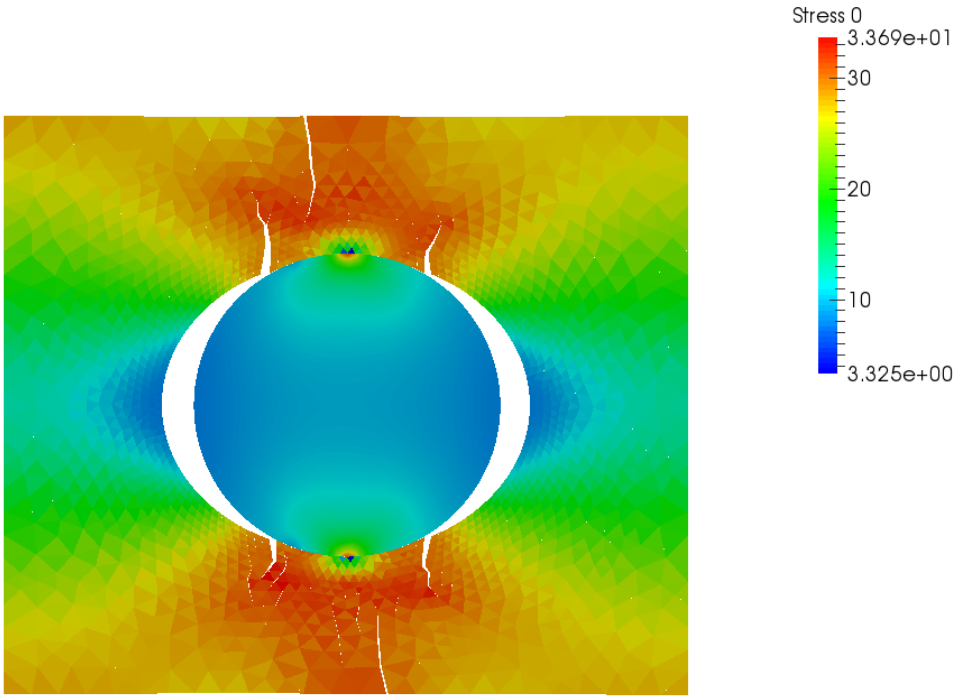


Figure 4.21 - σ_x when displacement = 0.006 mm (at peak force)

5 CONCLUSIONS AND RECOMMENDATIONS

Debonding and fracture simulations are an important step toward understanding how and when materials and ultimately structures fail. Intrinsic cohesive zone methods have been very prominent within the landscape of the finite element fracture simulations commonly employed in the literature. They do have particular drawbacks, in particular the issue of artificial compliance, which can lead to inaccuracies in the computed results because of the extra energy introduced by placing an artificial stiffness in the system. A recently proposed Discontinuous Galerkin (DG) method [11] for fracture mechanics aims to eliminate that stiffness, thereby restoring fidelity to the underlying material process and also lessening the number of unknown parameters which the user must calibrate. This DG method also avoids the mesh topological changes required for extrinsic cohesive zone models because the interface elements are present during the entire numerical simulation, allowing the method to be easily incorporated into existing finite element codes. Within this thesis, this DG method was compared against the PPR cohesive zone model and another hybrid DG method for problems relating to composites in the context of interfacial debonding and matrix cracking.

In these comparisons, the Truster and Masud [11] DG method performed very well against the other methods. The results support that the method eliminates the need for an artificial stiffness parameter, thereby removing all user defined parameters that are not present in the physical system. A new contribution of this thesis was showing that the DG elements can be inserted throughout the finite element model, enabling the crack path to be determined as an outcome of the numerical simulation without predefining a specific path within the model. The method should be considered as a viable alternative to the commonly used intrinsic method in most finite element codes today.

The insights and conclusions from the results presented in this work provide opportunities for future work. For example, the sensitivity of the crack paths computed in section 4.2 to the topology of the finite element mesh should be investigated in greater detail, both with respect to mesh refinement along with the degree of structure in the mesh. Also, the present results are for quasi-static problems. The issue of artificial compliance becomes more critical for dynamic fracture problems. Hence, this method should be further tested on problems containing dynamic loading.

The unit cell problem is also a simplified problem within the context of composites. Further testing on the method's robustness could be performed within this context. Knowing that the method’s performance on composite materials is satisfactory, different variations of composite material problems could be explored to further examine the mechanisms of damage and fracture. For example, the progressive debonding of unit cells with different fiber shapes or volume fractions could be examined. Also, larger unit cells could be considered, containing multiply fibers in a non-uniform arrangement. The

response function, namely the macro stress-strain curve, could be compared between these cases and assessed with respect to the single fiber case studied herein.

LIST OF REFERENCES

[1] G.I. Barenblatt, The mathematical theory of equilibrium cracks in brittle fracture, Advances in Applied Mechanics, 7 (1962) 55-129.
[2] A. Seagraves, R. Radovitzky, Advances in Cohesive Zone Modeling of Dynamic Fracture, in, Springer, 2010, pp. 349-405.
[3] T.J. Truster, A. Masud, A Discontinuous/continuous Galerkin method for modeling of interphase damage in fibrous composite systems, Computational Mechanics, 52 (2013) 499-514.
[4] A. Needleman, A Continuum Model for Void Nucleation by Inclusion Debonding, J. Appl. Mech., 54 (1987).
[5] V. Tvergaard, Effect of Fibre Debonding in a Whisker Reinforced Metal, Materials Science and Engineering: A, 125 (1990) 203-213.
[6] A. Needleman, An analysis of decohesion along an imperfect interface, International Journal of Fracture, 42 (1990) 21-40.
[7] V.P. Nguyen, Discontinuous Galerkin/extrinsic cohesive zone modeling: Implementation caveats and applications in computational fracture mechanics, Engineering Fracture Mechanics, 128 (2014) 37-68.
[8] G.T. Camacho, M. Ortiz, Computational modelling of impact damage in brittle materials, International Journal of Solids and Structures, 33 (1996) 2899-2938.
[9] F.M. Zhou, JF; Ramesh, KT, A cohesive model based fragmentation analysis: effects of strain rate and initial defects distribution, International Journal of Solids and Structures, 42 (2005) 5181-5207.
[10] O.M. Pandolfi A, An efficient adaptive procedure for three-dimensional fragmentation simulations, Eng Comput, 18 (2002) 148-159.
[11] T.J. Truster, A. Masud, A Discontinuous/continuous Galerkin method for modeling of interphase damage in fibrous composite systems, Computational Mechanics, 52 (2013) 499-514.
[12] E. Lorentz, A mixed interface finite element for cohesive zone models, Computer Methods in Applied Mechanics and Engineering, 198 (2008) 302-317.
[13] T.J. Truster, A. Masud, Primal interface formulation for coupling multiple PDEs: A consistent derivation via the Variational Multiscale method, Computational Methods in Applied Mechanics and Engineering, 268 (2014) 194 - 224.
[14] K. Park, G.H. Paulino, J.R. Roesler, A unified potential-based cohesive model of mixed-mode fracture, Journal of the Mechanics and Physics of Solids, 57 (2009) 891-908.
[15] C. Talon, A. Curnier, A model of adhesion coupled to contact and friction, European Journal of Mechanics - A/Solids, 22 (2003) 545-565.
[16] S. Li, S. Ghosh, Modeling interfacial debonding and matrix cracking in fiber reinforced composites by the extended Voronoi cell FEM, Finite Elements in Analysis and Design, 43 (2007) 397-410.
[17] V.P. Nguyen, An open source program to generate zero-thickness cohesive interface elements, Advances in Engineering Software, 74 (2014) 27-39.
[18] K. Park, G.H. Paulino, Computational implementation of the PPR potential-based cohesive model in ABAQUS: Educational perspective, Engineering Fracture Mechanics, 93 (2012) 239-262.

[19] F.C. Paris, E; Mantic, V, Kinking of Transversal Interface Cracks Between Fiber and Matrix, JOURNAL OF APPLIED MECHANICS-TRANSACTIONS OF THE ASME, 74 (2007) 703-716.

APPENDIX

The following is a copy of the element inserter written to create the input files for these simulations, and the input values for a simple two element mesh where the interface element will be created in between them. The first section is the 3 sections of the code, the next is the input file from ABAQUS. After that is the initial and final input values in the research code format. The last page has a figure showing the mesh pre and post insertion. Nodes 7 and 8 are directly on top of the original nodes 1 and 2.

```
% input: interface number(numfiber)
% output: output_data, new_connectivity_data, new_node_table_data,
updated
% node_table.


[filename,pathname] = uigetfile('*.inp', 'Select Abaqus .inp file');
fid = fopen([pathname filename]);

% skip 8 lines
% InputText=textscan(fid,'%s',8,'delimiter','\n'); % Read strings
delimited by a carriage return

% Search for node list
testchar = char('*Node');
line = fgetl(fid);
while strcmp(testchar,line) == 0
    line = fgetl(fid);
end

Block = 1;
j=1;
headerData{Block,j}=line;

% Read in node data
FormatString=repmat('%f',1,3);  % Create format string based on
parameter  ***changed from hex 4->2
InputText=textscan(fid,FormatString,'delimiter',','); % Read data block
Data{Block,j}=cell2mat(InputText); % Convert to numerical array from
cell

line=fgetl(fid);
testfront = strtok(headerData{Block,j},',');
front = strtok(line, ',');
if strcmp(testfront,front) == 0
      j=2;
else  Block=Block+1;
end

headerData{Block,j}=line;

% Read in element data
```

```
FormatString=repmat('%f',1,5);  % Create format string based on
parameter  ***changed from hex 9->5
InputText=textscan(fid,FormatString,'delimiter',','); % Read data block
Data{Block,j}=cell2mat(InputText); % Convert to numerical array from
cell
%[NumRows,NumCols]=size(Data{Block}); % Size of table


testchar = char('*Nset');
while strcmp(testchar,front) == 0
    line = fgetl(fid);
    front= strtok(line, ',');
end


testend=char('*End Assembly');
front_pre=1; check_Block_pre=1;
%read in the rest data block by block
while strcmp(testend,front) == 0 && feof(fid) == 0
    [front,rem]=strtok(line,',');
    Second=strtok(rem,',');
    [ntoken,check_Block]=strtok(Second,'=');
    if strcmp(testchar,front) == 1;
       j=1;
    else j=2;
    end
    if strcmp(check_Block,check_Block_pre)==0;
        Block=Block+1;
    end

    headerData{Block,j}=line;
    [front_pre,rem_pre]=strtok(line,',');
    Second_pre=strtok(rem_pre,',');
    [ntoken,check_Block_pre]=strtok(Second_pre,'=');

    FormatString=repmat('%f',1,16);  % Create format string based on
parameter  ***possibly needs to be changed from hex
    InputText=textscan(fid,FormatString,'delimiter',','); % Read data
block
    Data{Block,j}=cell2mat(InputText); % Convert to numerical array
from cell

    line=fgetl(fid);
    front=strtok(line,',');
    j=1;

end           % Node stuff in col 1
              % Elem stuff in col 2 always
              % pair up things with the same name, put different ones
in new rows


NumBlock=Block;
```

```
%set node_table = Data{1,1}, connectivity = Data{1,2}
node_table = Data{1,1};
connectivity = Data{1,2};
org_connectivity= Data{1,2};

%find Nset, Elset for interface
%find Efiber set
numfiber=1;

whereheader=zeros(numfiber,2);
contact_check=char('*Nset, nset=interface');
Efiber_check=char('*Elset, elset=Efiber');
generate_check= char(', generate');
for Block=2:NumBlock
    for j=1:2
        line=headerData{Block,j};
        endline=length(line);

        [front,rem]=strtok(headerData{Block,j},',');
        [Second,rest]=strtok(rem,',');
     if strcmp(generate_check,rest)==1      %reformatting generate
statement into one list
         fprintf('Data{% d,% d} is generate type \n', Block,j)

Data{Block,j}=((Data{Block,j}(1):Data{Block,j}(3):Data{Block,j}(2)));
     else    %change rectangular data into one long list
             n_elset=Data{Block,j}';
    N_elset=reshape(n_elset,size(n_elset,1)*size(n_elset,2),1);
    N_elset = N_elset(~isnan(N_elset));                    %cut out NaN
    Data{Block,j} = N_elset;
     end

     if strcmp(contact_check,line(1:min(21:endline)))==1
         fprintf('interface Data is in Block %d\n',Block)
         n=str2num(line(22));
         whereheader(n,1)=Block;
     end
     if strcmp(Efiber_check,line(1:min(20:endline)))==1
         fprintf('Efiber set is in Block %d\n',Block)
         n=str2num(line(21));
         whereheader(n,2)=Block;     % put interface in col 1 & Efiber
in col 2
     end                             % put interface1 & Efiber1 in
row1,exc.
    end
end

% define the input that You Li's code need and
% call You Li's code i times(i=numfiber)
 i_fiber=1;

    nset=Data{whereheader(i_fiber,1),1};
    Nset=reshape(nset,size(nset,1)*size(nset,2),1);
    Nset = Nset(~isnan(Nset));                    %cut out NaN
```

```
    elset=Data{whereheader(i_fiber,1),2};
    Elset=reshape(elset,size(elset,1)*size(elset,2),1);
    Elset = Elset(~isnan(Elset));                              %cut out
NaN
    efiber=Data{whereheader(i_fiber,2),2};
    Efiber=reshape(efiber,size(efiber,1)*size(efiber,2),1);
    Efiber=Efiber(~isnan(Efiber));                    %cut out NaN

    getsurf2D_quad
    % num_n = total number of nodes
    num_n = size(node_table,1);
    % num_n will change through updating new coordinates for nodes.
    size_n = num_n;
    getnew2D_quad

    output_data=output;
%     new_connectivity_data=new_connectivity;
%     new_node_table_data=new_node_table;
    connectivity(new_connectivity(:,1),:) = new_connectivity;
    node_table = [node_table; sortrows(new_node_table,1)];
%     Nset = [Nset; sort(new_node_table(:,1))];

if numfiber>1
   for i_fiber=2:numfiber

    nset=Data{whereheader(i_fiber,1),1};
    Nset=reshape(nset,size(nset,1)*size(nset,2),1);
    Nset = Nset(~isnan(Nset));                    %cut out NaN

    Nset = [Nset; node_table(Nset,1)];
    Nset = unique(Nset);
    elset=Data{whereheader(i_fiber,1),2};
    Elset=reshape(elset,size(elset,1)*size(elset,2),1);
    Elset = Elset(~isnan(Elset));                 %cut out NaN
    % efiber=Data{whereheader(i_fiber,2),2};
    % Efiber=reshape(efiber,size(efiber,1)*size(efiber,2),1);
    % Efiber=Efiber(~isnan(Efiber)),                    %cut out NaN

    getsurf2D_quad
    getnew2D_quad

    % put allthe output data in 1 col, all new_connectivity data in one
    % col, & all New nade table data in one col.
    output_data=[output_data;output];
%     new_connectivity_data=[new_connectivity_data;new_connectivity];
%     new_node_table_data=[new_node_table_data;new_node_table];
    connectivity(new_connectivity(:,1),:) = new_connectivity;
    node_table = [node_table; sortrows(new_node_table,1)];
    allnewnodes = unique(node_table(size_n+1:end,:),'rows');
    node_table = [node_table(1:size_n,:); allnewnodes];
   end
end
```

```
% sort of the new nodes and add into node_table
  allnewnodes = unique(node_table(size_n+1:end,:),'rows');
  node_table = [node_table(1:size_n,:); allnewnodes];


fclose(fid);
```

```
% getsurf2D_quad
% created by: Wes Hicks
% source: You Li "getsurf3D_hex"
% output: for 2D quad mesh, find which nodes and faces of which
elements are
% contacting on the interested inner surface.
% input connectivity,Nset(column vector),Elset(column
vector),node_table,Efiber
%
% step 1: find which elements are contacting the surface
contact = org_connectivity(Elset,:);
% contact = [e n1 n2 n3 n4]

% step 2: create empty output files
M = zeros(length(Elset),4);
% M = [ e n1 n2 face], all elements, nodes and faces on contact surface
output = zeros(length(Elset)/2,4);
% output = [e1 e2 face1 face2], pairs of contacting elements and faces

% step 3: fill out M
% n is the row number
n = 1;
face1 = [1,2];
face2 = [2,3];
face3 = [3,4];
face4 = [4,1];
match = [1,1];
% for each element on contacting surface, find which of its nodes are
on
% the contacting surface.
for l = 1:size(contact,1)
    one_zero = ismember(contact(l,2:5),Nset);
    index = find(one_zero);
    if ismember(index,face1) == match
        F1 = 1;
    elseif ismember(index,face2) == match
        F1 = 2;
    elseif ismember(index,face3) == match
        F1 = 3;
    elseif ismember(index,face4) == match
        F1 = 4;
    end
    M(l,:)=[contact(l,1),contact(l,(index+1)),F1];
end

% step 4: fill out output
% find which pairs of elements are connecting each other with which
faces.
for i = 1:(size(contact,1)-1)
    for j = (i+1):size(contact,1)
        check = ismember(M(i,2:3),M(j,2:3));
        if check == match
             if ismember(M(i,1),Efiber)
```

```
                output(n,:) = [M(j,1),M(i,1),M(j,4),M(i,4)];
                n = n+1;
            else
                output(n,:) = [M(i,1),M(j,1),M(i,4),M(j,4)];
                n = n+1;
            end
        end
    end
end

display('M = [ e n1 n2 face], includes all elements, nodes and faces on
contact surface.')
display('output = [e1 e2 face1 face2], includes pairs of contacting
elements and their faces.')
display('Input "M" and "output" to see the corresponding results.')
```

```
% getnew2D_quad
% created by: Wes Hicks
% source: You Li "getnew3D_hex"
% output: for 2D quad mesh, find a new connectivity(all elements) and a
new
% node table, with updated node numbers. Note both outputs contains
only
% elements with changed node numbers or nodes with changed numbers.
% input: must run getsurf3D first in order to get corresponding inputs

% step 1: initiation
% num_n = total number of nodes
% num_n = size(node_table,1);
% size_n is fixed.
% num_n will change through updating new coordinates for nodes.
% size_n = num_n;
% change_group = [e n1 n2 face], includes all elements on fiber and on
% contact surface.
a = ismember(M(:,1),Efiber);    %finds which elements in M are a part
of the fiber
b = find(a);        %gives coordinates in M of Fiber elements
change_group = M(b,:);      %pulls those entries out of M  (entire row)
% create a copy of node table for use.
change_node_table = node_table;

% step 2: new connectivity
% ele_col = all elements that need change
ele_col = change_group(:,1);
% new_connectivity includes all elements that need change.
new_connectivity = connectivity(ele_col,:);
% the loop below locates all nodes that need change. based on the face.
for i = 1:size(change_group,1)
    face_num = change_group(i,4);       %face of the element we change
the connectvity for
    if face_num == 1;
        node_pos = face1 + match;       %nodes that coorespond to the
face that matched
    elseif face_num == 2;
        node_pos = face2 + match;
    elseif face_num == 3
        node_pos = face3 + match;
    elseif face_num == 4;
        node_pos = face4 + match;
    end
    change_nodes = new_connectivity(i,node_pos)';       %pulls out the
node id in the global node list
    % update new node coordinates in node table
    for j = 1:2
        if change_node_table(change_nodes(j),1)<= size_n;
%records that the node is duplicated
            change_node_table(change_nodes(j),1)=[num_n + 1];
%duplicates node number
            num_n = num_n + 1;
```

```
        end
    end
    % input the new node coordinates in new_connectivity
    new_connectivity(i,node_pos) = change_node_table(change_nodes,1)';
%amking the elements aware of the duplicated node numbers by putting in
duplicated nodes
end

% step 3: new node table
% extract new_node_table from change_node_table
node_col = change_group(:,2:3); % node_col = all nodes that need change
(*change 5 -> 3 for 2D)
node_num = unique(node_col); % node_num = all nodes in order, no
duplicates
new_node_table = change_node_table(node_num,:);
node_table= change_node_table;   % store duplicated number
%output
display('new_connectivity = [e n1 n2 n3 n4], including elements with
changed node coordinates.')
display('new_node_table = [n x y], including nodes with new
coordinates.')
display('Input "new_connectivity" and "output" to see the corresponding
results.')
```

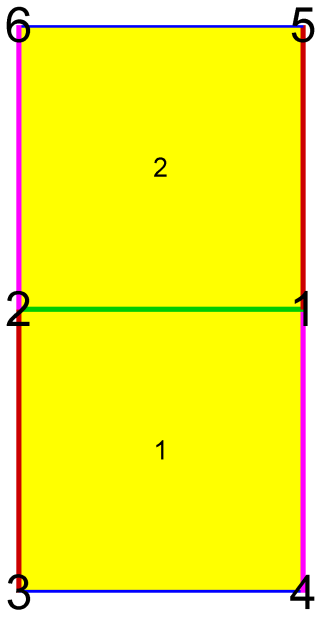
```
*Heading
** Job name: Composite_Basic_2D_1elem Model name: Model-1
** Generated by: Abaqus/CAE 6.12-3
*Preprint, echo=NO, model=NO, history=NO, contact=NO
**
** PARTS
**
*Part, name=Composite
*Node
      1,           1.,           0.
      2,           0.,           0.
      3,           0.,          -1.
      4,           1.,          -1.
      5,           1.,           1.
      6,           0.,           1.
*Element, type=CPS4R
1, 1, 2, 3, 4
2, 2, 1, 5, 6
*Nset, nset=Set-2
 1, 2, 5, 6
*Elset, elset=Set-2
 2,
*Nset, nset=Efiber1, generate
 1,  4,  1
*Elset, elset=Efiber1
 1,
*Nset, nset=interface1
 1, 2
*Elset, elset=interface1
 1, 2
** Section: Fiber
*Solid Section, elset=Efiber1, material=Fiber
,
** Section: Matrix
*Solid Section, elset=Set-2, material=Matrix
,
*End Part
**
**
** ASSEMBLY
**
*Assembly, name=Assembly
**
*Instance, name=Composite-1, part=Composite
*End Instance
**
*Nset, nset=Set-1, instance=Composite-1
 2, 3, 6
*Elset, elset=Set-1, instance=Composite-1
 1, 2
*Nset, nset=Set-2, instance=Composite-1
 1, 4, 5
*Elset, elset=Set-2, instance=Composite-1
 1, 2
*Nset, nset=Set-3, instance=Composite-1
```

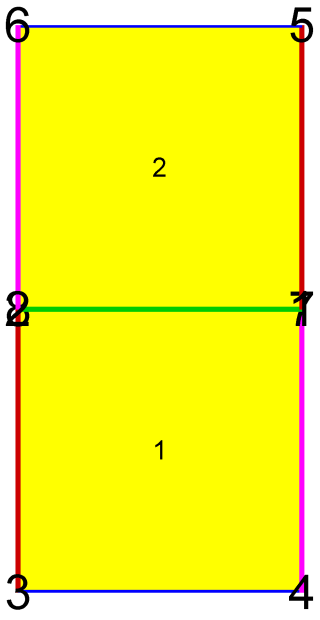
```
 3, 4
*Elset, elset=Set-3, instance=Composite-1
 1,
*Nset, nset=Set-4, instance=Composite-1
 5, 6
*Elset, elset=Set-4, instance=Composite-1
 2,
*End Assembly
**
** MATERIALS
**
*Material, name=Fiber
*Elastic
100., 0.2
*Material, name=Matrix
*Elastic
25., 0.3
**
** BOUNDARY CONDITIONS
**
** Name: BC-1 Type: Symmetry/Antisymmetry/Encastre
*Boundary
Set-3, YSYMM
** ----------------------------------------------------------------
**
** STEP: Step-1
**
*Step, name=Step-1
Displacement
*Static
1., 1., 1e-05, 1.
**
** BOUNDARY CONDITIONS
**
** Name: BC-2 Type: Displacement/Rotation
*Boundary
Set-4, 2, 2, 0.01
**
** OUTPUT REQUESTS
**
*Restart, write, frequency=0
**
** FIELD OUTPUT: F-Output-1
**
*Output, field, variable=PRESELECT
**
** HISTORY OUTPUT: H-Output-1
**
*Output, history, variable=PRESELECT
*End Step
```

```
Initial Input Reading

     node_table =

     1     1     0
     2     0     0
     3     0    -1
     4     1    -1
     5     1     1
     6     0     1

     connectivity =

     1     1     2     3     4
     2     2     1     5     6
```

```
During insertion

     new_connectivity =

     1     7     8     3     4


     new_node_table =

     7     1     0
     8     0     0

     M =

     1     1     2     1
     2     2     1     1

     output =

     2     1     1     1
```

```
Post insertion


     node_table =

     7      1      0
     8      0      0
     3      0     -1
     4      1     -1
     5      1      1
     6      0      1
     7      1      0
     8      0      0

     connectivity =

     1      7      8      3      4
     2      2      1      5      6
```





VITA

Wesley Hicks was born in Chattanooga, Tennessee to the parents of Kenneth and Elizabeth Hicks. He is the older of two siblings, younger sister, Sasha and himself. He began studying at the University of Tennessee in 2009 obtaining his B.S. in civil engineering in 2013. He then continued his education in 2014 at the same university for his M.S. of civil engineering studying under Dr. Timothy Truster and will finish this in May of 2015. While obtaining this degree he worked as a graduate research and teaching assistant. His coursework includes Finite Element Applications in Structural Engineering (CE 561), Advanced Structural Mechanics (CE 595), Engineering Fracture Mechanics (CE 595), Structural Dynamics (CE 565), Behavior of Reinforced Concrete (CE 574), Applications of Linear Algebra and Engineering Systems (CE 529), Prestressed Concrete (CE 573), and Masonry Design (CE 576).